

26th International Workshop on Principles of Diagnosis



D **X**
2015

Paris, France

August 31 - September 3, 2015

Editors: Yannick Pencolé, Louise Travé-Massuyès, Philippe Dague



Proceedings of the 26th International Workshop on Principles of Diagnosis (DX-15)

August 31-September 3, 2015

Paris, France

Yannick Pencolé, Louise Travé-Massuyès, Philippe Dague, Editors

Workshop Organization

Program Co-Chairs

Yannick Pencolé	LAAS-CNRS, Univ. Fédérale Toulouse, France
Louise Travé-Massuyès	LAAS-CNRS, Univ. Fédérale Toulouse, France
Philippe Dague	LRI, Université Paris-Sud, France

International Program Committee

Rui Abreu	PARC, USA
Jose Aguilar	Universidad de los Andes, Venezuela
Carlos Alonso	Universidad de Valladolid, Spain
Gautam Biswas	Vanderbilt University, USA
Anibal Bregon	Universidad de Valladolid, Spain
Luca Console	Università di Torino, Italy
Matthew Daigle	NASA Ames Research Center, USA
Johan de Kleer	PARC, USA
Michael Hofbauer	Joanneum Research, Austria
Alexander Feldman	PARC, USA
Gerhard Friedrich	Klagenfurt University, Austria
Alban Grastien	NICTA, Australia
Claudia Isaza	University of Antioquia, Medellín, Colombia
Meir Kalech	Ben-Gurion University of the Negev, Israel
Mattias Krysander	Linköping University, Sweden
Anastassia Küstenmacher	Bonn-Rhein-Sieg University of Applied Science, Germany
Ingo Pill	TU Graz, Austria
Gregory Provan	University College Cork, Ireland
Xavier Pucel	ONERA CERT, France
Martin Sachenbacher	LION Smart GmbH, Germany
Ramon Sarrate	Universitat Politècnica de Catalunya, Spain
Neal Snooke	Aberystwyth University, UK
Gerald Steinbauer	TU Graz, Austria
Peter Struss	TU München, Germany
Anna Sztyber	The Institute of Automatic and Robotics, Warsaw University of Technology
Gianluca Torta	Università di Torino, Italy
Franz Wotawa	TU Graz, Austria
Marina Zanella	Università degli Studi di Brescia, Italy

Additional Reviewers

Moussa Maiga	LAAS-CNRS, Univ. Fédérale Toulouse, France
Nathalie Barbosa Roa	LAAS-CNRS, Univ. Fédérale Toulouse, France
Euriell Le Corronc	LAAS-CNRS, Université Paul Sabatier, Univ. Fédérale Toulouse, France
Élodie Chanthery	LAAS-CNRS, INSA Toulouse, Univ. Fédérale Toulouse, France
Indranil Roychoudhury	SGT Inc, NASA Ames Research Center, USA

Workshop Organizing Committee

Yannick Pencolé	LAAS-CNRS, Univ. Fédérale Toulouse, France
Louise Travé-Massuyès	LAAS-CNRS, Univ. Fédérale Toulouse, France
Vincent Cocquempot	CRISTAL, Université Lille 1, France
Audine Subias	LAAS-CNRS, INSA Toulouse, Univ. Fédérale Toulouse, France
Nazih Mechbal	ENSAM Paris, France

Technical and Administrative Support

Christèle Mouclier	LAAS-CNRS, Toulouse, France
Régine Duran	LAAS-CNRS, Toulouse, France
Dominique Daurat	LAAS-CNRS, Toulouse, France
Fabienne Baduel	LAAS-CNRS, Toulouse, France
Bruno Birac	LAAS-CNRS, Toulouse, France
Stéphanie Saluden	Délégation Régionale CNRS, Toulouse, France
Régine Barthes	Délégation Régionale CNRS, Toulouse, France

Table of contents

Regular papers

A Divide-And-Conquer-Method for Computing Multiple Conflicts for Diagnosis <i>by Shechekotykhin Kostyantyn, Jannach Dietmar, Schmitz Thomas</i>	3
A Robust Alternative to Correlation Networks for Identifying Faulty Systems <i>by Traxler Patrick, Grill Tanja, Gomez Pablo</i>	11
Applied multi-layer clustering to the diagnosis of complex agro-systems <i>by Roux Elisa, Travé-Massuyès Louise, Le Lann Marie-Véronique</i>	19
A Bayesian Framework for Fault diagnosis of Hybrid Linear Systems <i>by Zhou Gan, Biswas Gautam, Feng Wenquan, Zhao Hongbo, Guan Xiumei</i>	27
ADS2 : Anytime Distributed Supervision of Distributed Systems that Face Unreliable or Costly Communication <i>by Herpson Cédric, El Fallah Seghrouchni Amal, Corruble Vincent</i>	35
Data Driven Modeling for System-Level Condition Monitoring on Wind Power Plants <i>by Eickmeyer Jens, Li Peng, Givehchi Omid, Pethig Florian, Niggemann Oliver</i>	43
Using Incremental SAT for Testing Diagnosability of Distributed DES <i>by Ibrahim Hassan, Dague Philippe, Simon Laurent</i>	51
Improving Fault Isolation and Identification for Hybrid Systems with Hybrid Possible Conflicts <i>by Bregon Anibal, Alonso-Gonzalez Carlos, Pulido Belarmino</i>	59
State estimation and fault detection using box particle filtering with stochastic measurements <i>by Blesa Joaquim, Le Gall Françoise, Jaubertie Carine, Travé-Massuyès Louise</i>	67
Minimal Structurally Overdetermined Sets Selection for Distributed Fault Detection <i>by Khorasgani Hamed, Biswas Gautam, Jung Daniel</i>	75
Condition-based Monitoring and Prognosis in an Error-Bounded Framework <i>by Travé-Massuyès Louise, Pons Renaud, Ribot Pauline, Pencolé Yannick, Jaubertie Carine</i>	83
Configuration as Diagnosis: Generating Configurations with Conflict-Directed A* - An Application to Training Plan Generation - <i>by Grigoleit Florian, Struss Peter</i>	91
Decentralised fault diagnosis of large-scale systems: Application to water transport networks <i>by Puig Vicenç, Ocampo-Martinez Carlos</i>	99

Self-Healing as a Combination of Consistency Checks and Conformant Planning Problems <i>by Grastien Alban</i>	105
Implementing Troubleshooting with Batch Repair <i>by Stern Roni, Kalech Meir, Shinitzky Hilla</i>	113
Formulating Event-Based Critical Observations in Diagnostic Problems <i>by Christopher Cody, Grastien Alban</i>	119
A Framework For Assessing Diagnostics Model Fidelity <i>by Provan Gregory, Feldman Alexander</i>	127
Posters	
A General Process Model: Application to Unanticipated Fault Diagnosis <i>by Wang Jiongqi, He Zhangming, Zhou Haiyin, Li Shuxing</i>	137
A SCADA Expansion for Leak Detection in a Pipeline <i>by Carrera Rolando, Verde Cristina, Cayetano Raúl</i>	145
Automatic Model Generation to Diagnose Autonomous Systems <i>by Santos Simón Jorge, Mühlbacher Clemens, Steinbauer Gerald</i>	153
Methodology and Application of Meta-Diagnosis on Avionics Test Benches <i>by Cossé Ronan, Berdjag Denis, Piechowiak Sylvain, Duvivier David, Gaurel Christian</i>	159
SAT-Based Abductive Diagnosis <i>by Koitz Roxane, Wotawa Franz</i>	167
Fault Tolerant Control for a 4-Wheel Skid Steering Mobile Robot <i>by Furlas George, Karras George, Kyriakopoulos Kostas</i>	177
Data-Driven Monitoring of Cyber-Physical Systems Leveraging on Big Data and the Internet-of-Things for Diagnosis and Control <i>by Niggemann Oliver, Biswas Gautam, Kinnebrew John, Khorasgani Hamed, Volgmann Sören, Bunte Andreas</i>	185
Diagnosing Advanced Persistent Threats: A Position Paper <i>by Abreu Rui, Bobrow Daniel, Eldardiry Hoda, Feldman Alexander, Hanley John, Honda Tomonori, De Kleer Johan, Perez Alexandre, Archer Dave, Burke David</i>	193
A Structural Model Decomposition Framework for Hybrid Systems Diagnosis <i>by Daigle Matthew, Bregon Anibal, Roychoudhury Indranil</i>	201
Device Health Estimation by Combining Contextual Control Information with Sensor Data <i>by Honda Tomonori, Liao Linxia, Eldardiry Hoda, Saha Bhaskar, Abreu Rui, Pavel Radu, Iverson Jonathan</i>	209

On the Learning of Timing Behavior for Anomaly Detection in Cyber-Physical Production Systems <i>by Maier Alexander, Niggemann Oliver, Eickmeyer Jens</i>	217
The Case for a Hybrid Approach to Diagnosis: A Railway Switch <i>by Matei Ion, Ganguli Anurag, Honda Tomonori, De Kleer Johan</i>	225
Design of PD observer-based fault estimator using a descriptor approach <i>by Krokavec Dusan, Filasova Anna, Liscinsky Pavol, Serbak Vladimir</i>	235
Chronicle based alarm management in startup and shutdown stages <i>by Vasquez John William, Travé-Massuyès Louise, Subias Audine, Jimenez Fernando, Agudelo Carlos</i>	241
Data-Augmented Software Diagnosis <i>by Mishali Amir, Stern Roni, Kalech Meir</i>	247
Faults isolation and identification of Heat-exchanger/ Reactor with parameter uncertainties <i>by Zhang Mei, Dahhou Boutaïeb, Cabassud Michel, Li Ze-Tao</i>	253
LPV subspace identification for robust fault detection using a set-membership approach: Application to the wind turbine benchmark <i>by Chouïref Houda, Boussaid Boumedyen, Abdelkrim Mohamed Naceur, Puig Vicenç, Aubrun Christophe</i>	261
Processing measure uncertainty into fuzzy classifier <i>by Monrousseau Thomas, Travé-Massuyès Louise, Le Lann Marie-Véronique</i>	269
Tools/Benchmarks	
Random generator of k-diagnosable discrete event systems <i>by Pencolé Yannick</i>	277
HyDiag: extended diagnosis and prognosis for hybrid systems <i>by Chanthery Elodie, Pencolé Yannick, Ribot Pauline, Travé-Massuyès Louise</i>	281

Regular papers

A Divide-And-Conquer-Method for Computing Multiple Conflicts for Diagnosis

Kostyantyn Shchekotykhin¹ and Dietmar Jannach² and Thomas Schmitz²

¹Alpen-Adria University Klagenfurt, Austria

e-mail: kostyantyn.shchekotykhin@aau.at

²TU Dortmund, Germany

e-mail: {firstname.lastname}@tu-dortmund.de

Abstract

In classical hitting set algorithms for Model-Based Diagnosis (MBD) that use on-demand conflict generation, a single conflict is computed whenever needed during tree construction. Since such a strategy leads to a full “restart” of the conflict-generation algorithm on each call, we propose a divide-and-conquer algorithm called MERGEXPLAIN which efficiently searches for multiple conflicts during a single call.

The design of the algorithm aims at scenarios in which the goal is to find a few leading diagnoses and the algorithm can – due to its non-intrusive design – be used in combination with various underlying reasoners (*theorem provers*). An empirical evaluation on different sets of benchmark problems shows that our proposed algorithm can lead to significant reductions of the required diagnosis times when compared to a “one-conflict-at-a-time” strategy.

1 Introduction

In Model-Based Diagnosis (MBD), the concept of *conflicts* describes parts of a system which – given a set of observations – cannot all work correctly. Besides MBD, the calculation of minimal conflicts is a central task in a number of other AI approaches [1]. Reiter [2] showed that the minimal *hitting sets* of conflicts correspond to *diagnoses*, where a diagnosis is a possible explanation why a system’s observed behavior differs from its expected behavior. He used this property for the computation of diagnoses in the breadth-first hitting set tree (HS-tree) diagnosis algorithm.

Over time, the principle of this MBD approach was used for a number of different diagnosis problems such as electronic circuits, hardware descriptions in VHDL, program specifications, ontologies, and knowledge-based systems [3; 4; 5; 6; 7]. A reason for the broad utilization of hitting set approaches is that its principle does not depend on the underlying knowledge representation and reasoning technique, because only a general *Theorem Prover* (TP) – a component that returns conflicts – is needed.

The implementation of a TP can be done in different ways. First, the conflict detection can be implemented as a reasoning task, e.g., by modifying a consistency checking algorithm [8; 9]. Second, “non-intrusive” conflict detection techniques can be used with a variety of reasoning

approaches, since they require only a very limited reasoning functionality like consistency or entailment checking without knowing the internals of the reasoning algorithm. Such methods can benefit from the newest improvements in reasoning algorithms, such as incremental solving, heuristics, learning strategies, etc., without any modifications.

A non-intrusive conflict detection algorithm which has shown to be very efficient in different application scenarios is Junker’s QUICKXPLAIN [10] (QXP for short) which was designed to find a single *minimal* conflict based on a divide-and-conquer strategy. The algorithm was originally developed in the context of constraint problems, but since its method is independent of the underlying reasoner, it was used in several of the hardware and software diagnosis approaches mentioned above.

In many classical hitting set based approaches, conflicts are computed individually with QXP during HS-tree construction when they are required, as in many domains not all conflicts are known in advance [11]. This, however, has the effect that QXP has to be “restarted” with a slightly different configuration whenever a new conflict is needed.

In this paper, we propose MERGEXPLAIN (MXP for short), a divide-and-conquer algorithm which searches for multiple conflicts during a single decomposition run. Our method is built upon QXP and is therefore also non-intrusive. The basic idea behind MXP is that (a) the early identification of multiple conflicts can speed up the overall diagnosis process, e.g., due to better conflict “reuses” [2], and that (b) we can identify additional conflicts faster when we decompose the original components into smaller subsets with the divide-and-conquer strategy of MXP.

The paper is organized as follows. After a problem characterization in Section 2, we present the details of MXP in Section 3 and discuss the properties of the algorithm. Section 4 presents the results of an extensive empirical evaluation using various diagnosis benchmark problems. Previous work is finally discussed in Section 5.

2 Preliminaries

2.1 The Diagnosis Problem

We use the definitions of [2] to characterize a system, diagnoses, and conflicts.

Definition 1 (System). *A system is a pair (SD, COMPS) where SD is a system description (a set of logical sentences) and COMPS represents the system’s components (a finite set of constants).*

A diagnosis problem arises when a set of logical sentences OBS, called *observations*, is inconsistent with the normal behavior of the system (SD, COMPS). The correct behavior is represented in SD with an “abnormal” predicate AB/1. That is, for any component $c_i \in \text{COMPS}$ the literal $\neg \text{AB}(c_i)$ represents the assumption that the component c_i behaves correctly.

Definition 2 (Diagnosis). *Given a diagnosis problem (SD, COMPS, OBS), a diagnosis is a minimal set $\Delta \subseteq \text{COMPS}$ such that $\text{SD} \cup \text{OBS} \cup \{\text{AB}(c) \mid c \in \Delta\} \cup \{\neg \text{AB}(c) \mid c \in \text{COMPS} \setminus \Delta\}$ is consistent.*

A diagnosis therefore corresponds to a minimal subset of the system components which, if assumed to be faulty (and thus behave abnormally) explain the system’s behavior, i.e., are consistent with the observations.

Two general classes of MBD algorithms exist. One relies on direct problem encodings and the aim is often to find one diagnosis quickly, see [12; 13; 14]. The other class relies on the computation of conflicts and their hitting sets (see next section). Such diagnosis algorithms are often used when the goal is to find *multiple* or all minimal diagnoses. In the context of our work, techniques of the second class can immediately profit when the conflict generation process is done more efficiently.

2.2 Diagnoses as Hitting Sets

Finding all minimal diagnoses corresponds to finding all minimal hitting sets (HS) of all existing conflicts [2].

Definition 3 (Conflict). *A conflict CS for (SD, COMPS, OBS) is a set $\{c_1, \dots, c_k\} \subseteq \text{COMPS}$ such that $\text{SD} \cup \text{OBS} \cup \{\neg \text{AB}(c_i) \mid c_i \in \text{CS}\}$ is inconsistent.*

Assuming that all components of a conflict work correctly therefore contradicts the observations. A conflict CS is *minimal*, if no proper subset of CS is also a conflict.

To find the set of *all minimal diagnoses* for a given problem, [2] proposed a breadth-first HS-tree algorithm with tree pruning and conflict reuse. A correction to this algorithm was proposed by Greiner et al. which uses a directed acyclic graph (DAG) instead of the tree to correctly deal with non-minimal conflicts [15]. Our work, however, does not depend on this correction as QXP as well as our proposed MXP method always return minimal conflicts. Apart from this, a number of algorithmic variations were suggested in the literature which, for example, use problem-specific heuristics [16], a greedy search algorithm, or apply parallelization techniques [17], see also [18] for an overview.

2.3 QUICKXPLAIN (QXP)

QXP was developed in the context of inconsistent constraint satisfaction problems (CSPs) and the computation of explanations. E.g., in case of an overconstrained CSP, the problem consists in determining a minimal set of constraints which causes the CSP to become unsolvable for the given inputs. A simplified version of QXP [10] is shown in Algorithm 1. The rough idea of QXP is to apply a recursive procedure which relaxes the input set of faulty constraints \mathcal{C} by partitioning it into two sets \mathcal{C}_1 and \mathcal{C}_2 (line 6). If \mathcal{C}_1 is a conflict the algorithm continues partitioning \mathcal{C}_1 in the next recursive call. Otherwise, i.e., if the last partitioning has split all conflicts in \mathcal{C} , the algorithm extracts a conflict from the sets \mathcal{C}_1 and \mathcal{C}_2 . This way, QXP finally identifies single constraints which are inconsistent with the remaining consistent set of constraints and the background theory.

Algorithm 1: QUICKXPLAIN(\mathcal{B}, \mathcal{C})

Input: \mathcal{B} : background theory, \mathcal{C} : the set of possibly faulty constraints
Output: A minimal conflict $\text{CS} \subseteq \mathcal{C}$
1 if *isConsistent*($\mathcal{B} \cup \mathcal{C}$) then return ‘no conflict’;
2 else if $\mathcal{C} = \emptyset$ then return \emptyset ;
3 return GETCONFLICT($\mathcal{B}, \mathcal{B}, \mathcal{C}$)
function GETCONFLICT ($\mathcal{B}, D, \mathcal{C}$)
4 if $D \neq \emptyset \wedge \neg \text{isConsistent}(\mathcal{B})$ then return \emptyset ;
5 if $|\mathcal{C}| = 1$ then return \mathcal{C} ;
6 Split \mathcal{C} into disjoint, non-empty sets \mathcal{C}_1 and \mathcal{C}_2
7 $D_2 \leftarrow \text{GETCONFLICT}(\mathcal{B} \cup \mathcal{C}_1, \mathcal{C}_1, \mathcal{C}_2)$
8 $D_1 \leftarrow \text{GETCONFLICT}(\mathcal{B} \cup D_2, D_2, \mathcal{C}_1)$
9 return $D_1 \cup D_2$

Theorem 1 ([10]). *Let \mathcal{B} be a background theory, i.e., a set of constraints considered as correct, and \mathcal{C} be a set of possibly faulty constraints. Then, QUICKXPLAIN always terminates. If $\mathcal{B} \cup \mathcal{C}$ is consistent it returns ‘no conflict’. Otherwise, it returns a minimal conflict CS.*

2.4 Using QXP During HS-Tree Construction

Assume that MBD is applied to find an error in the definition of a CSP. The CSP comprises the set of possibly faulty constraints \mathcal{C} . These are the elements of COMPS. The system description SD corresponds to the semantics of the constraints in \mathcal{C} . Finally, the observations OBS are encoded as unary constraints and are added to the background theory \mathcal{B} . During the HS-tree construction, QXP is called whenever a new node is created and no conflict reuse is possible. As a result, QXP can either return *one minimal conflict*, which can be used to label the new node, or return ‘no conflict’, which would mean that a diagnosis is found at the tree node. Note that QXP can be used with other algorithms, e.g., preference-based search [19] or boolean search [20], in the same way as with the HS-tree algorithm.

3 MERGEXPLAIN (MXP): Algorithm Details

3.1 General Considerations

The pseudo-code of MXP, which unlike QXP can return multiple conflicts at a time, is given in Algorithm 2. MXP, like QXP, is generally applicable to a variety of problem domains. The mapping to the terminology used in MBD (SD, COMPS, OBS) is straightforward as discussed in the previous section. In the following, we will use the notation and symbols from [10], e.g., \mathcal{C} or \mathcal{B} , and constraints as a knowledge representation formalism.

Note that there are applications of MBD in which the function *isConsistent* has to be “overwritten” to take the specifics of the underlying knowledge representation and reasoning system into account. The ontology debugging approach presented in [7] for example extends *isConsistent* with the verification of entailments of a logical theory. MXP can be used in such scenarios after the corresponding adaptation of the implementation of *isConsistent*.

Furthermore, MXP can be easily extended for cases in which the MBD approach has to support the specification of (multiple) test cases, i.e., sets of formulas that must be

consistent or inconsistent with the system description, e.g., [21; 22].

3.2 Algorithm Rationale

MXP (Algorithm 2) accepts two sets of constraints as inputs, \mathcal{B} as the assumed-to-be-correct set of background constraints and \mathcal{C} , the possibly faulty components/constraints.

In case $\mathcal{C} \cup \mathcal{B}$ is inconsistent, MXP returns a *set of minimal conflicts* Γ by calling the recursive function `FINDCONFLICTS` in line 3. This function again accepts \mathcal{B} and \mathcal{C} as an input and returns a tuple $\langle \mathcal{C}', \Gamma \rangle$, where Γ is a set of minimal conflicts and $\mathcal{C}' \subset \mathcal{C}$ is a set of constraints that does not contain any conflicts, i.e., $\mathcal{B} \cup \mathcal{C}'$ is consistent.

The logic of `FINDCONFLICTS` is similar to QXP in that we decompose the problem into two parts in each recursive call (lines 7–9). Differently from QXP, however, we look for conflicts in both splits \mathcal{C}_1 and \mathcal{C}_2 independently and then combine the conflicts that are eventually found in the two halves (line 10)¹. If there is, e.g., a conflict in the first part and one in the second, `FINDCONFLICTS` will find them independently from each other. Of course, there might also be conflicts in \mathcal{C} whose elements are spread across both \mathcal{C}_1 and \mathcal{C}_2 , that is, the set $\mathcal{C}'_1 \cup \mathcal{C}'_2 \cup \mathcal{B}$ is inconsistent. This situation is addressed in lines 11–15. The computation of a minimal conflict is done by two calls to `GETCONFLICT` (Algorithm 1). In the first call this function returns a minimal set $X \subseteq \mathcal{C}'_1$ such that $X \cup \mathcal{C}'_2 \cup \mathcal{B}$ is a conflict (line 12). In line 13, we then look for a subset of \mathcal{C}'_2 , say Y , such that $Y \cup X$ corresponds to a minimal conflict CS . The latter is added to Γ (line 15). In order to restore the consistency of $\mathcal{C}'_1 \cup \mathcal{C}'_2 \cup \mathcal{B}$ we have to remove at least one element $\alpha \in CS$ from either \mathcal{C}'_1 or \mathcal{C}'_2 . Therefore, in line 14 the algorithm removes $\alpha \in X \subseteq CS$ from \mathcal{C}'_1 .

Note that MXP allows us to use different split functions in line 7. In our default implementation we use a function that splits the set of constraints \mathcal{C} into two equal parts, i.e., $split(n) = n/2$, where $|\mathcal{C}| = n$. In the worst case this split function results in a perfect binary tree with n leaves. Consequently, the total number of nodes is $2n - 1$, which correspond to $2(2n - 1)$ consistency checks (lines 5 and 11). Other split functions might result in a similar number of consistency checks in the worst case as well, since in any case MXP has to traverse a binary tree with n leaves. For instance, the function $split(n) = n - 1$ results in a tree with one branch of the depth $n - 1$ and n leaves, that is, $2n - 1$ nodes to traverse. However, while the number of nodes to explore might be comparable, the important point is that the computational costs for the individual consistency checks can be different depending on the splitting strategy. Under the reasonable assumption that consistency checking of smaller sets of constraints requires less time, the function $split(n) = n/2$ allows MXP to split the set of constraints faster, thus, improving the overall runtime.

3.3 Example

Consider a CSP consisting of six constraints $\{c_0, \dots, c_5\}$. The constraint c_0 is considered correct, i.e., $\mathcal{B} = \{c_0\}$. Let $\{\{c_0, c_1, c_3\}, \{c_0, c_5\}, \{c_2, c_4\}\}$ be the set of minimal conflicts. Algorithm 2 proceeds as follows (Figure 1).

Since the input CSP ($\mathcal{B} \cup \mathcal{C}$) is not consistent, the algorithm enters the recursion. In the first step, `FINDCONFLICTS` partitions the input set (line 7) into the two subsets

¹The calls in line 8 and 9 can in fact be executed in parallel.

Algorithm 2: MERGEXPLAIN(\mathcal{B}, \mathcal{C})

Input: \mathcal{B} : background theory, \mathcal{C} : the set of possibly faulty constraints

Output: Γ , a set of minimal conflicts

```

1 if  $\neg isConsistent(\mathcal{B})$  then return ‘no solution’;
2 if  $isConsistent(\mathcal{B} \cup \mathcal{C})$  then return  $\emptyset$ ;
3  $\langle \_, \Gamma \rangle \leftarrow FINDCONFLICTS(\mathcal{B}, \mathcal{C})$ 
4 return  $\Gamma$ ;
    
```

function `FINDCONFLICTS` (\mathcal{B}, \mathcal{C}) **returns** tuple $\langle \mathcal{C}', \Gamma \rangle$

```

5   if  $isConsistent(\mathcal{B} \cup \mathcal{C})$  then return  $\langle \mathcal{C}, \emptyset \rangle$ ;
6   if  $|\mathcal{C}| = 1$  then return  $\langle \emptyset, \{\mathcal{C}\} \rangle$ ;
7   Split  $\mathcal{C}$  into disjoint, non-empty sets  $\mathcal{C}_1$  and  $\mathcal{C}_2$ 
8    $\langle \mathcal{C}'_1, \Gamma_1 \rangle \leftarrow FINDCONFLICTS(\mathcal{B}, \mathcal{C}_1)$ 
9    $\langle \mathcal{C}'_2, \Gamma_2 \rangle \leftarrow FINDCONFLICTS(\mathcal{B}, \mathcal{C}_2)$ 
10   $\Gamma \leftarrow \Gamma_1 \cup \Gamma_2$ ;
11  while  $\neg isConsistent(\mathcal{C}'_1 \cup \mathcal{C}'_2 \cup \mathcal{B})$  do
12     $X \leftarrow GETCONFLICT(\mathcal{B} \cup \mathcal{C}'_2, \mathcal{C}'_1)$ 
13     $CS \leftarrow X \cup GETCONFLICT(\mathcal{B} \cup X, \mathcal{C}'_2)$ 
14     $\mathcal{C}'_1 \leftarrow \mathcal{C}'_1 \setminus \{\alpha\}$  where  $\alpha \in X$ 
15     $\Gamma \leftarrow \Gamma \cup \{CS\}$ 
16  return  $\langle \mathcal{C}'_1 \cup \mathcal{C}'_2, \Gamma \rangle$ 
    
```

$\mathcal{C}_1 = \{c_1, c_2, c_3\}$ and $\mathcal{C}_2 = \{c_4, c_5\}$ and provides them as input to the recursive calls (lines 8 and 9). In the next level of the recursion – marked with ② in Figure 1 – the input is found to be inconsistent (line 5) and again partitioned into two sets (line 7). In the subsequent calls, ③ and ④, the two input sets are found to be consistent (line 5) and, therefore, the set $\{c_1, c_2, c_3\}$ has to be analyzed using `GETCONFLICT` (lines 12 and 13) defined in Algorithm 1. `GETCONFLICT` returns the conflict $\{c_1, c_3\}$, which is added to Γ . Finally, `FINDCONFLICTS` removes c_1 from the set \mathcal{C}'_1 and returns the tuple $\langle \{c_2, c_3\}, \{\{c_1, c_3\}\} \rangle$ to ①.

Next, the “right-hand” part of the initial input, the set $\mathcal{C}_2 = \{c_4, c_5\}$, is provided as input to `FINDCONFLICTS` ⑤. Since \mathcal{C}_2 is inconsistent, it is partitioned into two sets $\mathcal{C}_1 = \{c_4\}$ and $\mathcal{C}_2 = \{c_5\}$. The first recursive call ⑥ returns $\langle \{c_4\}, \emptyset \rangle$ since the input is consistent. The second call ⑦, in contrast, finds that the input comprises only one constraint that is inconsistent with the background theory \mathcal{B} . Therefore, it returns $\langle \emptyset, \{\{c_5\}\} \rangle$ in line 6. Since $\mathcal{C}'_1 \cup \mathcal{C}'_2 = \{c_4\} \cup \emptyset$ is consistent with \mathcal{B} , `FINDCONFLICTS` ⑤ returns $\langle \{c_4\}, \{\{c_5\}\} \rangle$ to ①.

Finally, in ① the set of constraints $\mathcal{C}'_1 \cup \mathcal{C}'_2 = \{c_2, c_3\} \cup \{c_4\}$ is found to be inconsistent with \mathcal{B} (line 11) and `GETCONFLICT` is called. The method returns the conflict $\{c_2, c_4\}$ and c_2 is removed from \mathcal{C}'_1 . The resulting set $\{c_3, c_4\}$ is consistent and MXP returns $\Gamma = \{\{c_1, c_3\}, \{c_5\}, \{c_2, c_4\}\}$.

3.4 Properties of MERGEXPLAIN

Theorem 2. *Given a background theory \mathcal{B} and a set of constraints \mathcal{C} , Algorithm 2 always terminates and returns*

- ‘no solution’, if \mathcal{B} is inconsistent,
- \emptyset , if $\mathcal{B} \cup \mathcal{C}$ is consistent, and
- a set of minimal conflicts Γ , otherwise.

Proof. In the first case, given an inconsistent background theory \mathcal{B} , the algorithm terminates in line 1 and returns ‘no solution’. In the second case, if the set $\mathcal{B} \cup \mathcal{C}$ is consistent,

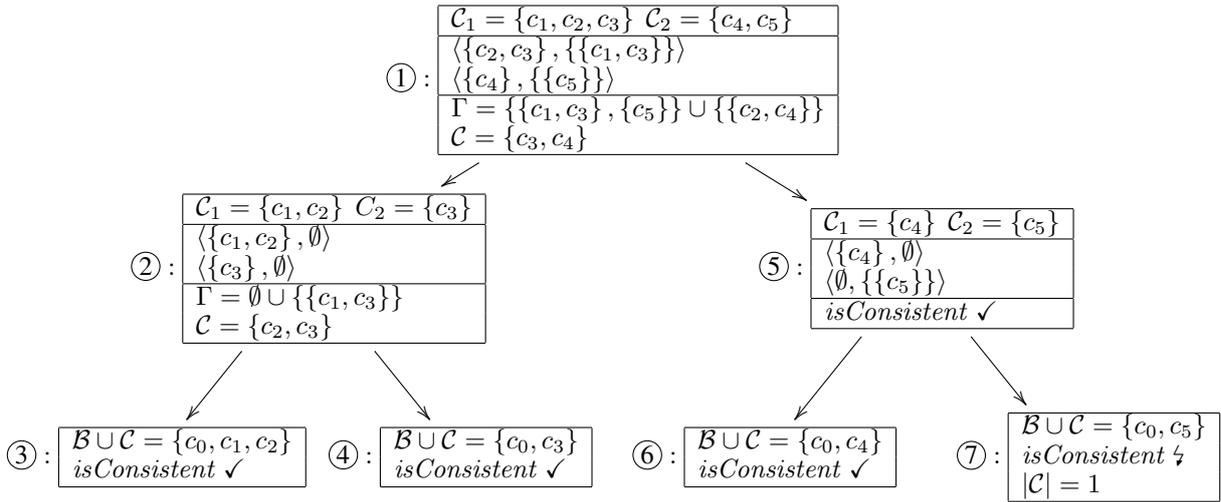


Figure 1: MERGEXPLAIN recursion tree. Each node shows values of selected variables in the FINDCONFLICTS function.

then no subset of \mathcal{C} is a conflict. MXP terminates and returns \emptyset .

Finally, if the set $\mathcal{B} \cup \mathcal{C}$ is inconsistent, the algorithm enters the recursion in line 3. The function FINDCONFLICTS in each call partitions the input set \mathcal{C} into two sets \mathcal{C}_1 and \mathcal{C}_2 . The partitioning continues until either the found set of constraints \mathcal{C} is consistent or a singleton conflict is detected. Therefore, every recursion branch ends after at most $\log |\mathcal{C}| - 1$ calls. Consequently, FINDCONFLICTS terminates if the conflict detection loop in lines 11–15 always terminates.

We consider two situations. If the set $\mathcal{C}'_1 \cup \mathcal{C}'_2$ is consistent with \mathcal{B} , the loop terminates. Otherwise, in each iteration at least one conflict in the set $\mathcal{C}'_1 \cup \mathcal{C}'_2$ is resolved. This fact follows from Theorem 1 according to which the function GETCONFLICT in Algorithm 1 always returns a minimal conflict if the input parameter \mathcal{C} is inconsistent with \mathcal{B} . Since the number of conflicts is finite and in each iteration one of the conflicts in $\mathcal{C}'_1 \cup \mathcal{C}'_2$ is resolved in line 14, the loop will terminate after a finite number of iterations. Consequently, Algorithm 2 terminates and returns a set of minimal conflicts Γ . \square

Corollary 1. *Given a consistent background theory \mathcal{B} and a set of inconsistent constraints \mathcal{C} , Algorithm 2 always returns a set of minimal conflicts Γ such that there exists a diagnosis $\Delta_i \subseteq \bigcup_{CS_i \in \Gamma} CS_i$.*

The proof follows from the fact that – similar to the HS-tree algorithm – a conflict is resolved by removing one of its elements from the set of constraints \mathcal{C}_1 in line 14. The loop in line 11 guarantees that every conflict $CS_i \in \mathcal{C}'_1 \cup \mathcal{C}'_2$ is hit. Consequently, FINDCONFLICTS hits every conflict in the input set \mathcal{C} and the set of constraints $\{\alpha_1, \dots, \alpha_n\}$ removed in every call of line 14 is a superset or equal to a diagnosis of the problem. The construction of at least one diagnosis from the found conflicts Γ can be done by the HS-tree algorithm.

MXP can in principle use several strategies for the resolution of conflicts in line 14. The strategy used in MXP by default is conservative and allows us to find several conflicts at once. Two additional *elimination strategies* can be used in line 14: (1) $\mathcal{C}'_1 \leftarrow \mathcal{C}'_1 \setminus X$ or (2) $\mathcal{C}'_1 \leftarrow \mathcal{C}'_1 \setminus CS$ and $\mathcal{C}'_2 \leftarrow \mathcal{C}'_2 \setminus CS$. These more aggressive strategies result in a smaller number of conflicts returned by MXP in each call but each call returns the results faster. However, for the latter

strategies MXP might not return enough minimal conflicts for the HS-tree algorithm to compute at least one diagnosis. For instance, let $\{\{c_1, c_2\}, \{c_1, c_3\}, \{c_2, c_4\}\}$ be the set of all minimal conflicts. If MXP returns $\Gamma = \{\{c_1, c_2\}\}$, which is one of the possible valid outputs, then the HS-tree algorithm fails to find a diagnosis as $\{c_1, c_2\}$ must be hit twice. In this case, the HS-tree algorithm must call MXP multiple times or another algorithm for diagnosis computation must be used, e.g., [23].

Corollary 2. *Algorithm 2 is sound, i.e., every set $CS \in \Gamma$ is a minimal conflict, and complete, i.e., given a diagnosis problem for which at least one minimal conflict exists, Algorithm 2 returns $\Gamma \neq \emptyset$.*

The soundness of the algorithm follows from Theorem 1, since the conflict computation of MXP uses the GETCONFLICT function of QXP. The completeness is shown as follows: Let \mathcal{B} be a background theory and \mathcal{C} a set of faulty constraints, i.e., $\mathcal{B} \cup \mathcal{C}$ is inconsistent. Assume MXP returns $\Gamma = \emptyset$, i.e., no minimal conflicts are found. However, this is impossible, since the loop in line 11 will never end. Consequently, Algorithm 2 will not terminate which contradicts our assumption. Hence, it holds that MXP is complete.

4 Evaluation

We have evaluated the efficiency of computing multiple conflicts at once with MXP using a number of different diagnosis benchmark problems. As a baseline for the comparison, we use QXP as a Theorem Prover, which returns exactly one minimal conflict at a time. Furthermore, we made measurements with a variant of MXP called PMXP in which the lines 8 and 9 are executed in parallel in two threads on a multi-core computer.

4.1 Benchmark Problems

We made experiments with different benchmark problems. First, we used the five first systems of the DX Competition (DXC) 2011 Synthetic Track. For each system, 20 scenarios are specified in which artificial faults were injected. In addition, we made experiments with a number of CSP problems from the CSP solver competition 2008 and several CSP encodings of real-world spreadsheets. The injection of faults was done in the same way as in [17].

In addition to these benchmark problems, we developed a diagnosis problem generator, which can be configured to generate (randomized) diagnosis problems with varying characteristics, e.g., with respect to the number of conflicts, their size, or their position in the system description SD.

4.2 Measurement Method

We implemented all algorithms in a Java-based MBD framework, which uses Choco as an underlying constraint solver, see [17]. The experiments were conducted on a laptop computer (Intel i7, 8GB RAM). As a performance indicator we use the time needed (“wall clock”) for computing one or more diagnoses. The reported running time numbers are averages of 100 runs of each problem setting that were done to avoid random effects. We furthermore randomly shuffled the ordering of the constraints in each run to avoid effects that might be caused by a certain positioning of the conflicts in SD. For the evaluation of MXP we used the most aggressive elimination strategy (2) as described in Section 3.4.

Since MXP can return more than one conflict at a time, it is expected to be particularly useful when the problem is to find a set of n first (leading) diagnoses, e.g., in the context of applying MBD to software debugging [5; 7]. We therefore report the results for the tasks “find-one-diagnosis” (as an extreme case) and “find- n -diagnoses”.

The task of finding a single diagnosis is comparably simple and “direct encodings” or algorithms like INVERSE-QUICKXPLAIN [23] are typically more efficient for this task than the HS-tree algorithm. For instance, INVERSE-QUICKXPLAIN requires only $O(|\Delta| \log(|C|/|\Delta|))$ calls to TP. If TP can check the consistency in polynomial time, then one diagnosis can also be computed efficiently. The problem of finding more than one diagnosis is very different and computationally challenging, because deciding whether an additional diagnosis exists is NP-complete [24]. In such settings the application of methods that are highly efficient for finding one diagnosis is not always advantageous. For instance, the evaluation presented in [14] demonstrates this fact for direct encodings. Therefore a comparison of our algorithm with approaches for the “find-one-diagnosis” problem is beyond the scope of our work, as we are interested in problem settings in which the HS-tree algorithm is favorable and no assumptions about the underlying reasoner should be made. When the task is to find all diagnoses, the performance of MXP is similar to that of QXP as all existing conflicts have to be determined.

4.3 Results

DXC Benchmark Problems Table 1 shows the characteristics of the analyzed and CSP-encoded DXC benchmark problems. Since we consider multiple scenarios per system, the number of faults and the corresponding diagnoses can vary strongly across the experiment runs.

Table 2 shows the observed performance gains when using MXP instead of QXP in terms of absolute numbers (ms) and the relative improvement. For the problem of finding the first 5 diagnoses (QXP-5/MXP-5), the observed improvements range from 15% up to 45%. For the extreme case of finding one single diagnosis, even slightly stronger improvements can be observed. The improvements when searching for, e.g., the first 10 diagnoses are similar for cases in which significantly more than 10 diagnoses actually exist.

System	#C	#V	#F	#D	$\overline{\#D}$	$\overline{ \Delta }$	$\overline{\#Cf}$	$\overline{ Cf }$
74182	21	28	4 - 5	30 - 300	139	4.66	4.9	3.3
74L85	35	44	1 - 3	1 - 215	66.4	3.13	5.9	8.3
74283	38	45	2 - 4	180 - 4,991	1,232.7	4.42	78.8	16.1
74181	67	79	3 - 6	10 - 3,828	877.8	4.53	7.8	10.6
c432	162	196	2 - 5	1 - 6,944	1,069.3	3.38	15.0	19.8

Table 1: Characteristics of selected DXC benchmarks. #C: number of constraints, #V: number of variables, #F: number of injected faults, #D: range of the number of diagnoses, $\overline{\#D}$: average number of the diagnoses, $\overline{|\Delta|}$: average diagnosis size, $\overline{\#Cf}$: average number of conflicts, $\overline{|Cf|}$: average conflict size.

System	QXP-5 [ms]	MXP-5 Improv.	QXP-1 [ms]	MXP-1 Improv.
74182	17.0	19%	17.0	19%
74L85	20.9	15%	16.1	19%
74283	61.2	29%	53.8	32%
74181	691.8	45%	637.0	47%
c432	707.5	25%	503.9	37%

Table 2: Performance gains for DXC benchmarks when searching for the first n diagnoses of minimal cardinality.

Constraint Problems / Spreadsheets The characteristics for the next set of benchmark problems (six CSP competition instances, five CSP-encoded real-world spreadsheets with injected faults [17]) are shown in Table 3.

Scenario	#C	#V	#F	#D	$\overline{ \Delta }$	$\overline{\#Cf}$	$\overline{ Cf }$
c8	523	239	8	4	6.25	7	1.6
costasArray-13	87	88	2	>5	3.6	>565	45.6
domino-100-100	100	100	3	81	2	2	15
graceful-K3-P2	60	15	4	>117	2.94	>12	29.2
mknap-1-5	7	39	1	2	1	1	2
queens-8	28	8	15	9	10.9	15	2.8
hospital payment	38	75	4	40	4	4	3
profit calculation	28	140	5	42	4.25	11	9
course planning	457	583	2	3024	2	2	55.5
preservation model	701	803	1	22	1	1	22
revenue calculation	93	154	4	1452	3	3	15.7

Table 3: Characteristics of selected CSP settings.

The results for determining the five first minimal diagnoses are shown in Table 4². Again, performance improvements of up to 54% can be observed. The obtained improvements vary quite strongly across the different problem instances: the higher the complexity of the underlying problem, the stronger are the improvements achieved with our new method. Only in the two cases in which only one single conflict exists (see Table 3), the performance can slightly degrade as MXP performs an additional check if further conflicts among the remaining constraints exist.

Systematically Generated MBD Problems To be able to systematically analyze which factors potentially influence the obtained performance improvements, we developed an MBD problem generator in which we could vary (i) the

²The results for finding one diagnosis follow the same trend.

Scenario	QXP		MXP	
	[ms]	[ms]	[ms]	Impr.
c8	615	376		39%
costasArray-13	1,379,842	629,366		54%
domino-100-100	417	389		7%
graceful-K3-P2	1611	1123		30%
mknab-1-5	32	36		-11%
queens-8	281	245		13%
hospital payment	1,717	1,360		21%
profit calculation	86	76		12%
course planning	2,045	1,544		25%
preservation model	371	391		-5%
revenue calculation	109	87		21%

Table 4: Results for CSP benchmarks and spreadsheets when searching for 5 diagnoses.

overall number of COMPS, (ii) the number of conflicts and their average size (and as a consequence the number of diagnoses), and (iii) the position of the conflicts in the database. We considered the last aspect because the performance of QXP and MXP can largely depend on this aspect³. If, e.g., there is only one conflict and the conflict is represented by the two “left-most” elements in SD, QXP’s divide-and-conquer strategy will be able to rule out most other elements very fast.

We evaluated the following configurations regarding the position of the conflicts (see Table 5): **(a) Random**: The elements of each conflict are randomly distributed across SD; **(b) Left/Right**: All elements of the conflict appear in exactly one half of SD; **(c) LaR (Left and Right)**: Conflicts are both in the left and right half, but not spanning both halves; **(d) Neighb.**: Conflicts appear randomly across SD, but only involve “neighboring” elements.

One specific rationale of evaluating these constellations individually is that conflicts in some application domains (e.g., when debugging knowledge bases) might represent “local” inconsistencies in SD.

Since the conflicts are known in advance in this experiment, no CSP solver is needed to determine the consistency of a given set of constraints. Because zero computation times are unrealistic, we added *simulated consistency checking times* in each call to the TP. The value of the simulated time quadratically increases with the number of constraints to be checked and is capped in the experiments at 10 milliseconds. We made additional tests with different consistency checking times to evaluate to which extent the improvements obtained with MXP depend on the complexity of an individual consistency check for the underlying problem. However, these tests did not lead to any significant differences.

Table 5 shows some of the results of this simulation. In this evaluation, we also include the results of the parallelized PMXP variant. The following observations can be made.

(1) The performance of QXP strongly depends on the position of the conflicts. In the probably most realistic *Random* case, MXP helps to reduce the computation times around 20-30%. In the constellations that are “unfortunate” for QXP, the speedups achieved with MXP can be as high as 75%. When QXP is “lucky” and all conflicts are clustered

³We assume a splitting strategy in which the elements are simply split in half in the middle with no particular ordering of the elements.

#Cp	#Cf	Cf	Cf Pos.	QXP [ms]	MXP Impr.	PMXP Impr.
50	5	2	Random	351	27%	30%
50	5	2	Left	161	6%	10%
50	5	2	Right	481	69%	70%
50	5	2	LaR	293	51%	57%
50	5	2	Neighb.	261	54%	58%
100	5	2	Random	417	33%	35%
100	5	2	Left	181	14%	17%
100	5	2	Right	622	75%	76%
100	5	2	LaR	351	58%	63%
100	5	2	Neighb.	314	62%	65%
50	15	4	Random	2,300	22%	20%
50	15	4	Left	452	-8%	-4%
50	15	4	Right	1,850	72%	73%
50	15	4	LaR	3,596	22%	18%
50	15	4	Neighb.	166,335	43%	43%

Table 5: Results when varying the problem characteristics.

in the left part of SD, some improvements or light deteriorations can be observed for MXP. The latter two situations (all conflicts are clustered in one half) are actually quite improbable but help us better understand which factors influence the performance.

(2) When comparing the results of the first two blocks in the table, it can be seen that the improvements achieved with MXP are stronger when there are more components in SD and more time is needed for performing the individual consistency checks. This is in line with the results of the other experiments.

(3) Parallelization can help to obtain modest additional improvements. The strongest improvements are observed for the *LaR* configuration, which is intuitive as PMXP by design explores the left and right halves independently in parallel. Note that in the experiments with the DXC and the CSP benchmark problems, in most cases we could *not* observe runtime improvements through parallelization. This is caused by two facts. First, the consistency checking times are often on average below 1 ms, which means that the relative overhead of starting a new thread can be comparably high. Second, the used CSP solver causes some additional overheads and thread synchronization when used in multiple threads in parallel.

5 Related Work

In [10], Junker informally sketches a possible extension of QXP to be able to compute multiple “preferred explanations” in the context of Preference-Based Search (PBS). The general goal of Junker’s approach is partially similar to our work and the proposed extended version of QXP could in theory be used during the HS-tree construction as well.

Technically, Junker proposes to set a choice point whenever a constraint c_i is found to be consistent with a partial relaxation during search and thereby look for (a) branches that lead to conflicts not containing c_i and (b) branches leading to conflicts in which the removal of c_i leads to a solution.

Unfortunately, it is not fully clear from the informal sketch in [10] where the mentioned choice point should be set. If applied in line 5 of Algorithm 1, conflicts are only found in the left-most inconsistent partition. The method would then return only a small subset of all conflicts

MERGEXPLAIN would return. If the split is done for every c_i consistent with a partial relaxation during PBS, the resulting diagnosis algorithm corresponds to the binary HS-tree method [25], which according to the experiments in [11] is not generally favorable over HS-Tree algorithms, in particular when we are searching for a limited set of diagnoses.

From the algorithm design, note that QXP applies a constructive conflict computation procedure prior to partitioning, whereas MXP does the partitioning first – thereby removing multiple constraints at a time – and then uses a divide-and-conquer conflict detection approach. Finally, our method can, depending on the configuration, make a guarantee about the existence of a diagnosis given the returned conflicts without the need of computing all existing conflicts.

In general, our work is related to a variety of (complete) approaches from the MBD literature which aim to find diagnoses more efficiently than with Reiter’s original method. Existing works for example try to speed up the process by exploiting existing hierarchical, tree-like or distributed structural properties of the underlying problem [16; 26], through parallelization [17], or by solving the dual problem [27; 28; 29]. A main difference to these works is that we make no assumption about the underlying problem structure and leave the general HS-tree procedure unchanged. Instead, our aim is to avoid a full restart of the conflict search process when constructing a new node by looking for potentially existing additional conflicts in each call, and to thereby speedup the overall process.

Beside complete methods, a number of approximate diagnosis approaches have been proposed in the last years, which for example use stochastic and heuristic search [30; 31]. The relation of our work to these approaches is limited as we are focusing on application scenarios where the goal is to find a few first diagnoses more quickly but at the same time maintain the completeness property. Finally, for some domains, “direct” and SAT-based, e.g., [32], or CSP-based, e.g., [33], encodings, have shown to be very efficient to find one or a few diagnoses in recent years. For instance, [33] suggests an encoding scheme that first translates a given diagnosis problem (SD, COMPS, OBS) into a CSP. Then a specific diagnosis algorithm is applied that searches for conflict sets with increasing cardinality, i.e., $1, 2, \dots, |\text{COMPS}|$. The same method is then used to search for diagnoses in the set of all found conflict sets. In order to speed up the computations the author suggests a kind of hierarchical approach that helps the user spot the relevant components. Generally, most of the “direct” methods require the use of additional techniques like hierarchical diagnosis or iterative deepening that constrain the cardinality of computed diagnoses while computing minimal diagnoses.

The concept of conflicts plays a central role in different other reasoning contexts than Model-Based Diagnosis, e.g., explanations or dynamic backtracking. Specifically, in recent years a number of approaches were proposed in the context of the maximum satisfiability problem (MaxSAT), see [34] for a recent survey. In these domains the conflicts are referred to as *unsatisfiable cores* or *Minimally Unsatisfiable Subsets* (MUSes); *Minimal Correction Subsets* (MSCes) on the other hand correspond to the concept of diagnoses in this paper. In [35] or [36], for example, different algorithms were recently proposed to find one solution to the MaxSAT problem, which corresponds to the problem of finding one minimal/preferred diagnosis. Other

techniques such as MARCO [29] aim at the enumeration of conflicts. In general, many of these algorithms use a similar divide-and-conquer principle as we do with MXP. However, such algorithms – including the ones listed above – often modify the underlying knowledge base by adding relaxation variables to clauses of a given unsatisfiable formula and then use a SAT solver to find the relaxations. This strategy roughly corresponds to the direct diagnoses approaches discussed above. MXP, in contrast, acts completely independently of the underlying knowledge representation language. Moreover, the problem-independent decomposition approach used by MXP is a novel feature which – to the best of our knowledge – is not present in the existing conflict detection techniques from the MaxSAT field. Specifically, it allows our algorithm to find multiple conflicts more efficiently because it searches for them within independent small subsets of the original knowledge base. In addition, MXP can find conflicts in knowledge bases formulated in very expressive knowledge representation languages, such as description logics, which cannot be efficiently translated to SAT, see also [23].

6 Conclusions

We have proposed and evaluated a novel, general-purpose and non-intrusive conflict detection strategy called MERGEXPLAIN, which is capable of detecting multiple conflicts in a single call. An evaluation on various benchmark problems revealed that MERGEXPLAIN can help to significantly reduce the required computation times when applied in a Model-Based Diagnosis setting in which the goal is to find a defined number of diagnoses and in which no assumption about the underlying reasoning engine should be made.

One additional property of MERGEXPLAIN is that the union of the elements of the returned conflict sets is guaranteed to be a superset of one diagnosis of the original problem. Recent methods like the one proposed in [23] can therefore be applied to find one minimal diagnosis quickly.

Acknowledgements

This work was supported by the Carinthian Science Fund (KWF) contract KWF-3520/26767/38701, the Austrian Science Fund (FWF), and the German Research Foundation (DFG) under contract numbers I 2144 N-15 and JA 2095/4-1 (Project “Debugging of Spreadsheet Programs”).

References

- [1] Ulrich Junker. QUICKXPLAIN: Conflict Detection for Arbitrary Constraint Propagation Algorithms. In *IJCAI '01 Workshop on Modelling and Solving problems with constraints (CONS-1)*, 2001.
- [2] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.
- [3] Gerhard Friedrich, Markus Stumptner, and Franz Wotawa. Model-Based Diagnosis of Hardware Designs. *Artificial Intelligence*, 111(1-2):3–39, 1999.
- [4] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. Model-Based Debugging of Java Programs. In *Proceedings AADEBUG '00 Workshop*, 2000.

- [5] Dietmar Jannach and Thomas Schmitz. Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. *Automated Software Engineering*, 2014.
- [6] Jules White, David Benavides, Douglas C. Schmidt, Pablo Trinidad, Brian Dougherty, and Antonio Ruiz Cortés. Automated Diagnosis of Feature Model Configurations. *Journal of Systems and Software*, 83(7):1094–1107, 2010.
- [7] Kostyantyn Shchekotykhin, Gerhard Friedrich, Philipp Fleiss, and Patrick Rodler. Interactive ontology debugging: Two query strategies for efficient fault localization. *Journal of Web Semantics*, 12-13:88–103, 2012.
- [8] Franz Baader and Rafael Penaloza. Axiom Pinpointing in General Tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2008.
- [9] Johan de Kleer. A Comparison of ATMS and CSP Techniques. In *Proceedings IJCAI '89*, pages 290–296, 1989.
- [10] Ulrich Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In *Proceedings AAAI '04*, pages 167–172, 2004.
- [11] Ingo Pill, Thomas Quaritsch, and Franz Wotawa. From Conflicts to Diagnoses: An Empirical Evaluation of Minimal Hitting Set Algorithms. In *Proceedings DX '11 Workshop*, pages 203–211, 2011.
- [12] Alexander Feldman, Gregory Provan, Johan de Kleer, Stephan Robert, and Arjan van Gemund. Solving Model-Based Diagnosis Problems with Max-SAT Solvers and Vice Versa. In *Proceedings DX '10 Workshop*, pages 185–192, 2010.
- [13] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A Novel SAT-Based Approach to Model Based Diagnosis. *Journal of Artificial Intelligence Research*, 51:377–411, 2014.
- [14] Iulia Nica, Ingo Pill, Thomas Quaritsch, and Franz Wotawa. The Route to Success – A Performance Comparison of Diagnosis Algorithms. In *Proceedings IJCAI '13*, pages 1039–1045, 2013.
- [15] R Greiner, B A Smith, and R W Wilkerson. A Correction to the Algorithm in Reiter’s Theory of Diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.
- [16] Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127(1):1–29, 2001.
- [17] Dietmar Jannach, Thomas Schmitz, and Kostyantyn Shchekotykhin. Parallelized Hitting Set Computation for Model-Based Diagnosis. In *Proceedings AAAI '15*, pages 1503–1510, 2015.
- [18] Johan de Kleer. Hitting set algorithms for model-based diagnosis. In *Proceedings DX '11 Workshop*, pages 100–105, 2011.
- [19] Ulrich Junker. Preference-Based Search and Multi-Criteria Optimization. *Annals of Operations Research*, 130:75–115, 2004.
- [20] Ingo Pill and Thomas Quaritsch. Optimizations for the Boolean Approach to Computing Minimal Hitting Sets. In *Proceedings ECAI '12*, pages 648–653, 2012.
- [21] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner. Consistency-based diagnosis of configuration knowledge bases. *Artificial Intelligence*, 152(2):213–234, 2004.
- [22] Gerhard Friedrich and Kostyantyn Shchekotykhin. A General Diagnosis Method for Ontologies. In *Proceedings ISWC '05*, pages 232–246, 2005.
- [23] Kostyantyn Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss. Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In *Proceedings ECAI '14*, pages 813–818, 2014.
- [24] Thomas Eiter and Georg Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM (JACM)*, 42(1):1–49, 1995.
- [25] Li Lin and Yunfei Jiang. The computation of hitting sets: Review and new algorithms. *Information Processing Letters*, 86(4):177–184, May 2003.
- [26] F Wotawa and I Pill. On classification and modeling issues in distributed model-based diagnosis. *AI Communications*, 26(1):133–143, 2013.
- [27] Ken Satoh and Takeaki Uno. Enumerating Minimally Revised Specifications Using Dualization. In *JSAI '05 Workshop*, pages 182–189, 2005.
- [28] Roni Stern, Meir Kalech, Alexander Feldman, and Gregory Provan. Exploring the Duality in Conflict-Directed Model-Based Diagnosis. In *Proceedings AAAI '12*, pages 828–834, 2012.
- [29] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, Flexible MUS Enumeration. *Constraints*, pages 1–28, 2015.
- [30] Lin Li and Jiang Yunfei. Computing Minimal Hitting Sets with Genetic Algorithm. In *Proceedings DX '02 Workshop*, pages 1–4, 2002.
- [31] A Feldman, G Provan, and A van Gemund. Approximate Model-Based Diagnosis Using Greedy Stochastic Search. *Journal of Artificial Intelligence Research*, 38:371–413, 2010.
- [32] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. Compiling Model-Based Diagnosis to Boolean Satisfaction. In *Proceedings AAAI '12*, pages 793–799, 2012.
- [33] Yannick Pencolé. DITO: a CSP-based diagnostic engine. In *Proceedings ECAI '14*, pages 699–704, 2014.
- [34] Antonio Morgado, Federico Heras, Mark Liffiton, Jordi Planes, and Joao Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.
- [35] Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MAXSAT solving. In *Proceedings CP '13*, pages 247–262, 2013.
- [36] Alexey Ignatiev, Antonio Morgado, Vasco Manquinho, Ines Lynce, and Joao Marques-Silva. Progression in Maximum Satisfiability. In *Proceedings ECAI '14*, pages 453–458, 2014.

A Robust Alternative to Correlation Networks for Identifying Faulty Systems

Patrick Traxler¹ and Pablo Gómez² and Tanja Grill¹

¹Software Competence Center Hagenberg, Austria

e-mail:patrick.traxler@scch.at

tanja.grill@scch.at

²Institute of Applied Knowledge Processing, Johannes Kepler University, Linz, Austria

e-mail: pablo.gomez@faw.jku.at

Abstract

We study the situation in which many systems relate to each other. We show how to robustly learn relations between systems to conduct fault detection and identification (FDI), i.e. the goal is to identify the faulty systems. Towards this, we present a robust alternative to the sample correlation matrix and show how to randomly search in it for a structure appropriate for FDI. Our method applies to situations in which many systems can be faulty simultaneously and thus our method requires an appropriate degree of redundancy. We present experimental results with data arising in photovoltaics and supporting theoretical results.

1 Introduction

The increasing number of technical systems connected to the Internet raises new challenges and possibilities in diagnosis. Large amount of data needs to be processed and analyzed. Faults need to be detected and identified. Systems exist in different configurations, e.g. two systems of the same type that have different sets of sensors. Knowledge about the system design is often incomplete. Data is often unavailable due to unreliable data connections. Besides these and other difficulties, the large amount of data also opens new possibilities for diagnosis based on machine learning.

The idea of our approach is to conduct fault detection and identification (FDI) by comparing data of similar systems. We assume to have data of machines, devices, systems of a similar type and want to know if some system is faulty and if so, to identify the faulty systems. This situation may deviate from classic diagnosis problems in that we just have limited information (e.g. sensor or control information) of system internals. Moreover, we may have incomplete knowledge about the system design. This makes manual system modeling hard or even impossible. The problem is then to compare the limited information of the working systems (perhaps only input-output information) to identify faulty systems.

In this work we tackle one concrete problem of this kind. It is motivated by photovoltaics. We describe it in more detail below. The problem that arises in our and other applications is that not every two systems can be compared. We thus need to learn relations between systems.

There are different approaches to learn structure, e.g. learning Bayesian networks, Markov random fields, or sim-

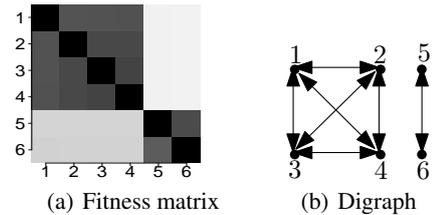


Figure 1: Learning relations between 6 systems. We draw an edge between two systems if there is a strong linear relation between them. First, we compute the fitness matrix, 1(a), our robust alternative to the sample correlation matrix. Darker colors mean a stronger linear relation. Going from Fig. 1(a) to 1(b) is a discretization step via thresholding. The digraph is the input for conducting FDI.

ilar concepts. The concept that fits our needs are correlation networks. A correlation network is some structure in the correlation matrix, e.g. a minimum spanning tree or a clustering. In our application we have n variables which represent the produced energy per photovoltaic system. Given that a single system correlates strongly with enough other systems, we use this information for FDI via applying a median.

We can also think of correlation networks as a method for knowledge discovery. It has been applied in areas such as biology [18; 10] and finance [12] to analyze gene co-expression and financial markets. In our situation, the first step is to learn linear relations between systems. For learning we need historical data. A sample result of this step is depicted in Fig. 1. In Fig. 1(a) the fitness matrix, our robust alternative to the correlation matrix, is shown. It represents the degree of linearity between any two systems. For FDI, the second step of our method, we work with the result as depicted in 1(b) and current data. In the example, we derive for every of the six systems an estimation \hat{m}_i of its current value y_i from its neighbors current values, e.g. for system 1 we get an estimate from the current values of the systems 2, 3, 4 and for system 5 from system 6. Finally, we test for a fault by checking if $|\hat{m}_i - y_i|$ is large.

The major difficulty we try to tackle with this approach is the presence of many faults. Faults influence both the learning problem and the FDI problem. Robustness is an essential property of our algorithms. Our result can be seen as a robust structure learning algorithm for the purpose of FDI. Robustness is a preferable property of many learning

and estimation algorithms. However, the underlying optimization problems unlike their non-robust variants are often NP-hard. This is for example the case for computing robust and non-robust estimators for linear regression, e.g. Least Median of Squares versus Ordinary Least Squares [16]. We avoid NP-hardness by a careful modeling of our problem. In particular, our algorithms are computationally efficient. Under some conditions, FDI can be done in (almost) linear time in the number of systems n .

To summarize our contributions, we introduce a novel alternative to the sample correlation matrix and present a first use of it to discover structure appropriate for general FDI and in particular for identifying faulty photovoltaic systems (PV). Our method works in the presence of many faults. Our algorithms are computationally efficient. Our method incorporates a couple of techniques from machine learning and statistics: (Repeated) Theil-Sen estimation for robust simple linear regression. Trimming to obtain a robust fitness measure. Randomized subset selection for improved running time. And a median mechanism to conduct FDI.

In Sec. 2 we discuss our method. In Sec. 3 we present experimental and theoretical results.

1.1 Motivating Application: Identifying Faulty Photovoltaic Systems

Faults influence the performance of photovoltaic systems (PV). PV systems produce less energy than possible if faults occur. We can distinguish between two kinds of faults. Faults caused by an exogenous event such as shading, (melting) snow, and tree leafs covering solar modules. And faults caused by endogenous events such as module defects and degradation, defects at the power inverter, and string disconnections.

We are going to detect faults by estimating the drop in produced energy. Most of the common faults result in such a drop. The particular problem is given by the sensor setup. We just assume to know the produced energy and possible but not necessarily the area (e.g. the zip code) where the PV system is located.

We apply our method to PV system data. Difficulties in the application are different system types and deployments of systems. For example, different number of strings and modules per string and differing orientation (north, west, south, east) of the modules; see Fig. 2. Moreover, the lack of information due to the lack of sensors and incomplete data due to unreliable data connections. Faults occur frequently, in particular exogenous faults during winter.

The novelty of our work in the context of photovoltaics is that it works in an extremely restrictive (only power measurement) sensor setting. To the best of our knowledge, we are the first to consider this restrictive sensor setting. We only need to know the produced energy of a PV system. There is also the implicit assumption, which is tested by the learning algorithm, that the systems are not too far from each other so that we can observe them in similar working (environmental) conditions. Distances of a couple of kilometers are possible. Systems which are very close to each other and have the same orientation such as systems in a solar power plant yield the best results. Other approaches assume the presence of a plane-of-array-irradiance sensor which are mostly deployed for solar power plants. Irradiance estimations via satellite imaging are usually not accurate enough.

1.2 Related Work

Correlation networks have applications in biology and finance. See e.g. [12; 18; 10] and the references therein. In biology [18; 10], they are applied to study gene interactions. The correlation matrix is the basis for clustering genes and the identification of biologically significant clusters. In [18; 10], a scale-free network is derived via the concept of topological overlap. Scale-free networks tend to have few nodes (genes) with many neighbors, so called hubs.

Correlation networks are primarily used for knowledge discovery. In particular, concepts such as clusters, hubs, and spanning trees are interpreted in the context of biology and finance. In our work, we introduce a robust alternative to correlation networks.

Other structural approaches, i.e. approaches based on graphical models, are based on Bayesian networks, Markov random fields and similar concepts. Gaussian Markov random fields are loosely related to correlation networks. Their structure is described by the precision matrix, the inverse covariance matrix (ch. 17.3, [9].)

Another structural approach is FDI in sensor networks [7; 4; 19; 20]. The current approach [7; 4; 19] mainly deals with wireless sensor networks. The algorithms usually use the median for FDI such as we do. The difference is that FDI in wireless sensor networks uses a geometric model similar to interpolation methods. It requires the geographic location of the sensors. It is assumed that two sensors close to each other have a similar value. This cannot be assumed in general. To overcome these problems of manual modeling, we apply machine learning techniques.

Models for PV systems are compared in [14]. All these models require the plane-of-array irradiance. Fault detection of PV systems is the topic of e.g. [3; 8; 5; 2; 17]. Firth et al. [8] consider faults if the PV system generates no energy. Another type of fault occurs if the panels are covered by snow, tree leaves, or something else. In this case, we can observe a drop in energy. It is considered e.g. in [5]. The fraction of panel area covered is a crucial parameter. All these approaches [3; 8; 5; 2; 17] require at least the knowledge of the plane-of-array irradiance, i.e. it requires an irradiance sensor installed. We do make this assumption.

The median is common in fault detection and identification. One reason for this circumstance is its optimal breakdown point [16]. We also make use of (repeated) Theil-Sen algorithms [6; 15] for learning. An ingredient of our fault identification algorithm is the algorithm for median selection [1] and an algorithm for generating uniform subsets efficiently (see e.g. the Fisher-Yates or Knuth shuffle in [13].) In our algorithm analysis we derive bounds for a partial Cauchy-Vandermonde identity (pg. 20 in [11]).

2 Method

2.1 Data Model for Incomplete Data

We have data from n systems and one data stream per system. A data stream for system $i \in \{1, \dots, n\}$ is given by a set $N_i \subseteq \{1, \dots, N\}$ of available data and values $x_{i,t} \in \mathbb{R}$ with $t \in N_i$. We can think of the parameter t as discrete time. With N_i , we explicitly model data availability. Incomplete data is a common problem in our situation. Causes in practice are unreliable data connections or unreliable sensors. We call $D := \{(x_{i,t})_{t \in N_i} : i \in \{1, \dots, n\}\}$ a *data*

set. Sets of historical and current data are the inputs to our algorithms.

2.2 Fitness Matrix – Definition and Robustness

The fitness matrix is intended as a robust replacement for the sample correlation matrix. The sample correlation coefficient such as the sample covariance is well known to be sensitive to faults (outliers) [16]. As an example, we generated the data for Fig. 1 with faults. The non-robust sample correlation matrix would have yielded a digraph without edges instead of the digraph in Fig. 1(b).

A fault can be an arbitrary corruption of a single data item $x_{i,t}$. That is, $x_{i,t} = \tilde{x}_{i,t} + \Delta$, $\Delta \neq 0$, where Δ is the fault. We think of $\tilde{x}_{i,t}$ as the actual or true but unobserved value.

We do not make any assumptions on faults themselves but only on their number. This is at core of the definition of the breakdown point. This statistical concept is defined for a particular estimation or learning problem. In our case for simple linear regression.

Linear regression is closely related to the correlation coefficient. For simple linear regression – a regression model with one independent and one dependent variable – the correlation coefficient can be seen as a fitness measure of the line which fits the data best w.r.t. vertical squared distances. See e.g. [16]. However, the corresponding estimator, namely ℓ_2 -regression a.k.a. ordinary least squares, is known to be sensitive to outliers [16]. On the other hand, there are estimators for simple linear regression which are robust to a large number of faults, i.e. they have a large breakdown point.

The idea underlying the fitness matrix is thus to replace the correlation coefficient (and ℓ_2 -regression) by a robust notion of fitness based on robust linear regression. In the remainder of this section we recall the definition of the breakdown point following [16], pg. 9, and we are going to formalize the notion of fitness matrices.

We define the breakdown only for simple linear regression. We fix two systems $i, j \in \{1, \dots, n\}$ and define $Z := Z_{i,j} := \{(x_{i,t}, x_{j,t}) : t \in N_i \cap N_j\}$. Let T be a regression estimator, i.e. $T(Z_{i,j}) = \hat{\theta} \in \mathbb{R}^2$ is the intercept and slope for the data set $Z_{i,j}$. For Z , we define Z' as Z with m data points arbitrarily corrupted. Define

$$\text{bias}(m; T, Z) := \sup_{Z'} \|T(Z) - T(Z')\|.$$

If $\text{bias}(m; T, Z)$ is infinite, then m faults (outliers) have an arbitrarily large effect on the estimate $T(Z')$. The (*finite sample*) *breakdown point* of T and Z is defined as

$$\varepsilon^*(T, Z) := \min \left\{ \frac{m}{|Z|} : \text{bias}(m; T, Z) = \infty \right\}.$$

To explain this notion, we consider four typical examples. The breakdown point $\varepsilon^*(T_{\ell_2}, Z)$ is $1/n$ for ℓ_2 -regression. This holds for any Z . The situation is different for ℓ_1 -regression in that $\varepsilon^*(T_{\ell_1}, Z) = 1/n$ for some Z .

In this work we are going to use the Theil-Sen estimator¹ T^{TS} a.k.a. median slope selection. The reason is its breakdown point of at least $1 - \frac{1}{\sqrt{2}} \geq 0.292$ (see e.g. [6]) and the

¹There is a subtle issue here we have to deal with. Regression problems are optimization problems. The solution to the concrete optimization problem does not need to be unique. In our situation, intercept and slope are unique for ℓ_2 -regression but not for ℓ_1 -regression. The estimator T_{ℓ_1} is however unique for a (deterministic) algorithm solving the optimization problem. We thus think of T_{ℓ_1} as the output of a particular (deterministic) algorithm.

wide availability of efficient implementations of near-linear time algorithms. There is also a variant of T^{TS} , called the repeated Theil-Sen estimator, which has a breakdown point of 0.5, but less efficient implementations. The concrete definition of T^{TS} can be found e.g. in [6]. It is however not important for our application, only its robustness property and the existence of efficient implementations are.

To define the breakdown point of a fitness matrix, let f be a real-valued function defined on any finite data set. We define the *fitness matrix* as

$$F_{ji} := f(Z_{i,j})$$

and its breakdown point as

$$\varepsilon^*(F) := \min_{i,j} \varepsilon^*(f, Z_{i,j}).$$

Next, we provide the fitness matrix we are going to use. It has the property that F_{ji} is close to zero if x_i and x_j are strongly linearly related and it has a high breakdown point.

Let $y_t := x_{i,t}$ and $\hat{y}_t := x_{j,t} \cdot \hat{\theta}_2 + \hat{\theta}_1$, $t \in N_i \cap N_j$, for the Theil-Sen estimate $\hat{\theta}$ of $Z_{i,j}$. Let $r_t := \hat{y}_t - y_t$ be the residuals. And let $i_1, \dots, i_k \in N_i \cap N_j$ with $k := |N_i \cap N_j|$ be such that $|r_{i_1}| \leq \dots \leq |r_{i_k}|$. We define

$$f^{\text{TS}}(Z_{i,j}) := \frac{1}{\sum_{t=1}^{\lfloor k/\sqrt{2} \rfloor} |y_{i_t}|} \cdot \sum_{t=1}^{\lfloor k/\sqrt{2} \rfloor} |r_{i_t}|. \quad (1)$$

We define F^{TS} w.r.t. f^{TS} , i.e.

$$F_{ji}^{\text{TS}} := f^{\text{TS}}(Z_{i,j}).$$

Note that the sum goes from 1 up to $\lfloor k/\sqrt{2} \rfloor$. This trimming together with the high breakdown point of Theil-Sen directly implies the following result.

Theorem 1. *It holds that $\varepsilon^*(F^{\text{TS}}) \geq 1 - \frac{1}{\sqrt{2}}$.*

Finally, we compare the sample correlation matrix and the fitness matrix. Let C denote the sample correlation matrix and define $C'_{ji} = 1 - |C_{ji}|$. Both matrices have the property that if some entry is close to 0 then x_i and x_j have a strong linear relation. It is guaranteed that C'_{ji} is at most 1. A value close to 1 means a weak linear relation. For F_{ji}^{TS} , it is not guaranteed that $F_{ji}^{\text{TS}} \leq 1$, but experimental results suggest that it is usually the case. We also note that both matrices obey a weak form of the triangle inequality since if x_i and x_j correlate strongly and x_j and x_k correlate strongly, then also x_i and x_k correlate.

There are two important benefits of fitness matrices over correlation matrices. They are robust and are also defined for incomplete data. On the negative side, the fitness matrix is not positive semi-definite, in particular not symmetric.

2.3 Structure in Fitness Matrices – Algorithm LEARN and IDENTIFY

We want to identify faulty systems. In a first step, we learn a structure appropriate for FDI; see algorithm LEARN. We obtain it via thresholding the fitness matrix. Most of the correlation networks, i.e. structures arising from the sample correlation matrix, are obtained in this way [12; 18; 10]. We denote the threshold by $\theta \geq 0$ and the *threshold fitness matrix* as

$$F_{ji;\theta} := \begin{cases} F_{ji} & \text{if } F_{ji} \leq \theta \\ 0 & \text{if } F_{ji} > \theta \end{cases}.$$

Algorithm 1 Algorithm LEARN with input D (data set) and parameter θ (fitness threshold). Output is a digraph G with edge labels (intercept, slope) representing the threshold fitness matrix.

Let $G = (V, E)$ be a digraph with $V = \{1, \dots, n\}$ and $E = \{\}$.

for all $i \in V$ and $j \in V \setminus \{i\}$ **do**

Learn (Theil-Sen) the intercept $a_{j,i}$ and slope $b_{j,i}$ between x_i (dependent variable) and x_j (independent variable).

end for

for all $i \in V$ and $j \in V \setminus \{i\}$ **do**

Compute the trimmed fitness $f = f^{\text{TS}}$ (Eq. 1) of $Z_{i,j}$.

end for

if $f \leq \theta$ **then**

Add to G the directed edge from j to i with edge labels $(a_{j,i}, b_{j,i})$.

end if

The input to algorithm LEARN is a data set D as described in Sec. 2.1. It outputs a digraph $G = (V, E)$, i.e. the (possible sparse) threshold fitness matrix F_{θ}^{TS} . Additionally, intercept and slope of the simple linear regressions are added as edge labels.

Algorithm 2 Algorithm IDENTIFY with input G (digraph with edge labels), current data y_i for the i -th system, and parameters k and s (deviation). It outputs the set of all faulty systems H .

Set $H = \{\}$.

for all $i \in V = \{1, \dots, n\}$ **do**

Let $N_i^- := \{j \in V : (j, i) \in E\}$.

if $|N_i^-| = 0$ **then**

Continue with the next (system) i .

end if

if $|N_i^-| \geq k$ **then**

Select uniformly at random a k -element subset S from N_i^- .

else

Set $S := N_i^-$.

end if

Compute $M_i := \{\hat{y}_j = b_{j,i} \cdot y_j + a_{j,i} : j \in S\}$.

Compute the median \hat{m}_i of M_i .

Add i to H if $|\hat{m}_i - y_i| > s$

end for

Output H .

In the second step, we identify the faulty systems; see algorithm IDENTIFY. Its input is the result of algorithm LEARN. Algorithm IDENTIFY constructs a random digraph of in-degree at most k for FDI. It works as follows. Independently for every system, we choose uniformly at random at most k of its neighbors in the digraph G and compute the median \hat{m}_i of estimated values derived from the selected neighbors values. We compare the median \hat{m}_i to the current system value and decide whether it has a fault or not via the deviation parameter s .

We discuss the threshold parameter θ and the deviation parameter s in Sec. 3.1. They essentially depend on the variance in the data set D . Parameter k in algorithm IDENTIFY has the purpose of improving running time efficiency. In

particular, we have the following result.

Theorem 2. Let D be a data set with n systems and let $m := \max_i |N_i^-|$. The running time of LEARN is $O(n^2 \cdot m \cdot \log(m))$. The running time of IDENTIFY is $O(k \cdot n)$.

Proof. LEARN. There are $O(n^2)$ pairs of systems. The Theil-Sen estimator can be computed in time $O(m \log(m))$ [6]. The computation of f^{TS} , Eq. 1, is done via sorting and thus takes time $O(m \log(m))$.

IDENTIFY. Assume $|N_i^-| \geq k$. We uniformly at random choose a k -element subset out of N_i^- and compute the median. For random selection we can use for example the Fisher-Yates (or Knuth) shuffle [13] which runs in time $O(k)$ and for median selection the algorithm in [1] which also runs in time $O(k)$. The second case, $1 \leq |N_i^-| \leq k-1$, is analogous. This shows that the overall running time of IDENTIFY is $O(kn)$. \square

In Sec. 3.2, we provide some sufficient conditions that IDENTIFY works correctly even if $k = O(\log(n))$. This is a strong running time improvement from $O(n^2)$ to $O(n \cdot \log(n))$.

3 Results

3.1 Experimental

In this section we are going to discuss how to apply our method, Sec. 2, to photovoltaic data. In particular, it remains to discuss how the use-case fits to the model. More precisely, why there is strong correlation between PV systems. Finally, we present experimental results to verify the estimation and fault identification quality of our algorithms.

Use-Case Photovoltaics

A simple system model for PV systems is as follows:

$$P_i = c_i \cdot I_i.$$

Here, P_i is the power, I_i the plane-of-array (POA) irradiance of the i -th system, and c_i a constant of the system which can be interpreted as the efficiency of converting solar energy into electrical energy. More complex physical models include system variables such as the module temperature [14; 17]. Our considerations translate to the more complex models as long as they are time-independent. We also note that these models are more accurate, but only slightly, since the POA-irradiance has the most critical influence on the produced energy.

We get from the above considerations that $P_i = c'_{ij} \cdot P_j$ given that $I_i = I_j$. In our situation we cannot test the condition $I_i = I_j$ since we do not know the POA-irradiance, but $I_i \approx I_j$ holds if the system operate under similar weather conditions and have a similar orientation. The former holds if the systems are close to each other. To reduce the effect of different orientations, see Fig. 2, we consider the following model: $P_i^{\Delta} = u_{ij} \cdot P_j^{\Delta} + v_{ij}$. The variable P_i^{Δ} is the power within a time interval Δ , usually one hour. The variables u_{ij} and v_{ij} are the unknowns.

In more general words, let Y_i be the output of the i -th system and let X_i describe the system input and system internals. Our model assumption is that for a reasonable number of system pairs (i, j) , the system outputs Y_i are Y_j are linearly related given that $X_i \approx X_j$. By the above considerations, it is plausible that PV systems fulfill these requirements.

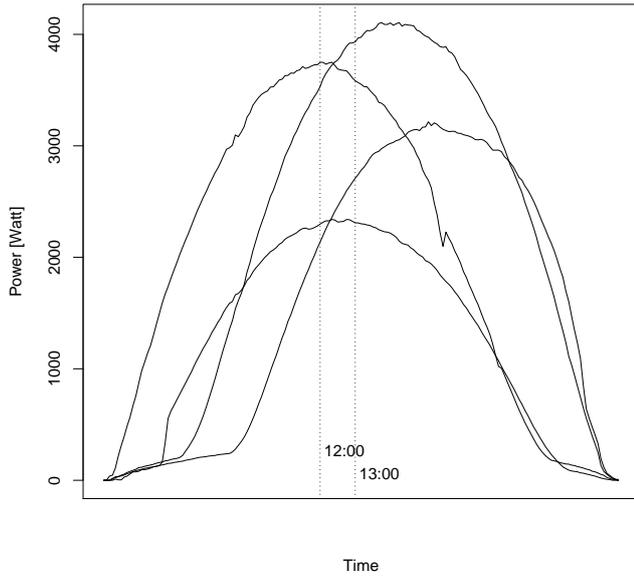


Figure 2: Four power curves of a sunny day in August, data set D_K . Two PV systems have their maximum power peak before and the other two after 13:00. They have different orientations, i.e. they produce more energy in the morning or evening.

We next describe our experimental setup to verify it by real data.

Experimental Setup

To demonstrate our method, we use two data sets D_A and D_K . D_A arises from 13 systems from a solar park located in Arizona². The PV systems there are geographically close. We use data for one year. D_K arises from 40 systems spread across a typical municipality located in Austria, i.e. the systems can be up to some kilometers apart. Their orientation can differ significantly. Some systems may be orientated to the west, others to the east. We have data for almost a year.

A system is faulty if it produces considerable less energy than estimated; see Fig. 3. This definition is motivated by the fact that most faults imply a drop in energy. The difficulty in setting up an experiment is that we do not know if a PV system is faulty in advance, i.e. we do not have labeled data. We thus design our experiment as follows: We verify the accuracy of the energy estimation, namely the relative deviation $|\hat{m}_i - y_i|/|\hat{m}_i|$ for every system i and over the period of a week, \hat{m}_i and y_i as in algorithm IDENTIFY.

This relative deviation is noted in column *Hour* of Table 1 for the time period 12:00 to 13:00. In column *Day* of Table 1 we note the same but for a whole day, i.e. \hat{m}_i is the estimated energy (power) for the whole day calculated from the hourly estimates and y_i the actual energy for the whole day. For the whole day we consider the time period from 9:00 to 16:00.

The number $|\hat{m}_i - y_i|/|\hat{m}_i|$ can be read as some relative deviation, i.e. the estimation is $100 \cdot x\%$ away from the truth value where x is some entry in the column *Hour* and *Day*.

²<http://uapv.physics.arizona.edu/>

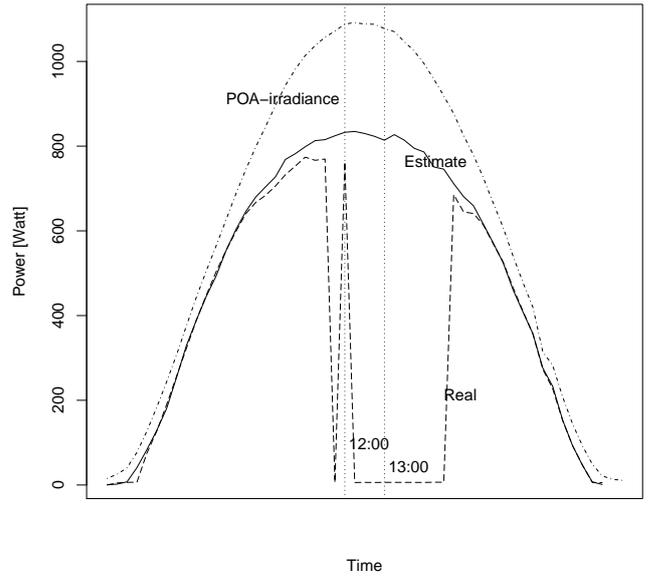


Figure 3: A faulty system. The real power curve of observed values shows a fault from roughly 11:00 to 14:30. The estimated values are considerable higher during this period. The PV system has a plane-of-array irradiance sensor installed. A cross check with its power curve reveals that the fault was detected correctly.

The entry x is an average over all systems and 7 days. The first day is noted in column *Start*.

Algorithm LEARN is executed once for every week and with $\theta = 0.8$ and roughly three months of historical data, e.g. for the months January, February, and March to get estimates for the days April, 1. to April, 7. Algorithm IDENTIFY is executed with $s = 0.25 \cdot |\hat{m}_i|$ and $k = 11$ for both data sets. The choice of parameters θ and s depend on the variance of the input data and were chosen manually, so to get a reasonable number of good estimates. Similar for k . The difficulty in choosing the parameters is that increasing θ will usually reduce the number of neighbors. For a reasonable number of good estimates we need both: A strong linear relation of a system to its neighbors and enough neighbors. The parameters were chosen accordingly. For parameter k , we derive a theoretical result in Sec. 3.2 which says that $k = O(\log(n))$ is a good choice for n the number of systems.

Experimental Results

The false positive rate (FPR), the false negative rate (FNR), and the estimation accuracy are the most interesting numbers for us. As remarked above, we do not have labeled data. The faults as recorded in Table 1 are faults as detected by our algorithm.

We make a worst case assumption, namely that all detected faults are false positives. This yields a FPR of at most 0% to 5% per 7 day period (rows in the table.) To get an understanding of FNR, we simulated faults by subtracting 33% percent of energy. The FNR in this case is at most 10% per 7 day period. In the rows *Sum* and *Sum-33%* in Table 1 we summed up the faults to get the FPR and FNR for the whole

(a) Results for D_A .

Start	Hour	Faults	Day	Faults
April, 1.	0.058	2/77	0.037	1/77
May, 1.	0.040	2/77	0.014	0/77
June, 1.	0.019	0/91	0.021	0/91
July, 1.	0.068	7/88	0.071	7/88
Aug., 1.	0.362	12/91	0.250	7/91
Sept., 1.	0.096	4/65	0.034	1/78
Oct., 1.	0.019	0/84	0.016	0/84
Nov., 1.	0.039	0/72	0.025	0/72
Dec., 1.	0.135	10/84	0.130	7/84
Sum		37/729		23/742
Sum-33%		673/729		682/742

(b) Results for D_K .

Start	Hour	Faults	Day	Faults
June, 1.	0.056	7/269	0.068	11/273
June, 15.	0.055	7/238	0.097	17/258
July, 1.	0.077	7/267	0.068	9/280
July, 15.	0.025	0/267	0.044	6/280
Aug., 1.	0.037	2/279	0.030	3/280
Aug., 15.	0.031	1/280	0.032	0/280
Sept., 1.	0.040	0/280	0.033	0/280
Sept., 15.	0.092	20/280	0.056	0/280
Sum		42/2160		46/2211
Sum-33%		1960/2154		2033/2207

Table 1: The values in column *Hour* and *Day* contain the relative deviation $|\hat{m}_i - y_i|/|\hat{m}_i|$, \hat{m}_i and y_i as in algorithm IDENTIFY. They are averages over all systems and the period of a week. Column *Start* contains the start date of the 7 day period. The two columns labeled *Faults* contain the number of (possible false detected) faults relative to the total number of analyzed hours and days, respectively. The rows *Sum* contain the summed up number of faults, once for the actual data sets and then with a simulated fault of -33% less energy.

data sets.

The interpretation of these results is as follows. Setting the parameter s to $0.25 \cdot |\hat{m}_i|$ means that we define a fault as a 25% relative deviation of the observed produced energy from its true value. Setting s to this value, yields the above mentioned FPR. Simulating a 33% drop in energy, which corresponds naturally to the faults we want to detect, yields the above FNR.

For the data set D_A we have knowledge about the POA-irradiance. We can thus cross-check with the irradiance to check if faulty systems were identified correctly; see Fig. 3. This manual inspection suggests that the FPR is much smaller than 5%, close to 1%. Furthermore, increasing the drop implies a decreasing FNR, i.e. stronger energy drops are easier to identify.

Depending on the application, these rates may be considered appropriate or not. In some applications, we may want to detect faults which yield a drop in energy of less -25%. This worsens the FPR and FNR. On the other side, if we want to improve the FPR and FNR, we may have to specify a fault as a drop in energy of -50%. In other words, our parameter setting is one out of many reasonable parameter settings.

3.2 Theoretical

We argued in Sec. 3.1 that algorithm LEARN yields good estimates for the systems current value. For an estimate to be good, the neighboring system j in G of system i needs to work correctly. Moreover, the regression estimates, the intercept and slope, need to be accurate enough. In this section, we provide a supporting theoretical result which says that, if enough estimates are good, algorithm IDENTIFY correctly identifies all faulty systems.

The input to IDENTIFY is a digraph $G = (V, E)$ with edge labels. Let y_i be the current value of system i . Let $y_i = \tilde{y}_i + \Delta_i$. We think of \tilde{y}_i as the true value. We say that system i is *correct* if $\Delta_i = 0$ and *faulty* otherwise.

The input to IDENTIFY has to satisfy two conditions, Eq. 2 and 3, to work correctly. These conditions state that there are more good than bad estimates. We formulate them below.

Theorem 3. *Let $0 < p < 1$ and $s > 0$. Let $H := \{i \in \{1, \dots, n\} : |\Delta_i| > 2s\}$. Assume that the input digraph G satisfies Eq. 2 and 3. Then, algorithm IDENTIFY outputs H with probability at least $1 - p$.*

Let \hat{y}_j be the estimates as computed in IDENTIFY. Fix a system i and let $j \in N_i^-$. We say that \hat{y}_j is s -good for system i if $|\tilde{y}_i - \hat{y}_j| \leq s$. Let $A_i := \{j \in N_i^- : |\tilde{y}_i - \hat{y}_j| \leq s\}$ be the s -good estimates for system i . Condition 2 is as follows: For every system i with $1 \leq |N_i^-| \leq k - 1$ it holds that

$$|A_i| > \frac{|N_i^-|}{2}, \quad (2)$$

i.e. there are more good than bad estimates. For the case that $|N_i^-| \geq k$ we assume

$$|A_i| > \left(1 - \frac{1}{c_{n,p,k}}\right) \cdot |N_i^-|, \quad (3)$$

with $c_{n,p,k} := \left(\frac{n}{p} \cdot 18^k\right)^{2/(k-1)}$. Setting $k = \Omega(\log(\frac{n}{p}))$ makes $c_{n,p,k}$ larger than some constant *independent* of n and p . This is the most reasonable setting as it implies that a constant fraction of estimates can be bad and IDENTIFY still identifies the faulty systems correctly. We remark that the asymptotic analysis which yields $c_{n,p,k}$ is not optimal. In particular, it seems that the factor 18^k is not optimal and may be improved to a factor as small as $2^{k/2}$. For practical applications, the following heuristic seems reasonable: For n systems and a failure probability p of IDENTIFY, set k to $10 \cdot \log(\frac{n}{p})$.

3.3 Proof of Theorem 3

We apply the following lemma with $A = G_i$ and $M = N_i^-$. It directly gives us the probability that IDENTIFY correctly identifies the faulty systems since the median works correctly if $|S \cap A_i| > |S \cap (N_i^- \setminus A_i)|$, where S is the (random) set chosen in IDENTIFY.

Lemma 1. *Let M be a finite set and $A \subseteq M$. Let $k \geq 2$ be an integer. Let $S \subseteq M$ be a k -element subset selected uniformly at random. Then*

$$\Pr_S(|S \cap A| > |S \cap (M \setminus A)|) \geq 1 - 18^k \left(\frac{|M \setminus A|}{|M|}\right)^{\lfloor k/2 \rfloor}.$$

Proof. Let $M := \{1, \dots, m\}$, $F := M \setminus A$, and $r := |F|$. First, we are going to bound the number of k -element subsets $S \subseteq M$ for which $|S \cap G| \leq k'$ with $k' = \lfloor k/2 \rfloor$. The exact number of these sets is

$$\sum_{i=0}^{k'} \binom{m-r}{i} \binom{r}{k-i} \quad (4)$$

since there are $\binom{|A|}{i}$ ways to choose an i -element subset from A and $\binom{|F|}{k-i}$ ways to choose from F for the remaining $k-i$ elements.

Note that $|S \cap A| > |S \cap F|$ iff $|S \cap A| > \lfloor |S|/2 \rfloor = k'$. Moreover, we can assume that $r = |F| \geq 1$ since the claim holds for $r = 0$. To provide a lower bound for the probability of this event we show an upper bound on the complementary event, i.e. $|S \cap A| \leq k'$. First, we derive an upper bound for Eq. 4 using

$$\binom{m}{k} \leq \binom{m}{k} \leq \left(\frac{me}{k}\right)^k \quad (5)$$

for $e = 2.714\dots$ and $1 \leq k \leq m$. (See e.g. pg. 12 in [11].) Since this inequality holds just for $k \geq 1$ we rewrite Eq. 4 as

$$\binom{r}{k} + \sum_{i=1}^{k'} \binom{m-r}{i} \binom{r}{k-i}. \quad (6)$$

It holds that $\binom{r}{k} \leq \left(\frac{re}{k}\right)^k$ and for the second term in Eq. 6

$$\begin{aligned} \sum_{i=1}^{k'} \binom{m-r}{i} \binom{r}{k-i} &\leq \sum_{i=1}^{k'} \left(\frac{(m-r)e}{i}\right)^i \left(\frac{re}{k-i}\right)^{k-i} \\ &= (re)^k \sum_{i=1}^{k'} \left(\frac{m-r}{r}\right)^i \left(\frac{k-i}{i}\right)^i \left(\frac{1}{k-i}\right)^k \end{aligned}$$

Next, we prove the upper bound on the probability p that $|S \cap A| \leq k'$. We select uniformly at random a k -element subset of M . Its probability is $\binom{m}{k}^{-1}$. We multiply Eq. 6 with $\binom{m}{k}^{-1}$ and get two parts $p_1 + p_2 \geq p$. For the first part $p_1 \leq \left(\frac{re}{m}\right)^k$ since $\binom{m}{k}^{-1} \leq (k/m)^k$. For the second part p_2 , we use $\frac{m-r}{r} \leq \frac{m}{r}$, $\left(\frac{k-i}{i}\right)^i \leq 2^k$, and $(k/(k-i))^k \leq 2^k$. The latter since $i \leq k'$. We get an upper for the second part:

$$\begin{aligned} p_2 &\leq \left(\frac{re}{m}\right)^k \sum_{i=1}^{k'} \left(\frac{m-r}{r}\right)^i \left(\frac{k-i}{i}\right)^i \left(\frac{k}{k-i}\right)^k \leq \\ &\quad \left(\frac{12r}{m}\right)^k \sum_{i=1}^{k'} \left(\frac{m}{r}\right)^i. \end{aligned}$$

An upper bound for the geometric sum is $k'(m/r)^{k'}$. In total

$$p \leq p_1 + p_2 \leq \left(\frac{re}{m}\right)^k + k' \left(\frac{12r}{m}\right)^k \left(\frac{m}{r}\right)^{k'}$$

Substituting $\frac{k-1}{2}$ for k' and further simplification yields

$$p \leq \frac{(k+2)(12)^k}{2} \left(\frac{r}{m}\right)^{(k-1)/2} \leq 18^k \left(\frac{r}{m}\right)^{(k-1)/2}$$

The latter since $((k+2)/2)^{1/k} \leq 1.5$ for $k \geq 3$. We have thus a lower bound for the probability $1-p$ and the claim follows. \square

Proof of Theorem 3. We show that the success probability of IDENTIFY is at least $1-p$. Let $p' := \frac{p}{n}$. We show that for every $i \in V$, $G = (V, E)$, the success probability of a single iteration in the loop of IDENTIFY is at least $1-p'$. This implies the above claim since $(1-p')^n \geq 1-p'n$ by e.g. the Binomial Theorem.

Fix some $i \in V$, i.e. we consider one iteration in the loop of IDENTIFY. We apply Lemma 1. Let us assume that $|S \cap A_i| > |S \cap (N_i^- \setminus A_i)|$, S the random k -element subset as in IDENTIFY and A_i the good estimates as defined above. Since $|S \cap A_i| > |S \cap (N_i^- \setminus A_i)|$, it follows that $|\hat{m}_i - \tilde{y}_i| \leq s$ for the median \hat{m}_i as computed in IDENTIFY and $y_i = \tilde{y}_i + \Delta_i$.

Assume $\Delta_i = 0$, i.e. system i works correctly. Then, $|\hat{m}_i - y_i| = |\hat{m}_i - \tilde{y}_i| \leq s$. Thus, i is not output.

Assume $\Delta_i \neq 0$, i.e. system i is faulty. Here, $|\hat{m}_i - y_i| = |\hat{m}_i - \tilde{y}_i - \Delta_i|$. It follows from $|\hat{m}_i - \tilde{y}_i| \leq s$ and $|\Delta_i| > 2s$ that $|\hat{m}_i - y_i| > s$. Thus, i is output.

Finally, we want that the probability of failure for a single step is at most $\frac{p}{n}$. By Lemma 1, $18^k \alpha^{\lfloor k/2 \rfloor} \leq 18^k \alpha^{(k-1)/2} \leq \frac{p}{n}$ with $\alpha := \frac{|N_i^- \setminus A_i|}{|N_i^-|}$. With $c = c_{n,p,k} := \left(\frac{n}{p} \cdot 18^k\right)^{2/(k-1)}$, $c \cdot |N_i^- \setminus A_i| \leq |N_i^-|$ and thus $(1-1/c) \cdot |N_i^-| \leq |A_i|$. \square

4 Conclusions and Open Problems

We presented a method for learning structure to identify faulty systems. The basic method of correlation networks has found many applications in biology and finance. In our application, the presence of many faults required the design and analysis of robust algorithms. We provided an experimental analysis of our algorithms to verify their estimation and fault identification quality. We also provided a supporting theoretical result which allowed us to considerably improve the running time of algorithm IDENTIFY.

Improving the running time of LEARN remains as an open problem. It is not directly clear that it is necessary to compare every two systems. The reason is that if systems (i, j) and (j, k) correlate strongly, then also (i, k) correlate, but not necessarily strongly. Thus, it may not be necessary to solve a simple linear regression problem for every system pair.

In other applications it may be useful to solve a general linear regression problem instead of a simple linear regression, e.g. if our model depends on more than one variable per system. The corresponding correlation networks are based on the partial correlation coefficient [12]. Since robust estimators for general linear regression are based on regression problems which are NP-hard, it remains as an open problem to find a robust alternative to partial correlation networks that can be computed efficiently.

Finally, to put our method and results into a broader context, we approached the problem of FDI via learning graphical models. It seems to be a challenge to learn classical component-models of technical systems to conduct diagnosis. In this work we were able to close the gap between (structure) learning on the one side and FDI on the other side for a concrete problem setting.

References

- [1] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Linear time

- bounds for median computations. In *Proc. of the 4th Annual ACM Symposium on Theory of Computing*, pages 119–124, 1972.
- [2] H. Braun, S. T. Buddha, V. Krishnan, A. Spanias, C. Tepedelenlioglu, T. Yeider, and T. Takehara. Signal processing for fault detection in photovoltaic arrays. In *37th IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1681–1684, 2012.
- [3] K. H. Chao, S. H. Ho, and M. H. Wang. Modeling and fault diagnosis of a photovoltaic system. *Electric Power Systems Research*, 78(1):97–105, 2008.
- [4] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Proc. of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, pages 65–72, 2006.
- [5] A. Chouder and S. Silvestre. Fault detection and automatic supervision methodology for PV systems. *Energy Conversion and Management*, 51:1929–1937, 2010.
- [6] R. Cole, J.S. Salowe, W.L. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM Journal on Computing*, 18(4):792–810, 1989.
- [7] M. Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 902–913, 2005.
- [8] S.K. Firth, K.J. Lomas, and S.J. Rees. A simple model of PV system performance and its use in fault detection. *Solar Energy*, 84:624–635, 2010.
- [9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2008.
- [10] Steve Horvath. *Weighted network analysis: applications in genomics and systems biology*. Springer Science & Business Media, 2011.
- [11] S. Jukna. *Extremal combinatorics: with applications in computer science*. Springer, 2nd edition, 2011.
- [12] Dror Y. Kenett, Michele Tumminello, Asaf Madi, Gittit Gur-Gershoren, Rosario N. Mantegna, and Eshel Ben-Jacob. Dominating clasp of the financial sector revealed by partial correlation analysis of the stock market. *PLoS ONE*, 5(12):e15032, 12 2010.
- [13] Donald E. Knuth. *The art of computer programming: seminumerical algorithms*, volume 2. Addison-Wesley Longman Publishing Co., Inc., 3rd edition, 1997.
- [14] B. Marion. Comparison of predictive models for PV module performance. In *33rd IEEE Photovoltaic Specialist Conference*, pages 1–6, 2008.
- [15] J. Matousek, D. M. Mount, and N. S. Netanyahu. Efficient randomized algorithms for the repeated median line estimator. *Algorithmica*, 20(2):136–150, 1998.
- [16] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John Wiley & Sons, 2005.
- [17] P. Traxler. Fault detection of large amounts of photovoltaic systems. In *Online Proc. of the ECML PKDD 2013 Workshop on Data Analytics for Renewable Energy Integration (DARE'13)*, 2013.
- [18] Bin Zhang and Steve Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(17), 2005.
- [19] Chongming Zhang, Jiuchun Ren, Chuanshan Gao, Zhonglin Yan, and Li Li. Sensor fault detection in wireless sensor networks. In *Proc. of the IET International Communication Conference on Wireless Mobile and Computing*, pages 66–69, 2009.
- [20] Yang Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: a survey. *Communications Surveys and Tutorials, IEEE*, 12(2):159–170, 2010.

Applied multi-layer clustering to the diagnosis of complex agro-systems

Elisa Roux¹, Louise Travé-Massuyès¹ and Marie-Véronique Le Lann^{1,2}

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

emails: lisa.roux@laas.fr, louise@laas.fr, mvlelann@laas.fr

²Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

Abstract

In many fields, such as medical, environmental, a lot of data are produced every day. In many cases, the task of machine learning is to analyze these data composed of very heterogeneous types of features. We developed in previous work a classification method based on fuzzy logic, capable of processing three types of features (data): qualitative, quantitative, and more recently intervals. We propose to add a new one: the object type which is a meaningful combination of other features yielding the possibility of developing hierarchical classifications. This is illustrated by a real-life case study taken from the agriculture area¹.

1 Introduction

Nowadays, large scale datasets are produced in various different fields such as social networks, medical, process operation, agricultural/environmental,... Many studies relate to data mining with the intention of analyzing and if possible extracting knowledge from these data. The data classification has to provide a relevant and well-fitted representation of reality. In this context, the issue of representing of data is crucial since the formalisms must be generic yet well suited to every new problem. For machine learning, the concern is to be able to detect adequate patterns from heterogeneous, large, and sometimes uncertain datasets. In diagnosis, the necessity to quickly recognize a problem to provide a sure solution to solve it appears to be essential. One of the main challenges is the necessity to process heterogeneous data (qualitative, quantitative...) and sometimes to merge data obtained in different contexts. We developed a classification method based on fuzzy logic [1] capable of processing heterogeneous data types and noisy data. The LAMDA (Learning Algorithm for Multivariate Data Analysis) method is a classification method, capable to process three types of data: qualitative, quantitative, and intervals [2]. We addressed one of the main difficulties encountered in data analysis tasks: the diversity of information types. Such information types are given by

qualitative valued data, which can be nominal or ordinal, mixed with quantitative and interval data. Many situations leading to well-conditioned algorithms for quantitative valued information become very complex whenever there are several data given in qualitative form. In a non-exhaustive list, we can mention, rule based deduction, classification, clustering, dimensionality reduction... During the last decades, few research works have been directed to defy the issue of representing multiplicity for data analysis purposes [3, 11]. However, no standard principle has been proposed in the literature to handle in a unified way heterogeneous data. Indeed, a lot of proposed techniques process separately quantitative and qualitative data. In data reduction tasks for example, they are either based on distance measures for the former type [12] and on information or consistency measures for the later one. Whereas in classification and clustering tasks, eventually only a Hamming distance is used to handle qualitative data [4,11,14]. Other approaches are originally designed to process only quantitative data and therefore arbitrary transformations of qualitative data into a quantitative space are proposed without taking into account their nature in the original space [12,15,16]. For example, the variable *shape* can take values in a discrete unordered set {round, square, triangle}. These values are transformed respectively to quantitative values 1, 2, and 3. However, we can also choose to transform them to 3, 2 and 1. Another inverse practice is to enhance the qualitative aspect and discretize the quantitative value domain into several intervals, then objects in the same interval are labeled by the same qualitative value [17,18]. Obviously, both approaches introduce distortion and end up with information loss with respect to the original data. Moreover, none of the previously proposed approaches combines in a fully adequate way, the processing of symbolic intervals simultaneously with quantitative and qualitative data. Although extensive studies were performed to process this type of data in the Symbolic Data Analysis framework [19], they were focused generally on the clustering tasks [8, 10] and no unified principle was given to handle simultaneously the three types of data for different analysis purposes. In [2], a new general principle, was introduced as “Simultaneous Mapping for Single Processing (SMSP)”, enabling the reasoning in a unified way about heterogeneous data for several data analysis purposes. The fact that SMSP together with

¹This work was supported by the FUI/FEDER project MAISEO involving the companies VIVADOUR, CACG, GEOSYS, METEO FRANCE, PIONEER and laboratories CESBIO, LAAS-CNRS.

LAMDA can process simultaneously these three types of data without pre-processing is one of its principal advantages compared to other classical machine learning methods such as SVM (Support Vector Machine [20]), K-NN [21]. Decision trees are very powerful tools for classification and diagnosis [22] but their sequential approach is still not advisable to process multidimensional data since, by their very nature, they cannot be processed as efficiently as totally independent information [23]. A complete description of the LAMDA method and comparison with other classification techniques on various well known data sets can be found in [24, 25, 26]. Its other main characteristic is the fuzzy formalism which enables an element to belong to several classes simultaneously. It is also possible to perform clustering (i.e. with no a priori knowledge of the number and the class prototypes).

Besides the three existing types, we propose to add another type: the class type which can be processed simultaneously with the three former ones: quantitative, qualitative, intervals thanks to the ‘‘SMSP’’. In this configuration the class feature represents a meaningful aggregation of other features. This aggregation can be defined by a class determined by a previous classification, or the result of an abstraction. This new type gives the possibility to develop hierarchical classifications or to fuse different classifications. It allows an easier representation of many various and complex types of data, like multi-dimensional data, while being realistic and conserving their constraints. In a first part, the LAMDA method is briefly explained. The second part is devoted to the new type of data introduced: the object type. Finally, this new method is exemplified through an agronomical project.

2 The LAMDA method

The LAMDA method is an example of fuzzy logic based classification methods [9]. The classification method takes as input a sample x made up of N features. The first step is to compute for each feature of x , an adequacy degree to each class C_k , $k = 1..K$ where K is the total number of classes. This is obtained by the use of a fuzzy adequacy function providing K vectors of Marginal Adequacy Degree vectors (MAD). This degree estimates the closeness of every single sample feature to the prototype corresponding to its class. At this point, all the features are in a common space. Then the second step is to aggregate all these marginal adequacy degrees into one global adequacy degree (GAD) by means of a fuzzy aggregation function. Thus the K MAD vectors become K GADs. Fuzzy logic[1] is here used to express MADs and GADs, since the membership degree of a sample to a given class is not binary but takes a value in $[0,1]$. Classes can be known a priori, commonly determined by an expert and the learning process is therefore supervised, or classes can be created during the learning itself (unsupervised mode or clustering). Three types of features can be processed by the LAMDA method: quantitative, qualitative and intervals for the MAD calculation [2]. The membership functions $\mu(x)$ used by LAMDA are based on the generalization of a probabilistic rule defined on 0, 1 to the $[0,1]$ -space.

2.1 Calculation of MAD for quantitative features

The quantitative type allows the representation of numerical values, assuming that the including space is known as a defined interval. For this type of descriptor, membership functions can be used, such as the Gaussian membership function so that the membership function for the x^{th} sample descriptor to the k^{th} class is:

$$\mu_k^i(x_i) = \exp \frac{-(x_i - \rho_k^i)^2}{2\sigma_i^2} \quad (1)$$

or the binomial membership function:

$$\mu_k^i(x_i) = \rho_k^i x_i (1 - \rho_k^i)^{1-x_i} \quad (2)$$

where:

$\rho_k^i \in [0, 1]$ is the mean of the i^{th} feature based on the samples belonging to the class C_k , $x_i \in [0, 1]$ is the normalized x^{th} feature and σ_i the standard deviation of the i^{th} feature value based on the samples belonging to the class C_k .

2.2 Calculation of MAD for qualitative features

In case of qualitative feature, the possible values of the i^{th} feature forms a set of modalities such as $D_i = Q_1^i, Q_2^i \dots Q_m^i$ with m the total number of modalities. The qualitative type permits to express by words the different modalities of a criterion.

The frequency of a modality Q_l^i of the i^{th} feature for the class C_k is the quantity of samples belonging to C_k whose modality for their i^{th} feature is Q_l^i [1]. So each modality

$Q_l^i \in D_i$ has an associated frequency. Let θ_{kj}^i be the frequency of a modality Q_j^i for the class C_k . The membership function concerning the i^{th} feature is:

$$\mu_k^i(x_i) = (\theta_{k1}^i)^{q_1^i} * (\theta_{k2}^i)^{q_2^i} * \dots * (\theta_{km}^i)^{q_m^i} \quad (3)$$

where $q_l^i = 1$ if $x_i = Q_l^i$ and $q_l^i = 0$ otherwise, for $l=1, ..m$.

2.3 Calculation of MAD for interval features

Finally, to take in account the potential uncertainties or noises in data, we can use the interval representation [2]. The membership function for the interval type descriptors is regarded as being the similarity $S(x_i, \rho_k^i)$ between the symbolic interval value for the i^{th} feature x_i and the interval $[\rho_k^{i-}, \rho_k^{i+}]$ which represents the value of the i^{th} feature for the class C_k , so that:

$$\mu_k^i(x_i) = S(x_i, \rho_k^i) \quad (4)$$

Let ω be defined as the scalar cardinal of a fuzzy set in a discrete universe as $\varpi[X] = \sum_{\xi \in V} \mu_x(\xi_i)$.

In case of a crisp interval, it becomes:

$$\varpi[X] = \text{upperBound}(X) - \text{lowerBound}(X).$$

Given two intervals $A=[a^-, a^+]$ and $B=[b^-, b^+]$, the distance is defined as:

$$\delta[A, B] = \max\left[0, \left(\max\{a^-, b^-\} - \min\{a^+, b^+\}\right)\right] \quad (5)$$

and the definition of the similarity measure between two crisp intervals:

$$S(I_1, I_2) = \frac{1}{2} \left(\frac{\varpi[I_1 \cap I_2]}{\varpi[I_1 \cup I_2]} + 1 - \frac{\delta[I_1, I_2]}{\varpi[V]} \right) \quad (6)$$

The similarity combines the Jaccard's similarity measure which computes the similarity when the intervals overlap, and a second term which allows taking into account the case where the intervals are not straddled.

2.4 Calculation of feature weights

It is possible to determine the relevance of a feature to optimize the separation between classes. The MEMBAS method [8, 9] is a feature weighting method based on a membership margin. A distinguishable property of this method is its capability to process problems characterized by mixed-type data (quantitative, qualitative and interval). It lies on the maximization of the margins between two closest classes for each sample. It can be expressed as:

$$\text{Max}_{w_f} \sum_{j=1}^J \beta_j (w_f) = \frac{1}{N} \sum_{j=1}^J \left\{ \begin{array}{l} \sum_{i=1}^N w_{fi} \mu_c^i(x_i^{(j)}) \\ - \sum_{i=1}^N w_{fi} \mu_{\tilde{c}}^i(x_i^{(j)}) \end{array} \right\} \quad (7)$$

Subject to the following constraints: $\|w_f\|_2^2 = 1, w_f \geq 0$.

The first constraint is the normalized bound for the modulus of w_f so that the maximization ends up with non-infinite values, whereas the second guarantees the nonnegative property of the obtained weight vector. Then can be simplified as:

$$\begin{array}{l} \text{Max}_{w_f} (w_f)^T s \\ \text{Subject to } \|w_f\|_2^2 = 1, w_f \geq 0 \end{array} \quad (8)$$

where: $s = \frac{1}{N} \sum_{j=1}^J \{U_{jc} - U_{j\tilde{c}}\}$ and

$U_{jc} = \left[\mu_c^1(x_i^{(j)}), \dots, \mu_c^N(x_i^{(j)}) \right]$, $\mu_c^i(x_i^{(j)})$ is the membership function of class c (c corresponds to the "right" class for sample $x^{(j)}$), \tilde{c} the closest class evaluated at the given value $x_i^{(j)}$ of the i^{th} feature of pattern $x^{(j)}$. s is computed with respect to all samples contained in the data base excluding $x^{(j)}$ ("leave-one-out margin").

This optimization problem has an analytical solution determined by the classical Lagrangian method. Details of the method can be found in [9].

3 The new object type

In order to allow the combination of various data types into one single global object and therefore to support multi-dimensional features, we develop a novel data type. Each feature of an object descriptor can be described by a measured value and an extrinsic object-related weight. A sample GAD calculus formula is then the weighted mean of all MADs:

$$GAD_k^j = \sum \left(MAD_k^{ji} \cdot \tilde{w}_{fi} \right) \text{ for } j=1 \dots J \quad (9)$$

where MAD_k^{ji} = MAD of the j^{th} sample for the i^{th} feature to class k and $\tilde{w}_{fi} \in [0, 1]$ = Normalized value of weight w_{fi} of the i^{th} feature determined by the MEMBAS method, and J is the total number of samples which have been classified.

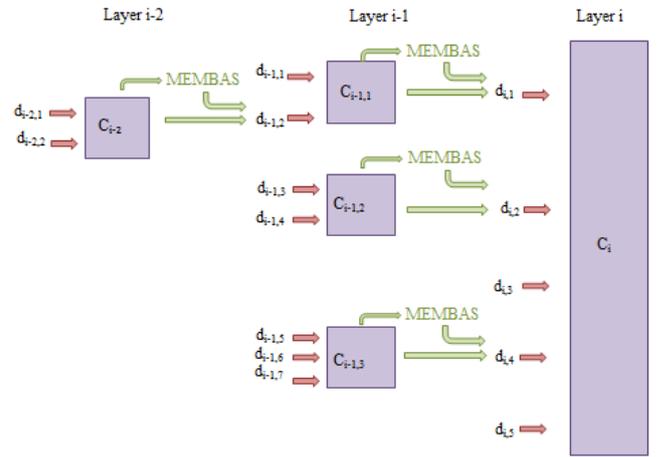


Figure 1: LAMDA architecture

The main advantage of using this new object-oriented data type is to capture the distinct features of a same object as a whole. An object of layer $i-1$ is regarded as one single feature for the layer i then can be processed as all other descriptors. The weights of the descriptors composing the objects are determined using MEMBAS once the clustering is finished for the layer $i-1$. An object is regarded as being a combination of features, each of which is associated to its weight. In other words, an object regarded as a single entity in reality can be processed as a complex unit. For instance, the weather can be considered as a global concept but also as detailed data (rain, temperature, etc...). All of its features are parts of a same object and are strongly connected together. That realistic consideration implies several distinct clustering layers. The layer i concerns the classification of a sample set called A and the $i-1$ one involves some of their constituent units. Obviously, a second layer of classification is consistent only in case at least one of the sample features is a complex entity. Therefore, for each sample of the set, an object feature becomes itself a whole sample in the layer $i-1$ and is compared to the others

to constitute a new sample set called B . Then a classification of the B samples is processed. Once the classification of the B samples has been done, its results are used to compute the classification of A . If the samples of the A set have C complex features, the second classification level implies C distinct sample sets B_1, B_2, \dots, B_C thus C distinct classifications.

The MEMBAS algorithm [8, 9] can then calculate the weights of every feature for the classes definition. It is applied on the B samples so that its involved features become the weighted components of a meaningful object. The complex features of an A sample is then a balanced combination of attributes.

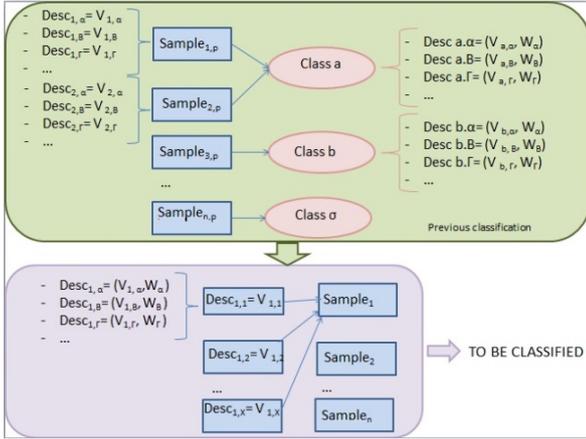


Figure 2: Principle of hierarchical classification

As explained in the Figure 2, the sample $Sample_1$ is described by X features, including the object-type feature $Desc_{1,1}$. $Desc_{1,1}$ is described by $Desc_{1,\alpha}$, $Desc_{1,\beta}$, etc.

To get their respective importance W_α , W_β etc in $Desc_{1,1}$ description, a previous classification is performed regarding $Desc_{1,1}$ as a sample ($Sample_{1,p}$), so that each weight can be calculated using the MEMBAS algorithm [8, 9]. Once the respective weights of each feature are known, objects are automatically instantiated to be involved in the main classification. $Desc_{1,1}$ is then described in line with the obtained weights W_α , W_β and the known values $V_{1,\alpha}$, $V_{1,\beta}$.

2.5 Evaluation of a classification quality

The comparison of two classifications can be performed by measuring their respective compactness and their separation. Better the classes are compact and separated easier will be the recognition process.

A method to measure the quality of a partition has been proposed by [10]. This index measures the quality partition in terms of classes compactness and separation. This partition index is the *Clusters Validity Index (CV, Eq.(10))* which depends only on the GADs (membership degree of an individual to a class) and not explicitly on data values.

$$CV = \frac{Dis}{N} \cdot D_{\min}^* \cdot \sqrt{K} \quad (10)$$

where N is the total number of individuals in the data base and K the total number of classes.

Dis represents the dispersion given by:

$$Dis = \sum_{k=1}^K 1 - \frac{\sum_{j=1}^J \delta_{kj} \cdot \exp(\delta_{kj})}{N \cdot GAD_{Mk} \cdot \exp(GAD_{Mk})} \quad (11)$$

$$\text{with: } \delta_{kj} = GAD_{Mk} - GAD_k^j \quad \forall j, j \in [1, J] \quad (12)$$

$$\text{and } GAD_{Mk} = \max \left[GAD_k^j \right] \quad (13)$$

D_{\min}^* is the minimum distance between two classes. This distance is computed by using the distance $d^*(A, B)$ between two fuzzy sets A and B [8] defined by:

$$d^*(A, B) = 1 - \frac{M[A \cap B]}{M[A \cup B]} = 1 - \frac{\sum_{j=1}^J \min(GAD_A^j, GAD_B^j)}{\sum_{j=1}^J \max(GAD_A^j, GAD_B^j)} \quad (14)$$

The highest value of CV corresponds to a better partition.

4 Application to an agronomical project

The agronomical project aims at developing a diagnosis system for an optimized water management system and an efficient distinctive guidance for corn farmers in order to decrease the use of phytosanitary products and the water consumption for irrigation. The project involves two aspects. The first one aims at complementing the benefits of adopting and implementing the cultural profile techniques [28, 29]. In this context, we perform a classification of plots based on various agronomic and SAFRAN meteorological data [30], so that each plot should mostly belong to one particular class whose features are known. Thanks to the provided information stemmed from the classification results, advice can be offered to the corn farmers concerning the corn variety they should sow and the schedule they should follow for an optimized yield. This study includes two steps which are described in figure 3. The first one concerns the clustering of a training set of 50 plots, using the unsupervised LAMDA classification.

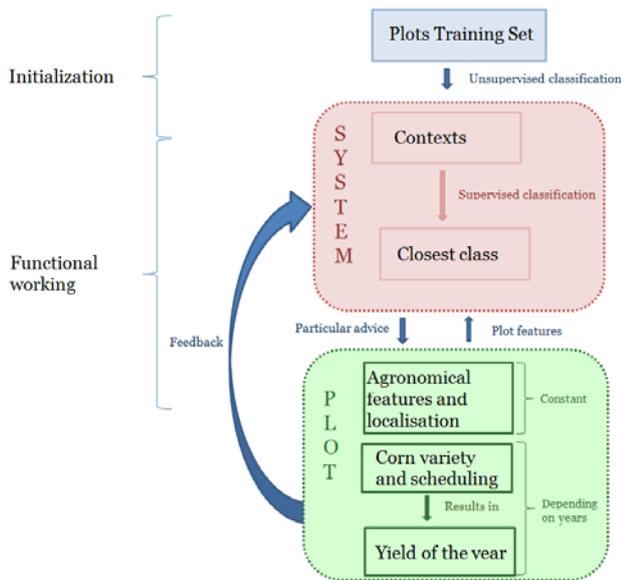


Figure 3: Learning System functioning

The data used for this classification are six distinctive agronomical descriptors, describing the plots' features and that are highly involved in their capacity for yield and water retention, and twenty-one weather features, defining the meteorological class in which the plot is situated. The second part of the project will be repeated annually to update and improve the clustering performed previously by adding new information returned by the farmers after harvest. In the following, only the first part is presented.

Firstly a previous meteorological clustering (A) is required to realize a realistic plot classification since the yield of seedling is highly related to the meteorological conditions. The weather is then regarded as a complex entity so that it is only one of a plot features. It is based on the historical meteorological data of the geographical position corresponding to the studied plot. Those descriptors refer to the temperature, the quantity of rainfall, and the evapotranspiration which occurred during three crucial periods of the year. Each feature is described in several distinctive ways. For instance, one period temperature is evaluated according three types of information. This meteorological clustering is an unsupervised classification based on weather data covering every single days of the determined periods during the fifty last years for all the geolocalized points belonging to the area studied in this project (South-West of France). In the event that the plot is part of the training set (studied area), the weather type of its area is known and the plot classification can be done directly. Otherwise, the weather type is obtained thanks to a supervised classification mode (B') delivering the most appropriate context. In any cases, the weather type is an object-feature. This hierarchical treatment permits to regard each meteorological type as a whole and let the weather contexts follow their natural evolution independently of agronomical variations. Moreover, considering the meteorological features as a single global object permits taking into account the environmental constraints and getting a realistic model. As we can observe in the Figure 4, the meteorological clustering (B) has permitted

to divide the area in three sub-areas. The results of clustering (B) and the meteorological supervised classification (B') have been first performed with every sample of the set and the distribution of the weights between the meteorological features has been determined.

The result of this classification is consistent and so, we can use the obtained classes and weights of the meteorological features (obtained with MEMBAS) as object-features in classification (A). To analyze the benefit of using hierarchical classification, a clustering (A') has been performed by using the twenty-one meteorological features separately and the agronomical features (twenty-seven features taken indistinctly). We can notice that the prototypes of the classes are highly dependent on the meteorological classes for clustering (A) while clustering (A') is mainly influenced by the ground type.

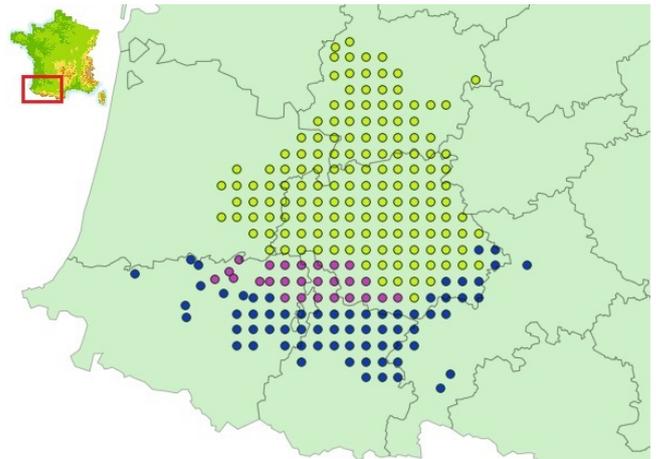


Figure 4: Meteorological sub-areas obtained with classification (B)

To enlighten this, we chose arbitrarily two very close classes containing the similar plots in both clustering. Each class prototype is described by the mean value of its marginal degree memberships (MAD). We represent in Figure 5 these prototype parameters for meteorological features only for both cases (A with diamond and A' with square) with in abscises, the marginal membership degree for class 1 and in ordinate the same marginal membership degree for class 2. For a better quantification of the benefits that the use of the object representation brings, the CV is systematically calculated in order to determine the better partition quality. The results are very encouraging since $CV = 0.69$ when the meteorological data are regarded as a whole object and 0.2 when they are treated separately. The object type representation enables to multiply by more than 3 this index and therefore the compactness of the obtained partition.

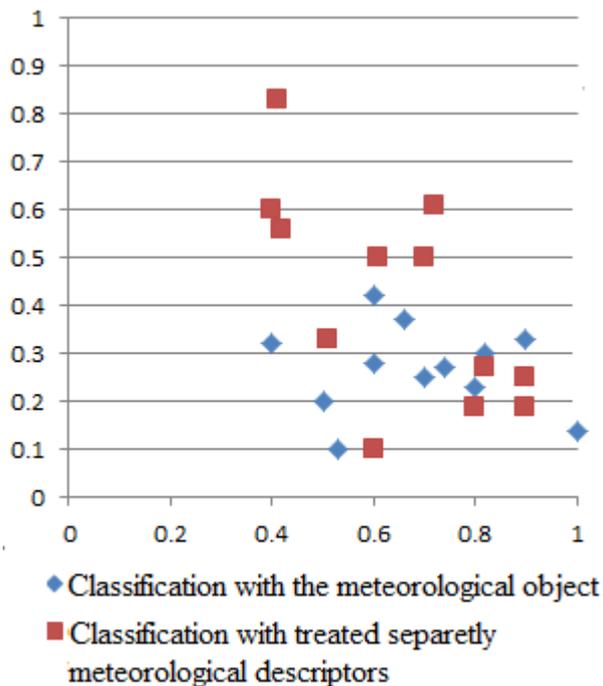


Figure 5: Meteorological prototypes for two close classes in case (A) and (A')

The second aspect of our implication in the project deals with the water utilization of various clusters of farmers with the aim of forecasting the needs of each cluster and adjusting the repartition. From this perspective, we realize an unsupervised classification of a training data-set of 2900 samples described by seven features: distance to the closest waterway, orientation, altitude... Orientation concerns cardinal points and we assume that it is not expressible with different modalities since continuity cannot be represented by qualitative descriptors. It cannot be a number nor an interval because of the cyclic form to be kept. Thus we choose to regard a cluster orientation as an object composed of two descriptors that correspond to the coordinates of its cardinal point in a trigonometric circle base. The orientation of each cluster can take eight different values: N, NE, E, SE, S, SW, W, and NW, which bring us to consider eight different combinations. In accordance with the trigonometrical circle, these eight combinations are respectively: $(0,1)$, $(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$, $(1,0)$, $(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$, $(0,-1)$, $(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$, $(-1,0)$, $(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$.

Once our results are validated by an expert, the classification is experimented twice: firstly treating each descriptor separately and secondly involving the object type. Such as meteorological data in the first example, the CV is calculated in order to determine the better partition quality.

In this case, which implies 2900 samples, $CV= 0.08$ when abscissa and ordinate are separated, and $CV= 0.13$ when using an orientation object. As in the first example, these results show a qualitative gain for the partition when the object type is used to express the semantically connected data.

4 Conclusion

This modular architecture allows more flexibility and a more precise treatment of data. As we can notice with the previous agronomical classification, the object approach makes each module able to be managed independently of the others so that they can evolve autonomously, depending on their own specific features and contexts. The object representation permits to preserve multi-dimensionality and makes fusion of datasets easier. A better overview is offered since we can percept the variations of each module distinctively and the evolution of their influences.

As a perspective, an agent-oriented architecture, based on the multi-agents theory [31] will be developed so that each sample could be considered independently of the others. They would be so able to create classes acting simultaneously and comparing themselves to the others, so that the classes definition won't depend on the samples order in the file anymore but will directly result from the samples set definition. This orientation will assure that the classification result of our method is unique and stable for a given samples set. We aim at developing some methods to allow a semantic data processing also.

References

- [1] D. Dubois and H. Prade editor. The three semantics of fuzzy sets, Fuzzy sets and systems, vol. 90,N° 2, pp141-150,Elsevier, London, 1997.
- [2] L. Hedjazi, J. Aguilar-Martin, M.V. Le Lann and T. Kempowsky, Towards a unified principle for reasoning about heterogeneous data: a fuzzy logic framework, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 20,N°2,pp. 281-302, World Scientific, 2012
- [3] R.S. Michalski and R.E. Stepp, automated construction of classifications: Conceptual clustering versus numerical taxonomy, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 4 (1980), pp. 396-410.
- [4] D.W. Aha, Tolerating noisy, irrelevant and novel attributes in instance based learning algorithms, *Int. Man-Machine Studies* 36 (1992), pp. 267-287.
- [5] S. Cost, S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, *Machine learning* (10) (1993), pp.57-78.
- [6] T. Mohri, T. Hidehiko, An optimal Weighting Criterion of case indexing for both numeric and symbolic attributes, in *D.W. Aha (Ed.)*, Case-based Reasoning: papers from the 1994 workshop. Menlo Park, CA:AIII Press, pp. 123-127.
- [7] C. Giraud-Carrier, M. Tony, An Efficient Metric for heterogeneous Inductive Learning Applications in the Attribute-Value Language, *Intelligent systems* (1995) 341-350.
- [8] K.C. Gowda, E. Diday, Symbolic clustering using a new similarity measure, *IEEE Trans. SMC* 22(2) (1992) 368-378.
- [9] Q.H Hu, Z.X. Xie, D.R. Yu, Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation, *Pattern Recognition* 40 (2007) 3509-3521.
- [10] F.A.T. De Carvalho, R.M.C.R. De Souza, Unsupervised Pattern Recognition Models for Mixed Feature-Type Symbolic Data, *Pattern Recognition Letters* 31 (2010) 430-443.

- [11] I. Kononenko, Estimating Attributes: Analysis and Extensions of Relief, *Proc. European Conf. Mach. Learning ECML* (1994), pp. 171-182.
- [12] K. Kira, L. Rendell, A practical approach to feature selection. *In proced. 9th Int'l Workshop on Machine Learning* (1992), pp. 249-256.
- [13] M. Dash, H. Liu, Consistency-based search in feature selection, *Artif. Intell.* 151 (2003) 155–176.
- [14] D.W. Aha, Incremental, instance-based learning of independent and graded concept descriptions, *in Proced. Of the 6th int'l Mach. Learning Workshop.* (1989) 387–391.
- [15] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, V. Vapnik, Feature Selection for SVMs, *Advances in Neural Information Processing Systems* (2001), pp. 668-674.
- [16] T. Cover, P. Hart, Nearest neighbor pattern classification, *IEEE Trans. Inf. Theory* 13 (1967), pp.21-27.
- [17] H. Liu, F. Hussian, C.L. TAM, M. Dash, Discretization: an enabling technique, *J. Data Mining and Knowledge Discovery* 6(4) (2002) 393–423.
- [18] M.A. Hall, Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning, *Int. Conf. Mach. Learning ICML* (2000), pp. 359-366.
- [19] H.H. Bock, Diday E. Analysis of Symbolic Data, Exploratory methods for extracting statistical information from complex data. (Springer, Berlin Heidelberg,2000).
- [20] V. Vapnik, The Nature of Statistical Learning Theory Data Mining and Knowledge Discovery ,pp. 1-47, 6, Springer Verlag, 1995
- [21] T. Cover and P. Hart, Nearest neighbor pattern classification Information Theory, IEEE Transactions on,13,1,pp. 21-27, 1967
- [22] Michie D., Spiegelhalter D.J., Taylor C.C., *Machine Learning, Neural and Statistical Classification*, Ellis Horwood series in Artificial Intelligence, february, 1994
- [23] Rakotomalala R., Decision Trees, review MODULAD, 33, 2005.
- [24] J. C. Aguado and J. Aguilar-Martin, A mixed qualitative-quantitative self-learning classification technique applied to diagnosis, The Thirteenth International Workshop on Qualitative Reasoning, (QR'99)pp. 124-128, 1999.
- [25] L. Hedjazi, J. Aguilar- Martin, M.V. Le Lann, Similarity-margin based feature selection for symbolic interval data, *Pattern Recognition Letters*, Vol.32, N°4, pp. 578-585, 2012
- [26] L. Hedjazi, J. Aguilar-Martin, M.V. Le Lann, and Tatiana Kempowsky-Hamon, Membership-Margin based Feature Selection for Mixed Type and High-dimensional Data: Theory and Applications, *Information Sciences*, *accepted to be published*, 2015.
- [27] C. V. Isaza , H. O. Sarmiento, , T. Kempowsky-Hamon , M.V. Le Lann , Situation prediction based on fuzzy clustering for industrial complex processes, *Information Sciences*, Volume 279, 20 September 2014, pp. 785-804, 2014
- [28] Henin S., Gras R., Monnier G., 1969, *Le profil culturel (2^e edition)*, Masson Ed. Paris.
- [29] Gautronneau Y., Gigeux C., 2002, *Towards holistic approaches for soil diagnosis in organic orchards*, Proceedings of the 14th IFOAM Organic World Congress, Victoria, p 34.
- [30] Quintana-Seguí, P., Le Moigne, P., Durand, Y., Martin, E., Habets, F., Baillon, M., ... & Morel, S. (2008). Analysis of near-surface atmospheric variables: Validation of the SAFRAN analysis over France. *Journal of applied meteorology and climatology*, 47(1), 92-107.
- [31] Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence* (Vol. 1). Reading: Addison-Wesley.

A Bayesian Framework for Fault diagnosis of Hybrid Linear Systems

Gan Zhou¹ Gautam Biswas² Wenquan Feng¹ Hongbo Zhao¹ and Xiumei Guan¹

¹ School of Electronic and Information Engineering, Beihang University, Beijing, China

email: zhouganterry@hotmail.com; buaafwq@buaa.edu.cn;

bhzhb@buaa.edu.cn; guanxm@buaa.edu.cn

² Institute for Software Integrated Systems, Vanderbilt University, Nashville, USA

email: gautam.biswas@vanderbilt.edu

Abstract

Fault diagnosis is crucial for guaranteeing safe, reliable and efficient operation of modern engineering systems. These systems are typically hybrid. They combine continuous plant dynamics described by continuous-state variables and discrete switching behavior between several operating modes. This paper presents an integrated approach for online tracking and diagnosis of hybrid linear systems. The diagnosis framework combines multiple modules that realize the hybrid observer, fault detection, isolation and identification functionalities. More specifically, a Dynamic Bayesian Network (DBN)-based particle filtering (PF) method is employed in the hybrid observer to track nominal system behavior. The diagnostic module combines a qualitative fault isolation method using hybrid TRANSCEND, and a quantitative estimation method that again employs a DBN-based PF approach to isolate and identify abrupt and incipient parametric faults, discrete faults and sensor faults in a computationally efficient manner. Finally, simulation and experimental studies performed on a hybrid two-tank system demonstrate the effectiveness of this approach.

1 Introduction

The increasing complexity of modern industrial systems motivates the need for online health monitoring and diagnosis to ensure their safe, reliable, and efficient operation. These systems are typical hybrid involving the interplay between discrete switching behavior and continuous plant dynamics. More specifically, the system configuration changes consist of known controlled mode transitions generated from external supervisory controller and autonomous mode transitions triggered by internal variables crossing boundary values. The continuous dynamic behavior is modeled by continuous-state variables that are a function of the particular discrete mode of operation. As a result, tasks like online monitoring and diagnosis have to seamlessly integrate continuous behaviors interspersed with discrete transitions that often require model switching to accommodate the discrete transitions [1].

For complex hybrid systems, faults will typically affect the continuous behavior and the discrete dynamics of the

system. Some faults may be parametric, and they directly affect the continuous behavior, others are discrete, thus they directly affect the mode of system operation. Both types of faults also have indirect effects on the other type of behavior. Moreover, faults can have different time-varying profiles, such as abrupt faults, intermittent faults and incipient faults [2]. In addition, faults may occur in the plant, the actuators and the sensors. The diagnosis of multiple fault types in the same framework is challenging, because some faults may produce similar effects in the particular measurements. Therefore, the diagnosis approach should provide more discriminatory power.

Previous model-based diagnosis approaches of hybrid systems were developed separately for parametric faults or discrete faults. For example, [1], [3] combined system monitoring with an integrated approach: qualitative and quantitative fault isolation to generate, refine, and identify parametric faults. [4]-[5] are typical discrete fault diagnosis approaches, which modeled the discrete faults as fault modes, and relied on estimating the system behavior for diagnosis. In recent years, some integrated approaches have been proposed for diagnosis of parametric and discrete faults together. [6] introduced a global ARRs (GARRs)-based mode diagnoser to track discrete system modes, and combined it with a quantitative approach to diagnose discrete and abrupt or incipient parametric faults within a common framework. The approach presented in [7] monitored system behavior using a timed Petri-Net model and mode estimation techniques, and isolated the faults by means of a decision tree approach. Unfortunately, this method was application-specific, and was not generalized.

Our goal in this paper is to propose an integrated model-based approach to diagnose single and persistent incipient or abrupt parametric faults, discrete faults and sensor faults in hybrid linear systems. This extends our earlier work [8] from continuous systems to hybrid systems. A PF technique using switched DBN is adopted for tracking nominal hybrid system behavior. When a non-zero residual value is detected using a statistical hypothesis testing method, this fault detection scheme triggers the fault isolation and identification modules. We combine a fast qualitative fault isolation (Qual-FI) scheme using the hybrid TRANSCEND approach [1] with quantitative fault isolation and identification (Quant-FII) scheme based on a PF-based parameter estimation technique to support the diagnosis of multiple faults types in hybrid linear systems. The

Quant-FII scheme derives a switched faulty DBN model for each fault hypothesis that remains when the switch from Qual-FI to Quant-FII is initiated. In addition, Quant-FII is also designed to estimate possible parameter values [8].

The rest of this paper is organized as follows. Section 2 briefly presents the different models employed in our diagnosis approach and some basic definition of the different types of faults. A hybrid two-tank system is used as a running example to explain the hybrid bond graph modeling method and the derivation of temporal causal graph and DBN from hybrid bond graph models. Section 3 gives a brief overview of our diagnosis architecture, and then presents our online tracking and fault detection, qualitative fault isolation and quantitative fault isolation and identification schemes in some detail. Section 4 discusses the results of the application of our algorithm to the hybrid two-tank system. Finally, the discussion and conclusions of this paper are presented in the last section.

2 Theoretical Background

In this section, we formalize the basic definitions, concepts and notation of the modeling approach that goes in conjunction with our diagnosis architecture.

2.1 Hybrid Bond Graphs

Bond graphs (BGs) are a domain-independent topological-modeling language that captures energy-based interactions among the processes that make up a physical system [9]. The nodes in bond graphs represent components of dynamic systems including energy storage elements (capacities, C and inertias, I), energy dissipation elements (resistors, R), energy sources (effort source, Se and flow source, Sf) and energy transformation elements (gyrators, GY and transformers, TF). Bonds, drawn as half arrows, represent the energy exchange paths between the bond graph elements. Two junctions (1 and 0), also modeled as nodes, represent the equivalent of series and parallel topologies respectively.

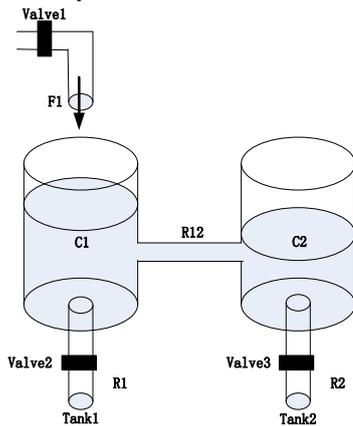


Figure 1 Schematics of hybrid two-tank system

Hybrid bond graphs (HBGs) extend BGs by introducing switched junctions to enable discrete changes in the system configuration [10]. The switched junctions may be dynamically switched on and off as system behavior evolves. When a switched junction is on, it behaves as a normal junction. When off, the 1 and 0 junctions behave as sources

of zero flow and zero effort, respectively. The dynamic behavior of switched junctions is implemented by a finite state machine control specification (CSPEC). A CSPEC defines finite number of states, and captures controlled and autonomous changes.

The hybrid two-tank system, shown in Figure 1, is the running example we employ in this paper. This system consists of two tanks connected by a pipe, a source of flow into the first tank, and drain pipes at the bottom of each tank. Three valves valve1, valve2 and valve3 can be turned on and off by commands generated from the supervisory controller. When the liquid level in tanks 1 (h_1) and/or 2 (h_2) reaches the height at which pipe R_{12} is placed (h), a flow is initiated through pipe R_{12} . The autonomous mode changes associated with this pipe are triggered when the liquid level in tank1 and/or tank 2 goes above or below the height of the pipe R_{12} . We assume five sensors: M_1 and M_2 measure the outflow from tank 1 and tank 2, respectively. M_3 measures the flow through the autonomous pipe R_{12} , and M_4 and M_5 measure the liquid pressure in tank 1 and tank 2, respectively.

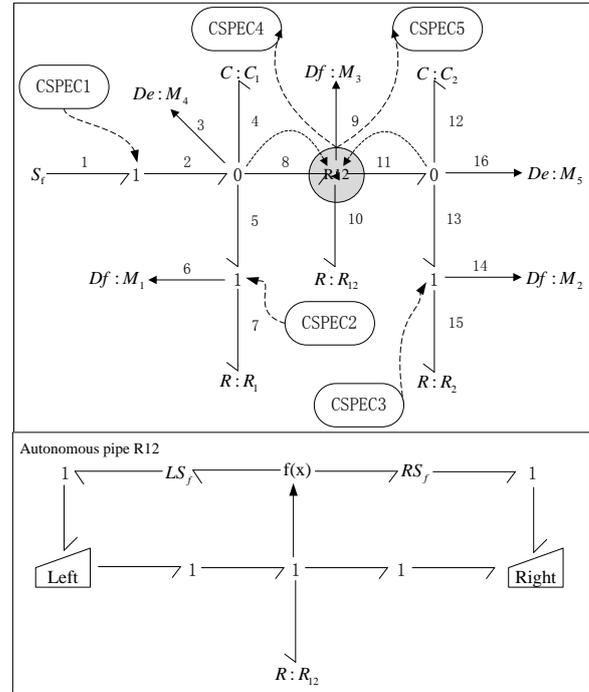


Figure 2 Hybrid bond graph of the plant

Figure 2 illustrates the HBG model for the plant in Figure 1 (The HBG model for autonomous pipe R_{12} is shown separately at the bottom part of Figure 2). The tanks and pipes are modeled as fluid capacitances C and resistances R , respectively. Measurement points occur at junctions. They are denoted by elements with symbols De for effort variable measurements and Df for flow variable measurements. Moreover, the two-tank system has five switched junctions: the CSPEC1, CSPEC2 and CSPEC3 describe the control logic for the three valves. CSPEC4 and CSPEC5 together capture the autonomous mode transitions of the connecting

pipe between the two tanks. Figure 3 (a) shows the CSPEC for a valve controlled by the switching signal sw . Figure 3 (b) shows CSPEC4 that describes the state of the left tank. When the liquid height in tank1 is below that of the autonomous pipe R_{12} , that state is OFF. If the liquid level exceeds the height of the pipe, this CSPEC transitions to the ON state. Similarly, CSPEC5 denotes the state of the right tank, and the mode of the autonomous pipe depends on the combination of these two CSPECs. Table 1 shows the discrete mode for pipe R_{12} and the corresponding state of CSPEC4 and CSPEC5 in detail. The corresponding bond graph configurations are described in [15].

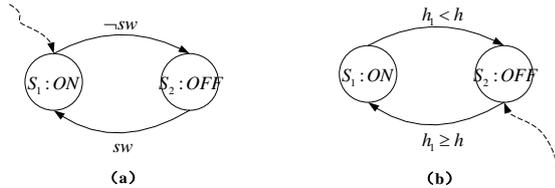


Figure 3 (a) Controlled transition; (b) Autonomous transition for CSPEC4

Table 1 Four different possible configurations for autonomous pipe R_{12}

Mode	Constraint Function	CSPEC4	CSPEC5
1	$h_1 \geq h \wedge h_2 < h$	ON	OFF
2	$h_1 < h \wedge h_2 \geq h$	OFF	ON
3	$h_1 < h \wedge h_2 < h$	OFF	OFF
4	$h_1 \geq h \wedge h_2 \geq h$	ON	ON

The temporal causal graph (TCG) is a signal flow diagram that captures the causal and temporal relations between system variables, and can also be systematically derived from a BG [11]. In our work, we can efficiently reason about the qualitative behavior of each continuous mode of hybrid system behavior using the TCG when a fault is detected. Formally, a TCG is defined as follows [2]:

Definition 1 (Temporal Causal Graph): A TCG is a directed graph that can be denoted by a tuple $\langle V, L, D \rangle$. $V = E \cup F \cup S \cup M$ is a set of vertices involving effort variables E , flow variables F , discrete fault event S and measurement M in hybrid bond graph model. L is a label set $\{1, -1, =, p, p^{-1}, N, Z, p \cdot dt, p^{-1} \cdot dt\}$. The propagation type of first seven labels is instantaneous, and the last two are temporal. $D \subseteq V \times L \times V$ is a set of edges.

For lack of space, the TCG for hybrid two-tank system is not shown in this paper, but the algorithms for deriving TCGs directly from bond graph model can be found in [2]. It should be noted that for each mode of operation, the TCG may need to be re-derived to capture the changes in the BG model configuration when mode transitions occur.

2.2 Dynamic Bayesian Networks

Assuming that the system is Markovian and time-invariant, we can model the system as a two-slice temporal Bayes net that illustrates not only the relations between system variables at any time slice t , but also the across-time relations

between the variables [12]. The system variables consists of four different set of variables (X, Z, U, Y) , which denotes the continuous state variables, other hidden variables, input variables and measured variables for dynamic system, respectively. The relations between these variables can be generated as equations in the state space formalism. The across-time links between the successive times slice t and $t+1$ are derived as transition equations between the state variables in the system. Since the TCG describes the causal constraints between system variables, the DBN can be easily constructed from TCG. More details of this process are presented in Lerner, et al. [13].

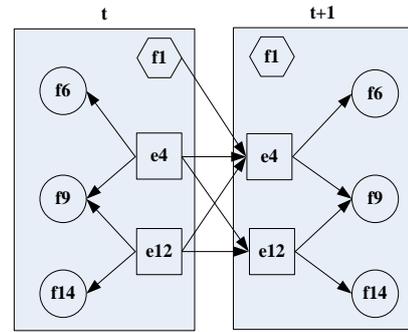


Figure 4 Nominal DBN

When all the valves are ON and the liquid level in tank1 and tank2 are above the height of the autonomous pipe R_{12} , the nominal DBN model for hybrid two-tank system is shown in Figure 4. This DBN model derived from the TCG as the following random variables: the continuous state variables $X = \{e_4, e_{12}\}$ presents the pressures at the bottom of each tank, input variables $U = \{f_1\}$ denotes the input flow into tank 1, and measured variables $Y = \{f_6, f_9, f_{14}\}$ indicates the outflow from tank1, the flow through the autonomous pipe R_{12} and the outflow from tank 2.

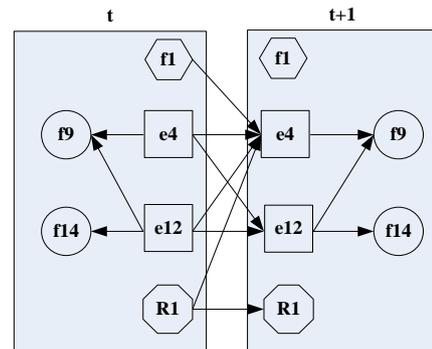


Figure 5 Single DBN model for both abrupt and incipient parametric fault

Since the discrete faults only influence the system mode, but not parameter variables, the DBN fault model corresponding to discrete fault will be constructed from the TCG in the particular discrete mode. For parametric faults, the DBN fault model is generated on the basis of nominal DBN model by augmenting a new random variable for each fault candidate. Figure 5 shows DBN model with parametric faults represented explicitly for the hybrid two-tank system.

The abrupt fault R_1^{+a} and incipient fault R_1^{+i} are represented in the same model. When the fault occurs, fault parameter R_1 becomes the additional state variable that need to be tracked.

2.3 Modeling Faults

In this paper, we focus on the diagnosis of persistent single faults. We consider incipient or abrupt parametric faults and discrete faults occurring in hybrid linear systems, as well as sensor faults. The precise definition for these faults can be given as follow.

Definition 2 (Incipient parametric fault): An incipient fault profile is defined by a gradual drift in the corresponding component parameter value $p(t)$ from the fault occurrence time t_f . The incipient fault parameter $p^i(t)$ can be described by:

$$p^i(t) = \begin{cases} p(t) & t \leq t_f \\ p(t) + d(t) = p(t) + \sigma_p^i(t - t_f) & t > t_f \end{cases} \quad (1)$$

where $d(t) = \sigma_p^i(t - t_f)$ is a linear function with a constant slope σ_p^i that added to the nominal parameter value from the time point of fault occurrence. Our approach to isolation and identification of incipient fault parameters is to calculate this constant slope σ_p^i [8].

Definition 3 (Abrupt parametric fault): An abrupt parametric fault is characterized by step changes in nominal component parameter value $p(t)$ from the fault occurrence time t_f . The abrupt fault parameter $p^a(t)$ is given by:

$$p^a(t) = \begin{cases} p(t) & t \leq t_f \\ p(t) + b(t) = p(t) + \sigma_p^a \cdot p(t) & t > t_f \end{cases} \quad (2)$$

where $b(t) = \sigma_p^a \cdot p(t)$ is a step function that gets added to the parameter value from the time point of fault occurrence. σ_p^a is the percentage change in the parameter expressed as a fraction, and our goal is to estimate this value [8].

Definition 4 (Discrete fault): A discrete fault manifests as a discrepancy between the actual and expected mode of a switching element in the model [2].

Discrete faults occur in discrete actuators, like valves and switches that operate in discrete modes (e.g., *on* and *off*). Consider the example of a valve, it may be commanded to close, but remain stuck open. Also, it may unexpectedly open or close without a command. This type of fault manifests as an unexpected system mode change, unlike parametric faults, which cause deviations in continuous behavior.

Definition 5 (Sensor fault): A sensor fault is a discrepancy between the measurement and actual value in the model.

In this paper, we only consider sensor bias fault, which can be represented as:

$$m^b(t) = \begin{cases} m(t) & t \leq t_f \\ m(t) + \Delta_m^b & t > t_f \end{cases} \quad (3)$$

where $m(t)$ is the true value, and Δ_m^b is the sensor bias term.

3 Diagnosis Approach of Hybrid Linear Systems

Our integrated diagnosis approach for hybrid linear systems (See Figure 6) combines the Hybrid TRANSCEND approach [2] with switched DBN-based PF scheme [14] together, which diagnoses abrupt or incipient parametric faults, discrete faults and sensor faults in a common framework. It includes three main parts: system monitoring, qualitative fault isolation (QFI) and quantitative fault isolation and identification (QFII). These three steps are summarized below.

Initially, a nominal DBN is constructed from the current TCG model. A hybrid observer uses a PF-based nominal DBN model to track the system behavior in individual modes of operation. At the same time, a finite automata method in hybrid bond graph scheme implements the CSPECs, executes controlled and autonomous mode changes, and determines the system model for hybrid observer.

The fault detection continually monitors the statistically significant deviations between the observation $y(t)$ and estimation $\hat{y}(t)$ generated by hybrid observer. Once a fault is determined, QFI is triggered to generate the initial fault hypothesis, and refine them as additional deviations are observed. When remaining fault hypothesis set satisfies particular condition, the QFII scheme is invoked to run in parallel with QFI. The goal of this scheme is to refine the fault hypothesis further and estimate the value of the fault parameter. The following subsections describe these steps in more detail.

3.1 Online Tracking and Fault Detection

Since the hybrid system is piecewise continuous, discrete mode changes of the hybrid system have to be detected accurately as the continuous behavior of the system evolves. In our work, we have designed hybrid observers that are based on the nominal DBN-based PF scheme to track the continuous behavior in individual modes of operation. PF is a general purpose Markov chain Monte Carlo method that approximates the belief state using a set of samples or particles, and keeps the distribution updated as new observations are made over time. Moreover, the PF approach for DBNs exploits the sparseness and compactness of the DBN representation to provide computationally efficient solutions, because each measured variable in a DBN typically depends on some but not all continuous state variables.

For discrete mode changes, the finite state machine (FSM) for each switched junction determines mode transitions. Since the continuous behavior and discrete mode changes will interact with each other as system evolves, the FSM needs to execute controlled or autonomous mode changes. Explicit controlled changes are relatively simple, but the autonomous mode changes depend on the internal continuous variables. If mode changes occur, the hybrid observer will regenerate the nominal DBN model from TCG in new mode, and use the PF to continuously track system dynamic behavior. The online tracking algorithm for hybrid systems is shown in Algorithm 1.

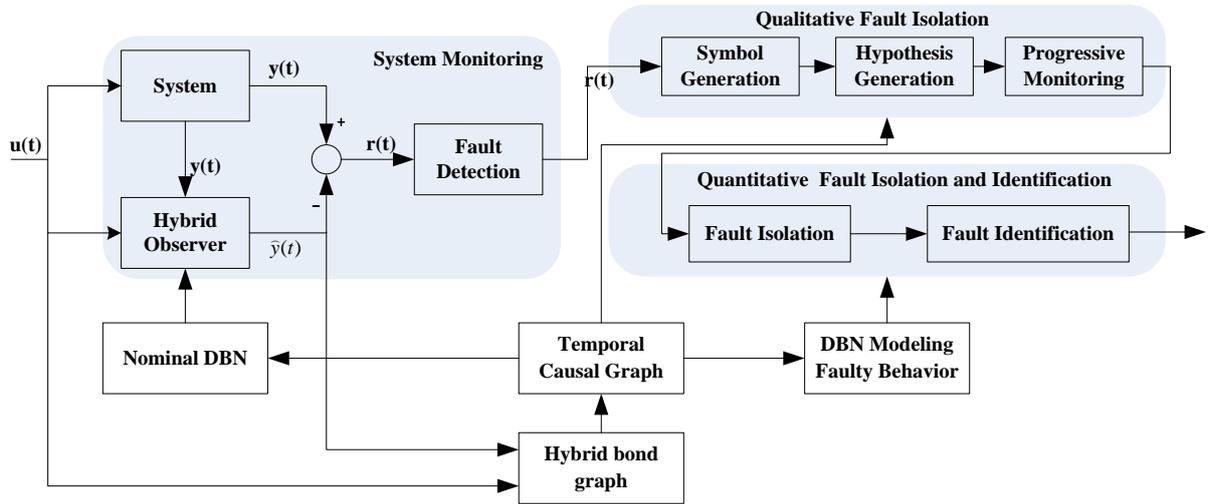


Figure 6 The diagnosis architecture

Algorithm 1: Online tracking algorithm

Input: Number of particles, N ; a initial DBN model $D = \{X, Z, U, Y\}$

For each particle i , from 1 to N do

Sample X_0^i from the prior probability distribution

Assign Y_0^i as the measurement at time step 0

End For

For each time-step $t > 0$ do

If the controlled or autonomous mode change occurs

Regenerate a DBN model D' from TCG in new system configuration

End If

Prediction: Sample each particle in DBN model D'

Weighting: Compute the weight considering the observation

Resampling: Normalize the weighted samples, and resample N new samples

Calculate the estimated continuous state variables X_t and Y_t at time step t

End For

The fault detection module compares the measured variable $y(t)$ from sensors with its estimate, $\hat{y}(t)$ computed by the hybrid observer at each time-step t . Ideally, any inconsistency $r(t) = y(t) - \hat{y}(t)$ implies a fault, and invokes the qualitative fault isolation module. However, to account for noise in the measurements and modeling errors, statistical techniques are employed to determine significant deviations from zero for the residual. In this paper, a Z-test, which uses a sliding window to compute the residual mean and variance, is adopted by reliable fault detection with low false-alarm rates [3].

3.2 Qualitative Fault Isolation

The QFI scheme is based on qualitative fault signature (QFS) method, which was proposed by Mosterman and Biswas [11] and then extended by Narasimhan and Biswas

[1] to hybrid systems. Daigle, et al. [2] extended this method to model discrete and sensor faults in continuous and hybrid systems. All of these methods are based on a formal definition of fault signature as follows:

Definition 6 (Qualitative Fault Signature): Given a fault f and measurement m , the qualitative fault signature can be denoted by $QFS(f, m) = \{(s_1, s_2, s_3), s_1, s_2 \in (+, -, 0, *), s_3 \in (N, Z, X, *)\}$; where \pm and 0 indicate an increase, decrease, and no change for residual magnitude or slope. N, Z and X imply zero to nonzero, nonzero to zero, and no discrete change behavior in the measurement from the estimate. * denotes the ambiguity in the signatures.

Table 2 Selected fault signature for hybrid two-tank system for the mode when all the valves are open and liquid level in both tanks are above the height of the autonomous pipe

Fault	f_6	f_9	f_{14}
C_1^{-a}	(+, X)	(+, X)	(0+, X)
C_1^{-i}	(0+, X)	(0+, X)	(0+, X)
R_1^{+a}	(-, X)	(0+, X)	(0+, X)
R_1^{+i}	(0-, X)	(0+, X)	(0+, X)
$v1.off$	(0-, X)	(0-, X)	(0-, X)
$v2.off$	(-, X)	(0+, X)	(0+, X)
f_6^+	(+0, *)	(00, X)	(00, X)
f_6^-	(-0, *)	(00, X)	(00, X)

When measurement deviations are detected, the symbol generator module in QFI scheme is triggered to calculate the QFS for the current mode of operation. However, since the fault may have occurred but not detected in an earlier mode, the fault hypothesis generation module rolls back to find the previous modes in which fault may have occurred, and generate fault hypothesis set $F = \{(f_i, \lambda_i, q_i)\}$, where λ_i denotes the deviation of fault parameter value, and q_i indicates the possible modes. The progressive monitoring module applies the forward propagation algorithm to continually refine the fault candidates in the fault hypotheses set. For hybrid systems, the progressive monitoring also has

to include forward propagation through mode changes, which makes the tracking algorithm much more complex. Narasimhan and Biswas [1] discuss the details of the roll back and roll forward algorithms used to support the progressive monitoring task. When a fault signature is no longer consistent with the observed measurements, and the changes cannot be resolved by autonomous mode transitions, this fault candidate is dropped.

The selected qualitative fault signature for hybrid two-tank system in particular mode is shown in Table 2. For incipient parametric faults, the QFS is shown as $(0\tau, s_3)$, where τ is the first nonzero symbol in the QFS for the abrupt faults with same system parameter. Sensor faults only affect the measurement provided by the sensor, so other measurements that are not affected are denoted by 00.

3.3 Quantitative Fault Isolation and Identification

Quant-FII scheme will be activated when any of the following conditions are fulfilled: 1) All the measurements have deviated from nominal, so the remaining fault candidates cannot be refined further only by the Qual-FI scheme; 2) The number of fault candidates has been reduced to a predefined value k ; 3) A predefined time l has elapsed. We restrict the length of Quant-FII scheme as a pre-specified value, and assume that no autonomous change occurs during this period.

The steps describing this scheme are illustrated as follows: First, a separate DBN faulty model will be constructed for each remaining fault candidate in the hypothesis set. Second, we combine each switched DBN faulty model with PF method to estimate the system behavior. Similar to fault detection scheme, a Z-test method is employed to detect the inconsistency between estimated values from PF and measurements. Ideally, only the correct true fault model will converge to the observed values of the measurements. Once the deviation is determined, the corresponding fault candidate will be dropped. This scheme runs in parallel with the qualitative fault isolation scheme, and if a controlled mode change occurs, these two schemes need to reload the DBN model for new system mode. This is the big difference between continuous systems and hybrid systems.

If the fault hypothesis cannot be refined further or only a single parametric or sensor fault candidate is left, fault identification scheme will be activated to identify the abrupt or incipient parametric fault in the same model and estimate the fault parameter value. We can use the PF result of the fault parameter to calculate the abrupt parameter fault magnitude σ_p^a , incipient parameter fault slope σ_p^i or sensor fault bias term Δ_m^b .

4 Experimental Results

To demonstrate the effectiveness of our approach, we apply it to the hybrid two-tank system in Figure 1. In this plant, the incipient parametric faults are modeled as gradual decrease in tank capacity and gradual increases in pipe resistances and denoted as $C_1^{-i}, C_2^{-i}, R_1^{+i}, R_2^{+i}$ and R_{12}^{+i} respec-

tively. The abrupt parameter faults are modeled as step decrease in tank capacity and step increases in pipe resistances and represented as $C_1^{+a}, C_2^{+a}, R_1^{+a}, R_2^{+a}$ and R_{12}^{+a} respectively. We consider discrete faults in each controlled valves including the valve gets stuck and valve changes mode without a command. For sensor faults, bias faults causing abrupt changes in the measurement are considered.

We assume that the tanks are initially empty, and start to fill in at a constant rate. The initial configuration of the system is all the valves are set to open. We will denote the system mode as q_{ijkm} , where i, j and k are the modes of valve1, valve2 and valve3 respectively, and m is the mode of autonomous pipe R_{12} . More specifically, the mode of valves includes $S_1 : on, S_2 : off, S_3 : Stuck_on$ and $S_4 : Stuck_off$. Therefore, the initial mode of the system is q_{1113} . At time step $t=6.7s$, the liquid level in tank 1 reaches the height of autonomous pipe R_{12} . The system mode transitions from q_{1113} into q_{1111} . Now the autonomous pipe R_{12} acts as an outflow pipe for the tank 1 but as flow source for the tank 2. As system evolves, the liquid level in tank 2 will also reach the autonomous pipe at time step $t=53s$. After that, system mode changes into q_{1114} . The experiments have been run for a total of 400s using a sampling period 0.1s. Gaussian white noise with zero mean and variances 0.018 is added to measurements.

4.1 Incipient Parametric Fault in R1

In this first experiment, we present our diagnosis approach for a fault scenario. A 10% rate of increase in pipe R_1 is injected as the incipient fault at time step $t = 60s$.

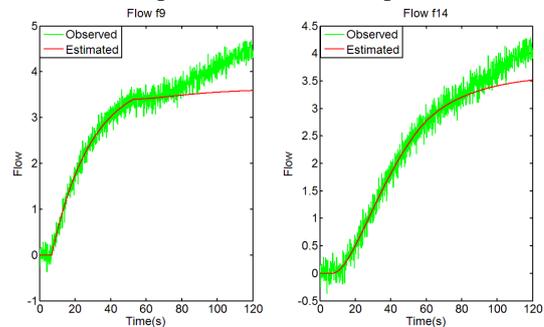


Figure 7 Observed and estimated result for nominal DBN model

We only consider the measurement M_3 and M_2 for the flow f_9 through the autonomous pipe R_{12} and the output flow f_{14} from tank 2. At time step $t=82s$, the fault detection scheme detects an increase in the flow f_9 , resulting in the initial fault hypothesis $F = \{(C_1^{-a}, q_{1114}), (C_1^{-i}, q_{1114}), (R_1^{+a}, q_{1114}), (R_1^{+i}, q_{1114}), (v2.off, q_{1414}), (f_9^+, q_{1114})\}$. At 88.4s, the flow f_{14} shows an increase above nominal (+). A possible autonomous transition is executed for the current inconsistent candidate (f_9^+, q_{1114}) . After that, the first order change of flow f_9 is determined to decrease and increase in mode

q_{1414} and q_{1114} at time steps $t=94.8s$ and $97.7s$, respectively, and finally the possible fault hypotheses are $F = \{(C_1^{-i}, q_{1114}), (R_1^{+a}, q_{1114}), (R_1^{+i}, q_{1114})\}$. According to the fault signatures in mode q_{1114} , these three candidates cannot be refined further using observed deviations. Figure 7 represents observed and estimated result generated by the nominal DBN model.

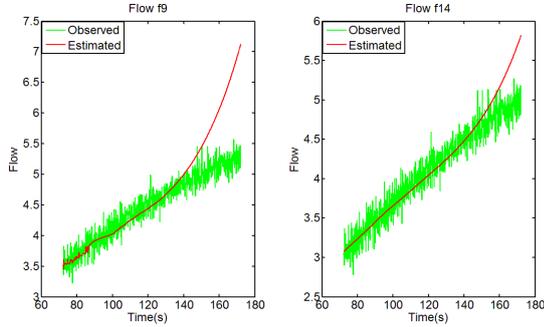


Figure 8 Estimated observation using fault model C_1^{-i}

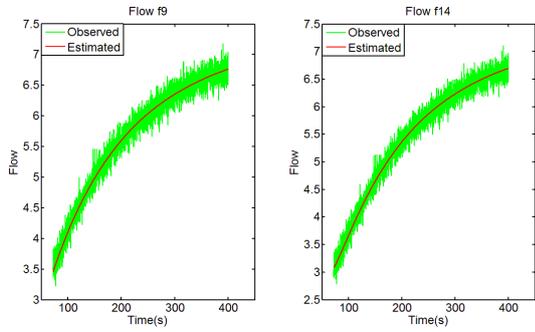


Figure 9 Estimated observation using fault model $R_1^{+a/i}$

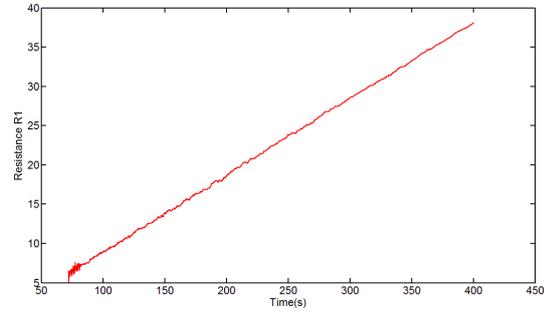


Figure 10 Estimated value of true fault parameter R_1^{+i}

The QFII scheme is initiated at time step $t=72s$, and two separate DBN fault model using C_1^{-i} and $R_1^{+a/i}$ are constructed. As more measurements are obtained, the Z-tests indicate a deviation in the measurement estimates obtained by the fault model C_1^{-i} , and the estimation generated by possible true fault model $R_1^{+a/i}$ is consistent with measurement. The quantitative fault identification part estimates the value of R_1 , and determines that R_1 indeed has an incipient fault. While the actual fault slope is 0.1, the estimated slope is 0.1009. The estimation using two faulty

models are shown in Figure 8 and Figure 9 respectively, and the plot for estimated value for R_1 is presented in Figure 10.

4.2 Discrete Fault in Valve 2

In this subsection, we investigate an unexpected switch fault: valve 2 closes without a command at time step $t=80s$. We only consider the flow f_6 and flow f_9 in this experiment.

Figure 11 shows the observed and estimated outputs using nominal DBN model. The fault is detected at time step $t=80.1s$, and the symbol generator reports a decrease in flow f_6 . QFI scheme generates the fault hypothesis set $F = \{(R_1^{+a}, q_{1114}), (R_1^{+i}, q_{1114}), (v1.off, q_{4114}), (v2.off, q_{4114}), (f_6^-, q_{1114})\}$. At time step $t=80.6s$, the symbol generator determines the flow f_6 to Z in mode q_{1114} and q_{4114} , because of estimated flow $\hat{f}_6 \neq 0$ and the observation $f_6 = 0$. This symbol eliminates all the parametric faults and discrete fault $v1.off$ from current trajectory. At 83.6s, the flow f_{10} shows a positive deviation (+), so the fault candidate $(v2.off, q_{4114})$ is correctly isolated. In this experiment, the real fault candidate is isolated by the QFI scheme, so the QFII scheme is not invoked.

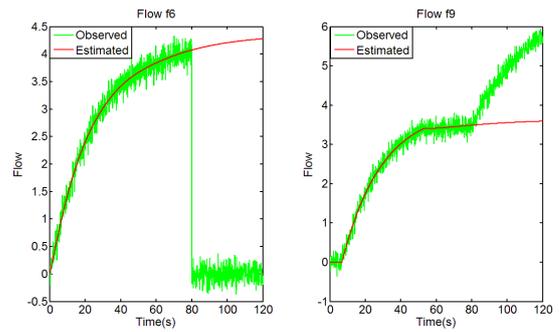


Figure 11 Observed and estimated result for nominal DBN model

We also perform several additional experiments with different fault types, fault magnitude, noise level and fault occurrence time, and obtain satisfactory results. For lack of space, we do not discuss these results in detail.

5 Conclusion

In this paper, we presented an integrated approach for online monitoring and diagnosis of incipient or abrupt parametric faults, discrete faults and sensor faults in hybrid linear systems. First of all, we adopt the HBGs to model the system, and construct the diagnosis models, i.e., the TCGs and the DBN models from the HBG model in different modes. A PF method based on the switched DBN model is employed for online monitoring of the system dynamic behavior. Once the discrete finite automaton in the HBGs detects the controlled or autonomous mode changes, HBGs will regenerate the TCGs and DBN model in new mode. These modeling approaches guarantee that the hybrid systems can be tracked correctly.

Then, we demonstrate that we can accommodate discrete faults and sensor fault models into the TCG and DBN models that represent dynamic system behavior. As a result, our model-based approach can diagnose parametric, discrete and sensor faults within the same modeling and tracking framework. Finally, QFI scheme using Hybrid TRANSCEND approach and QFII scheme by means of switched DBN-based PF approach are combined together into a common framework, which provides more discriminatory power and less computational complexity.

This work builds on approaches presented in [1][2][11][14]. [1] extends our previous work [11] from continuous systems to hybrid systems, but previous diagnosis framework could only handle abrupt parametric faults. Soon after, Daigle [2] further extended the work in [1] to capture discrete faults and sensor faults. Roychoudhury [8][14] combined a qualitative fault isolation scheme with an efficient DBN approach to diagnose both abrupt and incipient parametric faults for continuous systems. This paper proposes a comprehensive diagnosis methodology, which extends DBN-based PF observer [8][14] to track behavior of linear hybrid systems within and across mode changes, and combines qualitative fault isolation scheme in [2] with PF-based quantitative fault isolation and identification scheme in [8][14] to diagnose multiple fault types.

This method has been successfully applied to a hybrid two-tank system, and experimental results demonstrate the effectiveness of the approach. However, since the application in this paper is only a relatively simple hybrid linear system, our future work will scale up this methodology for more realistic linear and nonlinear hybrid systems. Moreover, distributed diagnostics techniques can efficiently decrease the computational complexity for complex real systems, so this is also a research direction in future [16].

Acknowledgments

This research was supported by China Scholarship Council under contract number 201306020068. The work was performed in Prof. Biswas' lab at the Institute for Software Integrated Systems (ISIS), Vanderbilt University, USA

References

- [1] Narasimhan, S. and Biswas, G. Model-based diagnosis of hybrid systems. *Systems, Man, and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 37(3): 348-361, 2007.
- [2] Daigle M J. *A qualitative event-based approach to fault diagnosis of hybrid systems*. PhD thesis, Vanderbilt University, 2008
- [3] Biswas, G., Simon, G., Mahadevan, N., Narasimhan, S., Ramirez, J. and Karsai, G. A robust method for hybrid diagnosis of complex systems. *Proceedings of the 5th Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 1125-1131, 2003
- [4] Dearden, R. and Clancy, D. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, 2002
- [5] Hofbaur, M. W. and Williams, B. C. Hybrid estimation of complex systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5): 2178-2191, 2004.
- [6] Levy, R., Arogeti, S., and Wang, D.. An integrated approach to mode tracking and diagnosis of hybrid systems. *IEEE Transactions on Industrial Electronics*, 61(4), 2024–2040, 2014.
- [7] Zhao, F., Koutsoukos, X., Haussecker, H., Reich, J. and Cheung, P. Monitoring and fault diagnosis of hybrid systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(6), 1225-1240, 2005
- [8] Roychoudhury, I., Biswas, G., Koutsoukos, X.. Comprehensive diagnosis of continuous systems using dynamic bayes nets. *Proceedings of the 19th International Workshop on Principles of Diagnosis*. 151-158, 2008
- [9] Karnopp, D. C., Margolis, D. L. and Rosenberg, R. C. *System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems*. Wiley. 2012
- [10] Roychoudhury, I., Daigle, M. J., Biswas, G. and Koutsoukos, X. Efficient simulation of hybrid systems: A hybrid bond graph approach. *Simulation*, 87(6), 467-498, 2011.
- [11] Mosterman, P. J., and Biswas, G. Diagnosis of continuous valued systems in transient operating regions. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 29(6), 554-565, 1999.
- [12] Murphy, K. P. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley. 2002
- [13] Lerner, U., Parr, R., Koller, D. and Biswas, G. Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI*, 531-537, 2000.
- [14] Roychoudhury, I. *Distributed diagnosis of continuous systems: Global diagnosis through local analysis*. PhD thesis, Vanderbilt University. 2009
- [15] Narasimhan, S. *Model-based diagnosis of hybrid systems*. PhD Dissertation, Vanderbilt University. Department on Electrical Engineering and Computer Science, August 2002.
- [16] Roychoudhury, I., Biswas, G., & Koutsoukos, X. (2009). Designing distributed diagnosers for complex continuous systems. *Automation Science and Engineering, IEEE Transactions on*, 6(2), 277-290.

ADS2 : Anytime Distributed Supervision of Distributed Systems that Face Unreliable or Costly Communication

Cédric Herpson* and Vincent Corruble and Amal El Fallah Seghrouchi

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

CNRS, UMR 7606, LIP6, F-75005, Paris, France

e-mail: firstname.lastname@lip6.fr

Abstract

The purpose of a supervision system is to detect, identify and repair any fault that may occur in the system it supervises. Nowadays industrial processes are mainly distributed, and their supervision systems are still centralized. Consequently, when communications are disrupted, it slows down or stops the supervision process. Increasing production rates make this subjection to the state of the communications no more acceptable. To allow the anytime supervision of such systems, we propose a distributed approach based on a multi-agent system where each supervision agent *autonomously* handles both diagnosis and repair on a given location. This degree of delegation, never considered in the literature nor in the industry outside of the theoretical framework, requires to overcome several difficulties : How can one agent autonomously make a diagnosis with dynamically arriving information ? How can several agents may coordinate and reach a consensus on a given diagnosis or repair with asynchronous communication ? Finally, how to allow a human to trust the decisions of such a system ? This paper develops our proposal along these three axis and evaluates ADS2 using an industrial case-study. Experiments demonstrate the relevance of our approach with an overall reduction of the supervised system down-time of 34%.

1 Introduction

Supervision systems were initially monitoring tools whose role was limited to collect and display information for their interpretation and use by the human expert. Today, the advent of complex and physically distributed systems leads to a semantic shift from supervision *tools* to supervision *systems*. Indeed, as the complexity of systems increases, humans can no longer process the flow of information arriving at each instant. The need to minimize the down-time and to improve system effectiveness requires the delegation of some of the decision-making power of the human supervisor to the supervision system. This requirement has led to the (re)birth of a research community around the notions of autonomic computing [1] and self-* systems [2]. Our work lies within this context.

Within the *Dem@tFactory*¹ project, our objective is thus to improve the supervision of an existing digitizing chain distributed over several sites (see Fig 1). Different faults – single or multiple – can occur and alter or prevent the processing of the documents (e.g. a scanner quits working, a disruption of the connection between different sites halts or corrupts a data transfer, an OCR software is poorly set and generates unexploitable results, etc.).



Figure 1: In red, the communication links between the main sites of the digitization chain of the *Dem@tFactory* project. In yellow, the links with the current (centralised) supervision system.

Centralized supervision systems are currently the most common in industry. However, they do not perform well in asynchronous contexts. Indeed, communication malfunctions between the supervision system and the geographically distributed regions of the supervised system delay the repair and do not allow to quickly return to normalcy, even though a number of malfunctions may have local predefined repair procedures available. The unbounded communication time between the supervision and the supervised system is the main reason for this problem.

To overcome this lack of robustness when facing unreliable communications and to reduce the supervised system down-time, we present in this article ADS2 : a multi-agent architecture where each supervision agent autonomously handles both diagnosis and repair on a given location. The proposed architecture is composed of three mechanisms: A *decision mechanism*, a *coordination and consistency recovery*

¹Project of the French R&D initiative Cap Digital federating 4 industrials and 3 laboratories.

ery mechanism and an *intertwining mechanism*. The *decision mechanism* tackles the dynamicity of the information available to an agent in order to make a diagnosis. The *coordination and consistency mechanism* deals with the problem of reaching a consensus between several agents on a global diagnosis (or repair) in a context of asynchronous communications. Finally, the *intertwining mechanism* address the problem of the size of the search-space in a multiple-faults context.

In this article, we first present our fault and repair model and the various assumptions made in section 2. We then describe the three mechanisms of our multi-agent architecture in section 3 to 5. We then demonstrate the viability of our proposal with experiments in section 6. Finally, we discuss related work in section 7 before concluding.

2 A Multi-Agent Architecture for the Supervision of Distributed Systems

Our architecture comes within the scope of fault-based model² approaches with spatially distributed knowledge. The supervision process is distributed among several autonomous agents having each a local view of the system to be supervised, and endowed with diagnosis and repair capabilities. The supervised system is partitioned into regions, each one is supervised by one agent. As illustrated in Fig. 2, the supervision agents (A_i) exchange information in order to establish a diagnosis and a repair consistent with the observations (O_j) they get from the various units of the supervised system (U_k). The links between the square units represent the standard workflow of the supervised system. The dashed arrows represent the fact that some elements may be reprocessed if the quality is not sufficient. The arrows between the units and the agents represent the communication links used to transmit alarms logs. The remaining links represent the communications between the supervision agents.

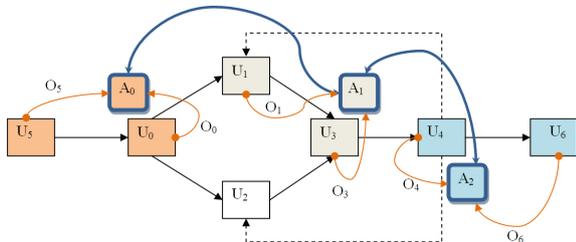


Figure 2: Example of our supervision systems deployed on a workflow.

2.1 Assumptions

We consider that communications are asynchronous and that there is no upper bound on transmission delay. We assume that the messages exchanged between supervised units may be lost or corrupted, and that some units are not supervised (e.g. unit U_2 on Fig. 2). This assumption is based on the fact that a complex industrial process commonly involves different actors that do not share their supervision information³. Moreover, we assume that the observations and the

²No model of the system's correct behaviour is available. The system can only use faults model, *a priori* known or dynamically learned from the system observation.

³subcontractors in the case of the Dem@tFactory project.

messages between agents can be lost but not corrupted. The agents are supposed to be reliable (no Byzantine behaviour). Finally, we consider that the simultaneous occurrence of different faults does not result in phenomena of masking observables.

2.2 Fault model and repair plan

Let F be the set of known faults of a system S and R be the set of existing repair plans. The signature of a fault f is a sequence of observable events generated by the occurrence of f . The set of signatures of a given fault f is $Sig(f)$.

To be able to represent any temporal dependencies, each fault is modeled as a t-temporised Petri net (Fig. 3). Each fault is supposed to be repairable, that is to say that there exists at least one partially ordered sequence of atomic repairs r_k that repairs it (a repair plan).

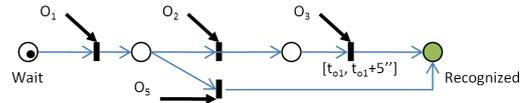


Figure 3: Let f be a fault that possesses 2 signatures. $Sig(f) = \{o_1 o_5; o_1 [o_2, o_3] [t_{o_1}, t_{o_1} + 5'']\}$. The o_i are the events observed on the supervised system. The t_{o_i} indicate the temporal constraints. Thus, $[t_{o_1}, t_{o_1} + 5'']$ constrains the sequence of observations $[o_2, o_3]$ to appear under the 5 seconds that follow the occurrence of o_1 for f to be recognized.

The supervised system is partitioned into regions rg_j . Each supervision agent is associated with one unique region and knows the models of the faults that may occur in the region it oversees. However, a fault can cover several regions. In that case, an agent only knows the part of the model that concerns its region. Its model is completed with the names of the agents responsible for the others regions. This hypothesis allow to model workflow involving different actors that do not share their data.

$$Sig(f) = \{o_1^{rg_b} o_2^{rg_b} o_3^{rg_c} o_4^{rg_c}\} \implies \begin{cases} Sig_{A^{rg_b}}(f) = o_1 o_2 A^{rg_c} \\ Sig_{A^{rg_c}}(f) = A^{rg_b} o_3 o_4 \end{cases}$$

Beside getting the models of faults, the issue of defining a global precedence relation between events that occur within the supervised system remains. Indeed, there is no common clock to the different regions. It is therefore necessary to add in each agent a stamping mechanism allowing to recreate this order relation. We will not detail here the concept of distributed clock. We consider in the following that the agents are able to recreate this partial-order relation.

2.3 Diagnosis and multiple faults

During the period of time $[t - \Delta_t, t]$, agent A_i collects a sequence of observations $seqObs_{A_i}(t, \Delta_t)$ generated by the occurrence of faults on the system. However agent A_i does not know which faults have occurred. It thus analyses $seqObs$ in order to determine the set of all faults $fp_{A_i}(t, \Delta_t)$ whose signatures partially or totally match elements of $seqObs$. A diagnosis dg is a set of faults that can explain $seqObs$. Dg is the set of all possible diagnoses of $seqObs$.

2.4 Fault cost and repair cost

Finally, each fault f (respectively each repair plan $rp(f)$) is associated with a cost of malfunction which depends of

the fault duration $Ct_{dysf}(f, t)$ (resp a cost of execution $Ct_{Ex}(rp(f))$). The cost of a diagnosis dg for the supervision system is the result of the aggregation of the respective costs of the faults that compose it. In the general case:

$$Ct_{dysf}(dg_i, t) = \text{Aggreg}_{f_j \in dg_i}(Ct_{dysf}(f_j, t)) \quad (1)$$

Similarly, the execution cost of a repair plan rp associated to a given diagnosis depends on the aggregation of the respective costs of the repairs that compose it. Thus, in the case the repair plan depends directly on the faults:

$$Ct_{Ex}(rp(dg_i)) = \text{Aggreg}'_{f_j \in dg_i}(Ct_{Ex}(rp(f_j))) \quad (2)$$

3 Agent Decision Model

We consider highly dynamic systems. Consequently, information available to an agent at a given time can be insufficient to determine with certainty which action to select. A supervision agent has thus to determine the optimal decision (D_{opt}) between the immediate triggering of the plan made under uncertainty ($Dimm_{opt}$), and a delayed action ($Ddelay_{opt}$) which lets him to wait and communicate with other supervisor agents during k time steps. This waiting time can yield information that reduces uncertainty and thereby improve decision-making. The counterpart is that the elapsed time may have a significant negative impact on the system. The *expected* potential gain in terms of accuracy must be balanced with the risks taken.

Let $Ct(x)$ the cost of an action x and $Ct_{wait}(k)$ the cost related to the extra time k before selecting a repair plan. The decision-making process of each supervision agent works as follows :

1. Observation gathering
2. Computation of the different sets of faults that can explain the current observations : Dg (set of diagnosis)
3. Determination of the immediate repair $Dimm_{opt}$ based on available information and on the constraints we chose to focus on (Most Probable Explanation, Law of parsimony, Worst case,...) and computation of its estimated cost $Ct(Dimm_{opt})$
4. A time t , an agent knows the set of the faults that may be occurring in the region it supervises $fp_{A_i}(t, \Delta_t)$. Knowing theirs signatures the agent is able to predict, for each fault of $fp_{A_i}(t, \Delta_t)$, the set of observables that can be expected to appear during the time interval $[t, t + k]$, with k an *a priori* fixed parameter. The agent uses these information to compute the waiting cost $Ct_{wait}(k)$, the expected potential gain of a delayed repair $Ddelay_{opt}$ and its associated cost $Ct(Ddelay_{opt})$.
5. Choice between the immediate repair $Dimm_{opt}$ and the delayed repair $Ddelay_{opt}$

This algorithm is executed at each time-step and by each agent when faults occur. The value k represents an upper bound delay as an agents' decision is updated each time an observation is received. We will detail in the following subsections the steps 3 and 4 relative to the determination of the immediate and delayed repair and of their respectives costs.

3.1 Immediate repair $Dimm_{opt}$

The knowledge of the different signatures of faults allows us to establish a list of potential diagnoses Dg . We sort

these explanations according to available information and to the constraints we chose to focus on (e.g the most probable explanation). After this step , the first element of Dg is the diagnosis considered as the most relevant at the current time. It is then necessary to estimate its cost.

The cost of the immediate repair $Ct(Drep_{opt})$ must take into account the execution cost of the repair plan associated to the diagnosis retained (Ct_{Ex} , equation 2), as well as a cost representative of the potential error relative to this decision, Ct_{Err} . Indeed, if the only cost considered is the one of the execution of the repair plan, the final decision (step 5) will always favour an immediate action compared with a delayed one due to the additional waiting cost of the delayed action.

$$Ct(rp(dg_i)) = Ct_{Ex}(rp(dg_i)) + Ct_{Err}(dg_i | Dg \setminus \{dg_i\}) \quad (3)$$

The computation of the error cost Ct_{Err} relies on the fact that we assume that the good diagnosis – and so the good repair – belongs to the sorted list Dg of the potential diagnoses. Thus, in case of misdiagnosis when selecting the first diagnosis dg_1 of Dg , the system will lose a time equal to the execution time of the first repair plan ($Ct_{ExecTime}(rp(dg_1))$) which will be supplemented by the execution cost of the newly chosen repair plan ($Ct_{Ex}(rp(dg_2))$) associated to the 2nd diagnosis of Dg . As this second choice may also turn out to be an error, we define Ct_{Err} recursively on Dg . Thus:

$$\begin{cases} Ct_{Err}(dg_1 | []) = 0 // Dg \text{ is empty, the diagnosis is correct.} \\ Ct_{Err}(dg_1 | Dg \setminus \{dg_1\}) = P(dg_1 | Dg \setminus \{dg_1\}) \times \\ \left[\begin{array}{l} Ct_{ExecTime}(rp(dg_1)) + Ct_{Ex}(rp(dg_2)) \\ + Ct_{Err}(dg_2 | Dg \setminus \{dg_1, dg_2\}) \end{array} \right] \end{cases}$$

with $P(dg_1 | Dg \setminus \{dg_1\})$, the probability that choosing dg_1 as the final diagnosis is an error.

3.2 Delayed repair $Ddelay_{opt}$

A time t , an agent knows the set of the faults that may be occurring in the region it supervises $fp_{A_i}(t, \Delta_t)$. The different faults models are represented using t-temporised Petri-nets (Fig. 3 page 2). The agent is thus able to predict, for each fault of $fp_{A_i}(t, \Delta_t)$, the set of observables that can be expected to appear during the time interval $[t, t + k]$, with k an *a priori* fixed parameter. Note that the agent uses the current transmission duration (computed over the interval $[t - \Delta_t, t]$) to determine the set of potential observations.

From this information, the agent builds the tree representing the set of all possibles futures working towards the current time plus k units of time, $Arb_{A_i}^{possibles}(k)$. Each node of the tree is associated with a set of observations and represent one possible future (Fig. 4 below). The agent then computes, for each node of the tree, the set of diagnoses that explain this future (Dg').

The agent can then compute, for each possible future, the immediate decision considered as optimal. At time t , the determination of the delayed decision with horizon k ($Ddelay_{opt}$) involves choosing between the various possibles situations. This choice is realised by sorting the first elements of each Dg' of the tree of the possibles futures with each other using the same criterion than the

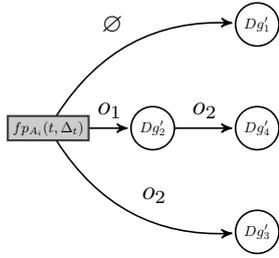


Figure 4: Illustrative example of a tree of the possible futures.

one used to identify the immediate repair in the sub-section 3.1.

Once the delayed decision is identified, its cost $Ct(Ddelay_{opt})$ is established using Equation (3). We then have to add to this cost the waiting cost Ct_{wait} . This waiting cost represents the consequences of the faults on the supervised system during the time where no action was triggered. The computation of the waiting cost depends on the respective costs of the malfunctions associated to the remaining diagnoses and of the elapsed time.

$$Ct_{wait}(k) = Aggreg_{dg_i \in Dg}(Ct_{dysf}(dg_i, k)) \quad (4)$$

4 Distributed Supervision and System Consistency

In the previous section, we addressed the problem of one agent making a decision. However, as each agent has a local view of the system, a decision about a diagnosis and/or a repair frequently requires information and knowledge from other supervision agents. It therefore becomes necessary to reach a consensus on the decision to make.

However, distributed supervision works in a context of asynchronous and unbounded communication. Within these constraints the theorem of Fisher-Lynch-Paterson [3] states the impossibility of guaranteeing the achievement of a consensus between different components.

To circumvent this difficulty, the literature on supervision frequently introduce hypotheses on the quality of the communications. As our work tends to work under real-life hypotheses, we do not make any regarding the (un)reliability of the communication. We discuss in this section the use the multi-Paxos algorithm [4] to reach a consensus when the state of the communication allows it, and propose a consistency mechanism to restore a common view of the system by the agents after a unilateral decision taken by some of them.

4.1 Consensus algorithm

In the general case, establishing a consensus must meet the following properties : (Agreement) All correct processes decide the same value. (Integrity) Every process decides at most once. (Validity) Each value determined belongs to the set of proposed values. (Termination) Every correct process eventually decides in a finite time. However, in the context of Fisher-Lynch-Paterson theorem, the supervision system can only offer a guarantee of “best-effort”. i.e, to assure that the consensus can be reached, but only if the system is stable on a sufficiently important period of time [5].

The multi-Paxos algorithm, initially developed for reaching an agreement in a network of unreliable processors, falls into this category. The interesting aspect of this algorithm is that it was designed to resist to halt failures - with recovery possibility - of a number of processes, including the coordinator. Its very low number of assumptions makes it operational in an environment with unreliable communications. These properties make it particularly suited to multiagent systems. Using the multi-Paxos, each agent is able to initiate, integrate or leave a coalition.

The fact that there is no upper bound on the time needed to reach a consensus will inexorably lead to some unilateral decision-making by agents or agent groups in case of communication disruption. This feature of our system guarantees the avoidance of deadlock situations when communications are too unstable to let the agents reach a consensus. However, this ability requires the introduction of an algorithm to restore a consistent view of the system state by all agents.

4.2 System consistency

Algorithm 1 works in the manner of producer-consumer with the decision-making process introduced in section 3. The two algorithms share, within an agent, a common inconsistency queue F_{inc} . When a coalition is left by at least one agent before reaching a consensus (due to a communication breakdown or to an agent’s decision), the members of the coalition store their respective decision-making context (the current sequence of observables, the set of considered explanations and the list of agents which belong to the coalition) into their own potential inconsistency queue F_{inc} .

The consistency maintenance algorithm is available within each agent as a behaviour, it continuously observes the state of the queue F_{inc} . When an entry is added to F_{inc} , the algorithm is automatically triggered.

Algorithm 1 Check consistency

Require: Pattern observer on F_{inc}

```

1: if  $F_{inc} \neq \emptyset$  then
2:   Try to contact  $F_{inc}.getFirst().getCoalition()$ 
3:   if contact successful then
4:     Send  $F_{inc}.getFirst()$ 
5:     Receive other agents decision context
6:     Make pairing between local decision context and others.
7:     if pairing is ok then
8:        $F_{inc}.removeFirst()$ 
9:     else
10:      start new paxos instance
11:    end if
12:  end if
13: end if
    
```

This algorithm lets each agent find a match between its actions and those selected by the other members of the coalition. Thus, in case of faults due to past inconsistency decisions taken by the agents, they are able to trigger a sequential diagnosis and to discriminate initials disturbances from the consequences of their decisions.

When the potential inconsistency queue of an agent contains one element, the agent tries to resolve it. The agent tries to contact each of the agents of the coalition concerned with this potential inconsistency F_{inc} . If these agents are able to communicate (the communications are restored),

they will exchange their respective decision-making contexts. By comparing them, they will be able to determine whether the decisions made locally by the different groups of agents are consistent with one another. If this is the case (the faults are repaired, the system is in a stable state and correct), each agent removes P_{inc} of the queue. Otherwise, the subset of agents involved initiates a new coalition in order to resynchronize their respective views of the system and make a decision consistent at the system's scale. If communications are too unstable (or too costly), this consensus will not be reached, which results in adding a new entry in F_{inc} .

Restoring the consistency of the system state as it is perceived by the supervision agents is again relying on the stability of the communication links for a sufficient amount of time.

5 Intertwining Diagnosis and Repair Stages

In the previous sections, we endowed the supervision agents with decision-making and coordination mechanisms. These abilities allow the agents to dynamically adapt their behaviours to the current state of the communications and of the supervised system. In case of uncertainty regarding the decision to make, the agents are thus able to explore the solution space, collectively as well as individually. However, the large size of this set remains a problem. Indeed, it is both a source of misdiagnosis in case of local decision-making and the cause of a large number of supervision messages when a consensus must be reached. In order to reduce the complexity of the decision process, we address in this section the question of obtaining the minimal set of diagnoses and of associated repair plans. To this aim, we discuss the idea of intertwining the diagnosis and repair stages.

This idea has been introduced by Cordier *et al* [6] on the formalization of self-healing systems [7]. Several failures may indeed have the same signature without calling into question the reparability of the system, all that is needed is that a repair be common to all of the faults involved (notion of *macrofault*).

However, restricted to the single-fault context, this formal model defines the diagnosability and reparability of a system as static properties that can be computed offline. This is not the case in the multiple-faults context. Indeed, the appearance of faults can prevent the triggering of a repair associated to another fault currently occurring in the system, and the possible situations are endless. Being able to represent this kind of interference is essential to our work. This led us to introduce context-dependent notions of diagnosability and reparability.

Definition 1 (Conditionnal Diagnosability).

$$\begin{aligned} \text{Diagnosable}(f_i, t) &\iff \forall x \in C^D(f_i), x \notin ss(t) \\ &\iff C_t^D(f_i) = \emptyset \end{aligned}$$

A fault f is diagnosable at time t if none of the faults that may prevent its diagnosability (e.g if they share the same signature) is appearing in the system at this instant. This set of faults is the conflict set in diagnosis of the fault f (denoted by $C_t^D(f)$). Following the same reasoning, we can define $C_t^R(f)$ as the conflict set in repair of the fault f .

Finally, the uncertainty regarding the faults that are currently occurring on the supervised system, may conduct the supervision agents' to "believe" the occurrence of

faults which are not real and which prevent the repair of the system. We call these situations virtual deadlocks. To disambiguate these situations, we added a relationship of innocuousness I to these definitions. Thus, for a fault f , a repair plan $r(f)$, its repair conflict set $C^R(f)$ and considering the current state of the system, I returns the set of sets of faults belonging to C_t^R on which the execution of repair plan $r(f)$ leaves the system unchanged. The result of this innocuousness relation is the set of disambiguation under repair, denoted by $D^R(f)$. Taking into account all this information, we are then able to propose an algorithm to plan the order of the repairs and to resolve some conflicts. We illustrate how it operates below :

Example : Let $F = \{f_1, f_2, f_3\}$ with $Sig(f_1) = \{a\}$, $Sig(f_2) = \{a\}$ and $Sig(f_3) = \{b\}$. $Rp(f_1) = \{r_1\}$, $Rp(f_2) = \{r_2\}$ and $Rp(f_3) = \{r_3\}$. Moreover, we know that $C^R(f_1) = \{\{f_3\}\}$ and that $D^R(f_2) = \{\{f_2\}, \{f_1\}\}$. As the signatures of f_1 and f_2 are identical, it follows that $C^D(f_1) = \{\{f_2\}\}$ and $C^D(f_2) = \{\{f_1\}\}$. We assume that an agent detects the observables $a \wedge b$.

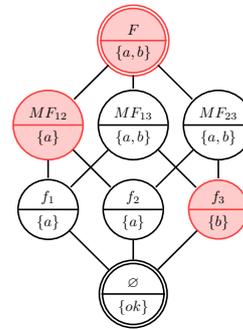


Figure 5: Diagnosis state for the agent : $dg_1 = \{f_2, f_3\}$, $dg_2 = \{f_1, f_3\}$

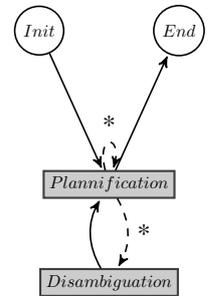


Figure 6: Operation scheme of the active repair algorithm

In Fig. 6, the initialization of the algorithm determines for each potential fault f_i all repair conflicts existing at the current time $C_t^R(f_i)$. The planning phase recursively builds the repairing order from C_t^D and C_t^R adding the faults whose conflict sets are empty, and then updates the remaining ones. At the end of this phase, if some faults remain, they potentially are in a deadlock. In our example, the agent has to choose between $\{f_2, f_3\}$ and $\{f_1, f_3\}$. As highlighted in Fig.5, the agent can repair f_3 but is unable to make a distinction between f_1 and f_2 (we assume that this conflict is virtual and that only one of these faults is occurring).

The disambiguation phase then attempts - from the proven innocuousness of some repairs in the current context - to solve these conflicts. If one of them is solved, the planning phase is retried after updating the conflict sets. If the disambiguation does not work, it means that the agent does not have, at the current time, enough information to solve the problem. The decision-making process previously introduced in section 3 is then triggered. In our example, repair r_2 is selected. If the system returns to normalcy, then both diagnosis and repair phases end. If not, the previous action guarantee the occurrence of f_1 , and the associated repair plan is executed.

6 Experimental Evaluation

To evaluate our approach we developed a simulator for distributed systems. Based on the JADE multi-agent platform [8], our environment allows us to model both physical units and communications links, and to simulate the occurrence of failures in it. For a given simulation, a list of faults is associated with each site and each communication link (A communication link can increase its transmission time, a unit may stop working properly,...). Each fault is associated with one or more trigger conditions: a date and/or the occurrence of another failure. This allows us to simulate cascading faults. Our supervision system is deployed on this simulator. When running the simulation, some faults trigger the sending of an alarm message to the agent responsible for the site where they appear. The agents will try to determine the appropriate behaviour from these messages.

Our agent decisional model is generic. It can be instantiated with various criteria (the most probable explanation, the worst case hypothesis, etc.) depending on available information and on the constraints we choose to focus on. In the *Dem@tFactory* project, it appeared essential to consider the utility of a decision based on the cost of the occurrence of a given set of faults rather than on its occurrence probability. This reasoning led us to favor a robust decision criterion. The decisions taken by the agents will therefore rely on the worst case hypothesis.

The upper bound k which is the horizon considered by the agents of *ADS2* for the computation of the delayed decision is set to 15 units of time. Moreover, we assume in these experiments that the respective costs of the faults that compose a diagnosis are additive. Finally, in order to have benchmarks for the evaluation of the principles underlying our architecture (*ADS2*), we also implemented a centralized supervision systems (*SC*) where all observables are transmitted to a single supervision agent.

6.1 Experimental setup

Our goal is to study the behaviour of the supervision system facing an industrial case-study.

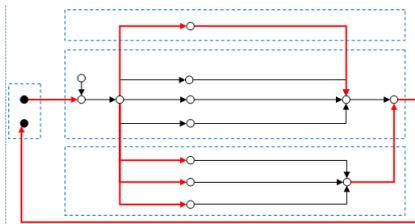


Figure 7: Workflow of the digitizing chain of the *Dem@tfactory* project.

Fig. 7 represents the digitizing chain of the *Dem@tFactory* used for the experiments. Each dotted rectangle corresponds to a factory situated on a given geographical location (2 in France, 1 in Madagascar and 1 in Mauritius). The circles correspond to the various process required to the digitization. The inter-rectangle links correspond to communications between the different factories, and the intra-rectangle one to local communications. All these components are modeled in the simulator and can engender the occurrence of faults.

We fixed *a priori* the locations and responsibilities (the regions) of the supervision agents according to the geographical location of the units that compose the supervised process. We used a dataset provided by our industrial partners (8 GB of data corresponding to 48 hours of logs) to extract nineteen different faults models. From these data, we determined that the probability of occurrence of n faults per unit of time follows a Poisson distribution with parameter $\lambda = 0.043$. Finally, using information gathered from our partners, we were able to estimate the costs of the faults over time (constant, logarithmic,...) and the associated repair costs.

6.2 Experiments

Our experiments study the behaviour of the supervisions systems when varying the (heterogeneous) transmission cost. We used a random generator to affect a transmission time (between 0 to 30 units of time) to each transmission link for each time unit of the simulation. We arbitrarily set at 10% the probability of a link to get a transmission time greater than 1. In order to obtain a baseline, we first performed different simulations with homogeneous transmission costs.

The performance evaluation is based on three criteria: (1) The average response time to a malfunction. (2) The average number of supervision messages exchanged. (3) The average total cost of repairs made during the experiment.

Figs. 8(a) and 8(b) present the evolution of the behaviour of supervision systems *ADS2* and *SC* for the two first criteria in the case of homogeneous (Ho) and heterogeneous (He) communication links. The vertical bar at $t=15ut$ is the horizon considered by the agents of *ADS2* for the computation of the delayed decision.

Fig. 8(a) shows that our architecture is very robust, allowing the supervised system to rapidly recover from failures. The response time of *ADS2* (Ho and He) progressively stabilized around 15ut, when the response time of *SC* increases over time and becomes higher than *ADS2*. This is due to the fact that the agents of *ADS2* can decide to act without waiting for the reception of all the messages that come from the units of the supervised system.

We can observe an increase of the average response time of *ADS2*(Ho) when the transmission delay is close to 15 ut. This is due to the parameter k of our algorithm, *a priori* fixed to 15 ut. This parameter defines the agent's horizon for the computation of the delayed decision. When the transmission time becomes greater or equal to k , an agent no longer sees interest in waiting or trying to exchange information with other agents of *ADS2*; so it decides to act despite the risk of making a mistake. The impact of parameter k is less important on *ADS2*(He). Indeed, as the communication links are in this case heterogeneous, a supervision agent is still able to exchange information with some of the other agents. This leads to a better response time for *ADS2*(He) than for *ADS2*(Ho). This behaviour is clearly highlighted in figure 8(b). The number of messages exchanged by the agents of *ADS2* drop with the increase of the transmission delay. We see a sudden drop of this number when the transmission delay becomes greater than 15 ut for *ADS2*(Ho), confirming the local decision-making of the agents.

Fig. 8(c) shows that the decisions of *ADS2*(He) agents generate a limited repair extra-cost in comparison to the cen-

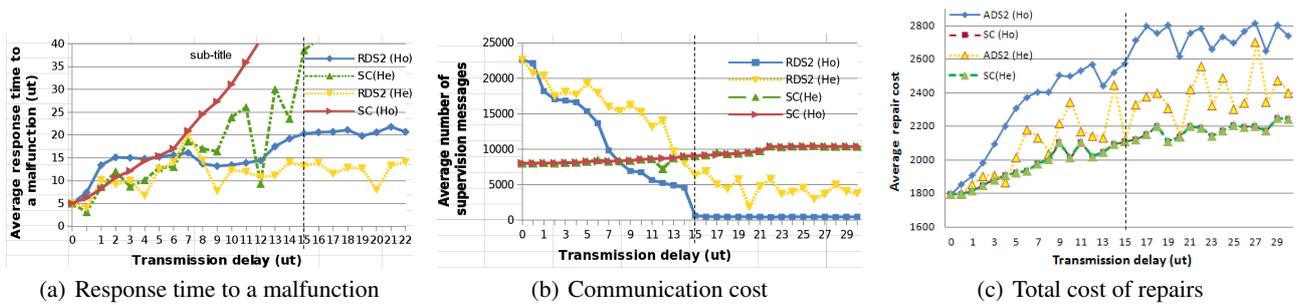


Figure 8: Experimental results. The curves corresponding to homogeneous/heterogeneous communication links are respectively marked with (Ho) and (He). The x-axis is the transmission delay. The y-axis correspond for each figure to one of the evaluation criteria.

tralised approach (9%). With the Dem@tFactory fault models, the overall gain regarding the supervised system downtime reach 34%.

Considering the reactivity of *ADS2* and the limited repair extra-cost it generates, the communication extra-cost for low transmission delays can be considered as an acceptable consequences compared with a total-absence of supervision (*SC*).

Our next set of experiments evaluates the impact of the intertwining of the diagnosis and repair phases on the performances of the supervision system. In order to evaluate the impact of this behaviour for a supervision agent, we initially activated this capability in only one agent of *ADS2*. We realised 100 simulations. The 10 first simulations are performed with a number a simultaneous faults restricted to 1. Then the number is gradually incremented every 10 simulations to reach 10 simultaneous faults. For each simulation, the number of potential diagnoses considered by the agent is saved at 5 specific time steps. In order to obtain a baseline, we performed the same 100 simulation with the intertwining behaviour deactivated. Fig. 9(a) shows that the interleaving of the diagnosis and repair process does lead to a reduction of the diagnosis search space of an agent between 10 to 20% for the set of faults of the *Dem@tFactory* project.

Fig. 9(b) shows that the reduction of the number of potential explanations of each agent is of an extend sufficient to allow the agent to reduce the number of supervision messages. The everage response time to a malfunction is not significantly improved (Fig. 9(c)) but the repair extra-cost fall from +9% (*ADS2*) to 7.2% (*ADS2+*) (with $p < 0.05$).

7 Related Work

The supervision of a system consists of four steps: Detection, Isolation, Identification and Repair. The literature aggregates the 3 first steps under the name FDI (Fault Detection and Isolation) [9]. Although several approaches for the distributed supervision of distributed systems have been proposed in the literature, whether work is from the diagnosis and control communities [10; 11] or from the multi-agent domain [12; 13], they do not cover the repair phase.

In the work from areas related to distributed systems, emphasis is placed on the distribution of available knowledge on the status and behaviour of the supervised system. Fröhlich *et al* [14] and Roos *et al* [13] have addressed the question of the ability of a set of agents to determine an overall diagnosis according to the shape of this distribution.

They have shown that to obtain a minimum overall diagnosis is NP-hard in the case of spatially distributed information and that the complexity of obtaining the diagnosis is independent of the communications costs engendered during its establishment [13].

Given these theoretical results, reducing the space of potential solutions is generally based on a hierarchical structure of the diagnosis agents [12] and on the choice of not returning to previously excluded explanation. Though the no back-track of past decisions guarantees convergence and termination of the algorithm, it is a source of diagnosis errors in an asynchronous environment. The *best-effort* approach we chose allows us to reduce these diagnosis errors, and the termination of the diagnosis algorithm is guaranteed through the anytime decision-making process of our agents.

To our knowledge, the work of Nejdil *et al* [15] is the only one that addresses the distribution of both the diagnosis and repair phases. However, placed at a relatively abstract level of analysis, this work makes the assumptions that communication links are reliable and that messages can be exchanged between agents at no cost. In a real situation these hypotheses are too restrictive. Indeed, to not consider the communication state may render the supervision system ineffective or inoperable. Our proposal does not make such assumptions.

The problem of online decision-making under uncertainty is the central point of the work by Horvitz [16], Hansen et Zilberstein [17] on the control of anytime algorithms. Indeed, they propose a formal framework to dynamically determine the time to stop a calculation taking into account the quality of the current solution and the cost of the algorithm computation.

The first distinction between these work and ours is that in their work the authors determine when to stop the computation based on the distance between the current solution and the optimal one. This requires knowing the optimal solution (or an estimation) and to be able to dynamically determine this distance. In our work, talking about the quality of a solution (i.e a diagnosis) is meaningless insofar as a diagnosis is right or wrong, and its “value” is only known a posteriori. The second point of divergence is that we try to select a candidate (a diagnosis or a repair) among a set of potential solutions. The complexity of the task is therefore increased.

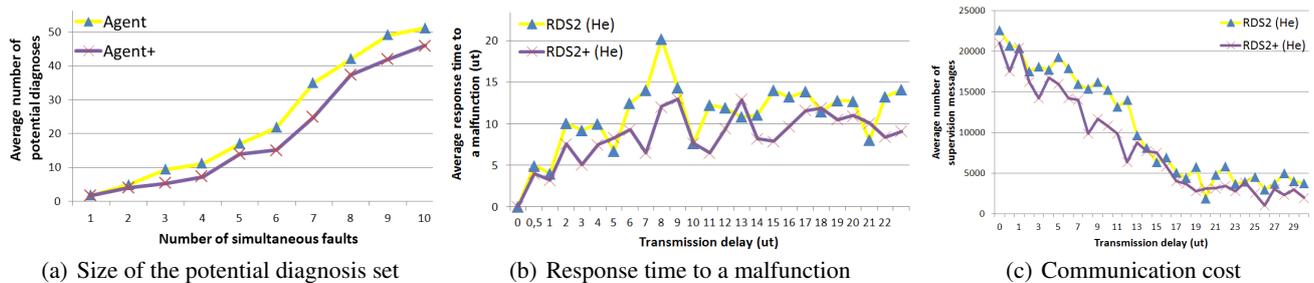


Figure 9: Simulation results when integrating the interleaving of the diagnosis and repair steps. The symbol “+” is associated to the components which integrate this additional mechanism.

8 Conclusions

We presented the first anytime multi-agent architecture for the supervision of distributed systems that is able to dynamically adapt its behaviour to the current state of the supervised system. In particular, the decision model allows each supervision agent to find a balance between a quick local diagnosis and repair under uncertainty, and a delayed, systemic one, based on the respective costs of misdiagnosis and communication. The distributed consistency algorithm allows each agent to form a coalition to reduce its uncertainty or to restore a consistent view of the system state in case some had to act locally with incomplete information at an earlier stage. Moreover, the intertwining of the diagnosis and the repair phases allows an efficient reduction of the diagnosis search-space. The overall reduction of 34% of the Dem@tFactory system down-time associated with a repair extra-cost of 7.2% demonstrate that ADS2 is able to efficiently supervise complex systems under real-life assumptions.

A fully autonomous supervision system is presently not realistic in an industrial context as Humans want to keep control on what they perceive as critical decisions. ADS2 represents what we see as an acceptable trade-off as the definition of its autonomy degree can be easily accomplished. Thus ADS2 organizes the set of known faults and repairs in several subclasses : the ones whose repair plan can be triggered automatically, and those whose final repair decision rests with a human supervisor. The risk aversion of the users defines the size of these two respective sets. If the confidence in the efficiency of the autonomous supervision of complex and distributed systems is not common today, we believe that the work presented herein provides a step towards this goal.

References

- [1] J. O. Kephart and D. M. Chess. The vision of autonomous computing. *Computer*, 36(1):41–50, 2003.
- [2] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):1–42, 2009.
- [3] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [4] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32:18–25, 2001.
- [5] R. De Prisco, B. Lampsom, and N. Lynch. Revisiting the paxos algorithm. *Distributed Algorithms*, pages 111–125, 2000.
- [6] Marie-Odile Cordier, Y. Pencolé, L. Travé-Massuyes, and T. Vidal. Self-healability = diagnosability + repairability. In *The 18th International Workshop on Principles of Diagnosis*, volume 7, pages 251–258. Citeseer, 2007.
- [7] Philip Koopman. Elements of the self-healing system problem space. In *ICSEWADS03*, 2003.
- [8] K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak, and M. Paprzycki. Efficiency of JADE agent platform. volume 13, pages 159–172. IOS Press, 2005.
- [9] Giovanni Betta and Antonio Pietrosanto. Instrument fault detection and isolation: state of the art and new research trends. volume 49, pages 100–107, 1998.
- [10] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen. Failure diagnosis of dynamic systems: an approach based on discrete event systems. In *Proc. American Control Conference*, volume 3, pages 2058–2071, June 25–27, 2001.
- [11] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.
- [12] H. Wörn, T. Längle, M. Albert, A. Kazi, A. Brighenti, S. Revuelta Seijo, C. Senior, M A S. Bobi, and JV. Collado. Diamond: distributed multi-agent architecture for monitoring and diagnosis. *Production Planning & Control*, 15:189–200, 2004.
- [13] Nico Roos, Annette ten Teije, André Bos, and Cees Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, page 7. ACM, 2003.
- [14] Peter Fröhlich and Wolfgang Nejdl. Resolving conflicts in distributed diagnosis. In *ECAI Workshop on Modelling Conflicts in AI*, 1996.
- [15] W. Nejdl and M. Werner. Distributed intelligent agents for control, diagnosis and repair. *RWTH Aachen, Informatik, Tech. Rep.*, 1994.
- [16] E. Horvitz and G. Rutledge. Time-dependent utility and action under uncertainty. pages 151–158, 1991.
- [17] E.A. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126:139–157, 2001.

Data Driven Modeling for System-Level Condition Monitoring on Wind Power Plants

Jens Eickmeyer¹, Peng Li², Omid Givehchi², Florian Pethig¹ and Oliver Niggemann^{1,2}

¹Fraunhofer Application Center Industrial Automation IOSB-INA

e-mail: {jens.eickmeyer, florian.pethig, oliver.niggemann}@iosb-ina.fraunhofer.de

² inIT - Institute Industrial IT

e-mail: {peng.li, omid.givehchi, oliver.niggemann}@hs-owl.de

Abstract

The wind energy sector grew continuously in the last 17 years, which illustrates the potential of wind energy as an alternative to fossil fuel. In parallel to physical architecture evolution, the scheduling of maintenance optimizes the yield of wind power plants. This paper presents an innovative approach to condition monitoring of wind power plants, that provides a system-level anomaly detection for preventive maintenance. At first a data-driven modeling algorithm is presented which utilizes generic machine learning methods. This approach allows to automatically model a system in order to monitor the behaviors of a wind power plant. Additionally, this automatically learned model is used as a basis for the second algorithm presented in this work, which detects anomalous system behavior and can alarm its operator. Both presented algorithms are used in an overall solution that neither rely on specialized wind power plant architectures nor requires specific types of sensors. To evaluate the developed algorithms, two well-known clustering methods are used as a reference.

1 Introduction

According to a wind market statistic by the GWEC (Global Wind Energy Council) [1], the global wind power capacity grew continuously for the last 17 years. In 2014, the global wind industry had a 44 % rise of annual installations and the worldwide total installed capacity accumulated to 369553 megawatt at the end of 2014. In Europe, renewable energy from wind power plants (WPP) covers up to 11% of the energy demand [2]. With this rapid continuous growth, the wind power is considered as one of the most competitive alternative to fossil fuels.

In a case study, Nilsson [3] denotes an unscheduled downtime with 1000 € per man-hour, with costs of up to 300000 € for replacements. This does not take into account the reduced yield through production loss. Therefore, the objective of maintenance is to reduce WPPs downtimes and provide high availability and reliability.

High availability is currently achieved by two different strategies. On the one hand, maintenance is planned as regular time-interval based on the manufacturer's data of specific WPP parts. This is performed in order to prevent wearout

failure. On the other hand, there is the strategy of corrective maintenance, which reacts to occurred failures. Both strategies need time for actual maintenance, which lead to non productive downtimes. Especially, when considering offshore WPP, these downtimes produce high costs.

To reduce these downtimes a precise proactive scheduling of maintenance task is needed. This is achieved through condition monitoring (CM) systems [4]. Those systems try to reason about the inherent system states such as wear, although these conditions cannot be measured directly, but the growing amount of sensors in modern WPP enable an adequate description of the machines state. To make use of this, CM systems need a model of the WPP, which describes the system behavior based on observed data.

Existing CM solutions for WPP rely on specific sensors and are specialized to monitor single parts of the system. The gearbox [5], the bearing [6], the generator [7] or the blades [8] have been monitored in order to perform proactive maintenance. Here, specific sensors are needed as a requirement for these specialized methods.

This article presents a system-level solution which handles heterogeneous WPP architecture regardless of installed sensor types. Also, an algorithm for modeling a WPP on system level and another algorithm for anomaly detection are stated. To achieve this, three challenges are tackled and their solutions are presented:

- I. *Logging data* from available sensors of a WPP, using existing infrastructure independent of the architecture. Additionally, the opportunity must be given to add new sensors and sensor types on demand.
- II. *Automatic modeling* of a WPP, by combining existing and generic data-driven methods. Such a model must be able to learn the complex sensor interdependencies without extra manual effort.
- III. *Anomaly detection* for a WPP regardless of its kind of architectures, especially with no assumptions on available types of sensors.

The article is structured as follows. Section 2 deals with state of the art technology in WPP CM. Hardware and data acquisition for the presented solution are specified in section 3, here point I is the central issue. Data-driven models realizing point II and the analyzed machine learning approaches are the purpose of section 4. Anomaly detection and its general approach, according point III is stated in section 4.2. The results of an evaluation of the presented methods is content of section 5. Finally, this paper concludes in section 6 and describes future aims of the presented work.

2 Related Work

The core task of a CM system is anomaly detection. As stated in [9], the models used for anomaly detection of complex systems should be learned automatically and data-driven approaches to learning such models should be moved into the research focus.

A wide range of data-driven algorithms that deal with modeling the system behavior for anomaly detection are available in the literature.

Because of its simplicity in processing huge amounts of data, the Principal Component Analysis (PCA) based algorithms are widely applied in the condition monitoring of WPP [10][11].

As one of the classic density based clustering method, DBSCAN shows its advantages over the statistical method on anomaly detection in temperature data [12].

Piero and Enrico proposed a spectral clustering based method for fault diagnosis where fuzzy logic is used to measure the similarity and the fuzzy C-Means is used for clustering the data [13].

Due to the high complexity of a WPP and its harsh working environment, the modeling of WPPs on system level is very challenging. Most data-driven solutions to WPP condition monitoring concentrate on the errors of one particular component (in component level) [4]. These methods are designed to detect specific faults (e.g. fault in gearbox, generator).

The application of such methods is available in different studies. In [6], a shock pulse method is adapted for bearing monitoring. A multi-agent system is developed in [5] for condition monitoring of the wind turbine gearbox and oil temperature. In [8], the ultrasonic and radiographic techniques are used for non-destructive testing of the WPP blades. Using these methods can prevent the WPP breakdowns caused by the particular faults. For enhancing the availability and the reliability of the whole WPP, a method for monitoring the WPP on system-level is desired.

In this work, a PCA-based algorithm for condition monitoring of WPP is presented. This approach is aimed to model a WPP on system-level in order to perform automatic anomaly detection. As a comparison, DBSCAN and spectral clustering are utilized for the same purpose. To the best of our knowledge, no application of either DBSCAN or spectral clustering in condition monitoring of WPP exists.

3 Data Acquisition Solution

A WPP includes different types of sensors, actuators and controllers installed to monitor and control the different devices and components as shown in Figure 1. To monitor the condition of a WPP, it is necessary to collect process data from its sensors and components accurately and continuously feed this data to the diagnosis algorithms. To maximize accuracy, data should be acquired directly from the sensors and components or via the existing communication systems. Despite the fact that IEC 61400-25 [14] addresses a variety of standards and protocols in WPP, lots of proprietary solutions exist today. A general approach to accurate data acquisition in an uniform way implies protocol adapters or data loggers (DL) to connect the diagnosis framework. This is done not only for IEC 61400-25 conformant WPP, but also for proprietary ones using e.g. the MODBUS protocol or a direct connection via general-purpose input/output (GPIO) [15]. Also the data logger should model data based

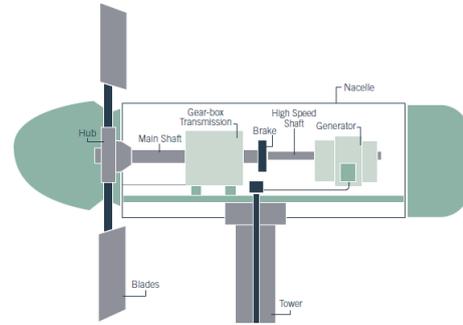


Figure 1: Diagram showing the inside of a nacelle and main components [4]

on generic industrial standards (IEC 61400-25) and transfer them to a database for storage and processing. Such a data logger meets point I (see section 1). In addition, the timestamp of the data should be synchronized between data loggers, database and application accurately.

In this work we followed a three layer architecture for data acquisition as shown in Figure 2 which covers all of the CM system components. In layer 1, the physical machine components are connected to the data logger hardware using different industrial connections and protocols e.g. digital GPIO, RS485, MODBUS, etc. The data loggers are time synchronized using global positioning system (GPS) or network time protocol (NTP) time references via an embedded time client running in the data logger. Collected sensor data is attached to their accurate timestamps by an embedded OPC UA server inside the data logger. The sensor data is categorized based on an OPC unified architecture (OPC UA) data model (e.g. conformant to IEC 61400-25) for a standalone WPP.

The communication between data logger, OPC UA server and layer 2 is realized with a secure general packet radio service network (GPRS) or a virtual private network (VPN), while it can be accessed for widely distributed WPPs in different geographical locations. The layer 2 comprises a middleware to collect and host the sensor data coming from distributed data loggers. It mainly covers a database with support of historical data and also an OPC UA server aggregating the data incoming from distributed WPP data loggers and pushes them to the database using an OPC UA database wrapper. As shown in Figure 2, the main component of layer 3 is an analysis engine. This engine applies algorithms on the database. Based on the learned machine models an output about the machines condition is presented to the operator by a human machine interface (HMI).

4 Modeling Solution

The main idea of the presented solution is to automatically learn a model of normal system behavior from the observed data using data-driven methods. Classical manual modeling utilizes expert process knowledge to build a simulation model as a reference for anomaly detection. But a process such as a WPP contains numerous continuous sensor values, which make it difficult to model the system manually. Therefore, as first step of the solution a model is learned from a set of data. The second step utilizes this model as reference to perform anomaly detection. This section considers these two steps.

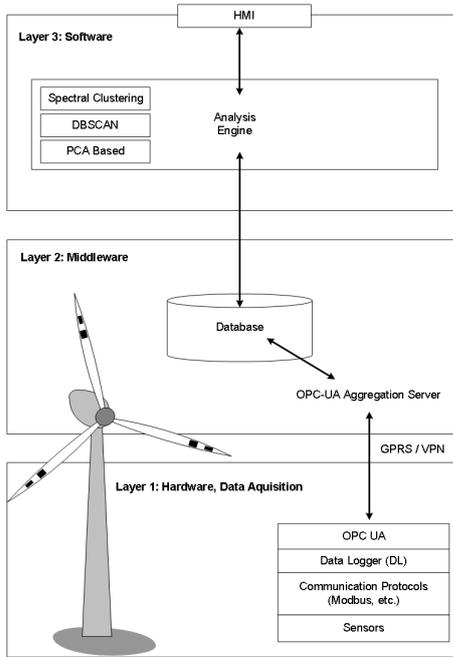


Figure 2: Architecture overview of the presented system-level condition monitoring solution for a WPP

4.1 Step 1: Data-Driven Modeling

In order to automatically compute a system model, the presented solution uses generic methods to analyze training data and aim for process knowledge. These methods from the field of machine learning reduce effort of time for generating a system model caused by the complex sensor interdependencies. Additionally, a WPP is influenced by seasonal components and a normal state of work cannot be declared as precise as for a machine that works in a homogeneous environment of a factory. This meets the requirement in point II (see section 1). In this solution, step 2 detects anomalies as deviation between an observation and the learned reference model of the system, this is described in section 4.2.

Common strategies for data-driven modeling are supervised and unsupervised learning methods. Supervised methods such as Multilayer Perceptron, Support Vector Machines or Naive Bayes Classifier (see [16] for more information) can be used to directly classify data according to learned hyperplanes in the data space. To be reliable, those methods need a-priori knowledge from labeled data of possible faults and the normal state. Gathering those precise data for a continuous production system like a WPP is hard to realize, as faults are rare and environmental conditions increase the number of possible faults dramatically.

In comparison, unsupervised learning methods (e.g. Clustering, Self Organizing Maps) seek to model data without any a-priori knowledge. Therefore, they are able to extract knowledge from unlabeled data sets and generate a model out of this knowledge. In this article, two types of unsupervised learning methods are investigated to model a WPP using unlabeled data. The PCA based modeling is compared against cluster based modeling methods, which are used as reference.

Clustering based modeling

The goal of cluster analysis is to partition data points into different groups. Similarity of points is defined by a minimal intra-cluster distance, whereas different clusters aim for a maximum inter-cluster distance. Thus, cluster analysis can be utilized to find the pattern of a system directly using the multi-dimensional data without explicit descriptions about the system features. This is the main advantage in using cluster analysis for modeling complex systems with seasonal components, e.g. WPP.

In the presented solution, a system model for anomaly detection should characterize the normal system behavior and can be used to identify unusual behavior. For most complex systems, the normal behavior might consist of multiple modes that depend on different factors, e.g. work environments, operations of the systems. When the cluster analysis is performed on a data set representing the normal behavior of a system, multiple clusters can be recognized. Each cluster (group) represents a particular status of the system. Then such multiple clusters can be used as the normal behavior model of a system for anomaly detection.

In this paper, two well-known clustering algorithms, DBSCAN and spectral clustering, are utilized to model the normal behavior of a WPP on system level. Each of them has advantages in clustering the data with complex correlations.

DBSCAN is resistant to noise and can recognize patterns of arbitrary shapes. In DBSCAN, the density for a particular point is defined as the number of neighbor points within a specified radius of that point [17]. Two user-defined parameters are required: Eps - the radius; $MinPts$ - the minimal number of neighbors in the Eps . DBSCAN uses such center-based density to classify the data points as core point (Eps -neighbors $\geq MinPts$), border point (not core point but the neighbor of minimal one core point) or noise point (neither a core nor a border point). Two core points that are within Eps of each other are defined as density-reachable core points. DBSCAN partitions the data into clusters by iteratively labeling the data points and collecting density-reachable core points into same cluster. As result, DBSCAN delivers several clusters in which noise points are also collected in a cluster. DBSCAN is not suitable to cluster high dimensional data because density is more difficult to define in high dimensional space. Therefore, a method to reduce dimensionality should be applied to the data before using the DBSCAN. This leads to a density based description of the normal behavior.

This method assumes that the training data perfectly describe the distribution of system normal states. For WPP, some special states of the plant occur so rarely that the recorded data can not represent such special states very well. In addition, environmental influences lead to noise points within the data set. Therefore, a complete coverage of the normal states of a WPP in learning data set is unrealistic to achieve.

Compared to the traditional approaches to clustering (e.g. k-means, DBSCAN), spectral clustering can generally deliver better results and can be solved efficiently by standard linear algebra methods [18]. Another advantage of spectral clustering is the ability to handle the high dimensional data using spectral analysis. Thus, extra dimensionality reduction method is not required. The idea of spectral clustering is to represent the data in form of a similarity graph $G(V, E)$ where each vertex $v_i \in V$ presents a data point

Algorithm 1 PCA based modeling

```

1: Input:  $\mathbf{X}$  ▷ learning data set
2: Output:  $\text{Model}_{\mathbf{X}}$  ▷ model of input data

3: procedure PCA_BASED_MODELING ( $\mathbf{X}$ )
4:   1: reduced dimensionality

5:    $\text{PCA\_Matrix} = \text{performPCA}(\mathbf{X})$ 
6:    $\mathbf{X}_{PCA} = \text{mapToLowDimension}(\mathbf{X})$ 
7:    $\text{Model}_{\mathbf{X}} = \text{generate\_N-Tree}(\mathbf{X}_{PCA})$ 
8: end procedure

9: function GENERATE_N-TREE( $\mathbf{X}_{PCA}$ )
10:  Tree: List with length  $2^l$ 
11:  for ( $\mathbf{x}_{pca}$  in  $\mathbf{X}_{PCA}$ ) do
12:     $\mathbf{i} = \text{determine\_orthant}(\mathbf{x}_{pca})$ 
13:     $\text{Tree}_{\mathbf{i}} = \text{append}(\text{Tree}_{\mathbf{i}}, \mathbf{x}_{pca})$ 
14:  end for
15:  for ( $\text{leaf}$  in  $\text{Tree}$ ) do
16:    if ( $\text{sizeOf}(\text{leaf}) > 1$ ) then
17:       $\text{leaf} = \text{generate\_N-Tree}(\text{leaf})$ 
18:    end if
19:  end for
20:  return ( $\text{Tree}, \text{PCA\_Matrix}$ )
21: end function
    
```

in the dataset. Each edge $e_{ij} \in E$ between two vertices v_i and v_j carries a non-negative weight (similarity between the two points) w_{ij} . Then, the clustering problem can be handled as graph partition[19]. G will be divided into smaller components, such that the vertices within the small components have high connection and there are few connections between the small components. These small components correspond to the clusters in the results of spectral clustering and can be used as normal status model for anomaly detection.

PCA based modeling

Algorithm 1 presents the stated modeling solution for a system-level approach to a WPP. The algorithm utilizes the Principal Component Analyses (see, line 5 algorithm 1) as a very first step to achieve a dimensional reduced description of the training data set. Although a part of the information is lost due to the reduction, the sensor correlations in the low dimensional space are reduced drastically, which minimizes the computational effort.

The PCA is based on the assumption, that most of the information is located in the direction of most variance. Therefore, this method aims to project a data set to a subspace with a lower dimension by minimizing the sum of squares of \mathbf{y}_i and to their projections θ_i following cost function:

$$\sum_{i=1}^m = \|\mathbf{y}_i - \theta_i\|^2.$$

Let $\mathbf{x}_1, \dots, \mathbf{x}_m$ be the data point of m sensor values and \mathbf{X} is a historical dataset of N scaled data points.

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,m} \end{bmatrix} \in \mathbf{R}^{N \times m}$$

Then as first step for computing the PCA, the covariance matrix is formed as

$$\Sigma_0 \approx \frac{1}{N-1} \mathbf{X}^T \mathbf{X}$$

By means of EVD (eigen value decomposition) or the equivalent SVD (singular value decomposition) the covariance matrix is decomposed as follows:

$$\Sigma_0 = P^T \Lambda P, \Lambda = \begin{bmatrix} \Lambda_{pc} & 0 \\ 0 & \Lambda_{res} \end{bmatrix}$$

With $\Lambda = \text{diag}(\sigma_1^2 < \sigma_2^2 < \dots < \sigma_m^2)$ where σ_i , $i = 1, \dots, m$ is the i -th eigenvalue and P is a matrix of the eigenvectors, sorted according to the eigenvalues of Λ . Λ_{pc} are the chosen principal components according to a threshold l and Λ_{res} denotes the less informative rest. l is a parameter which depends on the eigenvalues proportion of total variance and determines the dimension of the reduced normal space.

$$\mathbf{Y} = P^T \mathbf{X}$$

Transforms the p -dimensional dataset \mathbf{X} into a dataset \mathbf{Y} of a lower dimension l , with a minimum of information loss. The axes of the dimensionally reduced data space are orthonormal and aligned to the maximum variance of data. Prerequisite for modeling a WPP with this kind of transformation is the input data to calculate eigenvalues and the rotation matrix. Therefore, the presented data set of a WPP needs to describe a period of fault free operation, which is denoted by the term 'normal state'. Using this data set as a learning base, the PCA described above spans a reduced normal state space, where signal covariances are taken into account due to the eigenvalues of the covariance matrix as the basis for transformation. The input variables are transformed within the algorithm 1 in line 6.

In comparison to clustering methods only the covariance matrix stores explicit shape informations. This leads to the necessity of taking into account all data points for classifying a new observation. That is why computational effort for this model increases with the number of data points in the data set and their dimension. To overcome this issue, the model is extended with an N-Tree as geometrical data structure (see function *generate_N-Tree* in algorithm 1). The axis of the PCA transformed normal state space divides the data into 2^l subspaces. Centering these subspaces in each iteration divides the subspaces recursively until each leaf of the tree contains one data point or is empty. Note, that the mean of each subspace needs to be stored.

4.2 Step 2: Anomaly Detection

To comply with point III (see section 1), the prerequisite for a system-level anomaly detection is a data-driven model as stated above. Given such a model, a distance measure is needed to calculate the deviation between a new system observation and the model in order to identify anomalies. Therefore, an observation vector needs to be transformed into the dimensionally reduced space of the model. Then the deviation of an actual observation and the learned model can be calculated using a distance metric, such as Euclidean distance, Mahalanobis distance or Manhattan distance.

DBSCAN generated cluster provide a discrimination of core and border data points. Distance computation in DBSCAN use the euclidean distance metric. Only core points are used to measure the distance between an observations

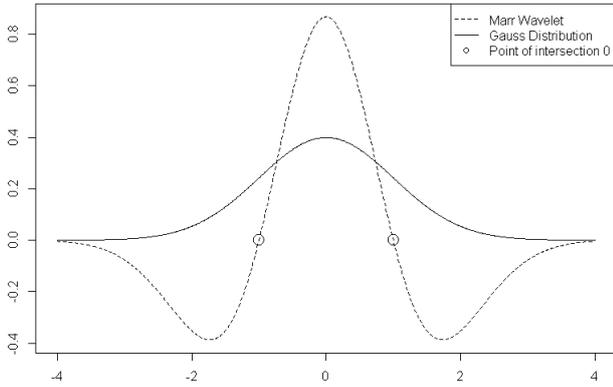


Figure 3: Characteristics of Gaussian distribution in comparison to Marr Wavelet (dashed). Spots are marked where the Marr Wavelet reach zero

and the core points. This leads to the decision whether an observation is part of the models cluster or not.

Spectral clustering computes clusters in a dimensionally reduced space but gives no further information about core or border points. Measuring the distance between such clusters can be achieved by a prototype, for example the cluster center. Then, for computing the distance, a metric like the Mahalanobis distance is used, which is sensible for the multidimensionality of such cluster. Representing a cluster based on a prototype is a generalization.

The PCA based modeling approach uses the dimensionally reduced input data as description of the multidimensional normal state space. Algorithm 2 shows how the model, computed with algorithm 1, is used for anomaly detection. At first a new observation is mapped to the low dimensional space of the model, using the rotation matrix from the PCA (see line 5). Then the mapped observation is compared with the normal state space. Therefore, the N-Tree is searched for its corresponding subset first (see function *get_subset*). If an empty leaf is found, all neighbor leafs are aggregated to a most relevant subset of data points. As the data is not generalized by border points or cluster means as prototypical points it is necessary to measure distance of the observation to each point of this subset. Now the distance is computed (see line 7).

Absolute distance measuring is missing a threshold to decide when an observation meets the model or not. Even when utilizing a Gaussian density function to provide an indicator for classification, a threshold needs to be estimated for classification. In this project, a Marr wavelet function is used to decide whether a new observation is part of the learned normal space. Instead of a Gaussian distribution the characteristic form of a Marr wavelet [20] allows a classification where the threshold can be set to zero, see figure 3. Taking into account the Marr wavelet and the euclidean distance function the process of distance measuring is computed as follows.

Let $X_{pca} = [x_1, \dots, x_l]$ be a vector of the models' principal normal-space and $O_{pc} = [o_1, \dots, o_l]$ a transformed observation, where l denotes the number of principal components. Then the distribution function to measure if a new observation is part of the normal state space is formed as:

$$\psi(X_{pca}, O_{pca}) = \frac{2}{\sqrt{3\sigma\pi^{\frac{1}{4}}}} \cdot 1 - \frac{k}{\sigma^2} \cdot \exp\left(-\frac{k}{2\sigma^2}\right)$$

Algorithm 2 Anomaly detection

```

1: Input: Tree      ▷ (Learned model, see algorithm 1)
2: Input: O        ▷ Input observation
3: Output: Boolean ▷ Anomaly
4: procedure ANOMALY_DETECTION(Tree, O)
5:   OPCA = mapToLowDimension(O)
6:   subset = get_subset(Tree, O)
7:   dist = calculate_distance(O, subset)

8:   if ( dist > 0 ) then
9:     anomaly: TRUE
10:  else
11:    anomaly: FALSE
12:  end if
13:  return ( anomaly )
14: end procedure

15: function GET_SUBSET( Tree , OPCA )
16:   i = determine_orthant(xpca)
17:   if ( size(leafi) > 1 ) then
18:     get_subset(leafi)
19:   else
20:     subset = neighbors(leafi)
21:   end if
22:   return ( subset )
23: end function
    
```

Where l denotes dimensions of reduced normal-space and

$$k = \sqrt{\sum_i^l (O_{pcai} - X_{pcai})^2}$$

k is the l -space euclidean distance. For $\psi > 0$ an observation in principal space O_{pca} is denoted part of the normal state space (see line 17).

5 Results

The data used in the evaluation is collected over a duration of 4 years from 11 real WPPs in Germany with 10 minutes resolution. The dataset consists of 12 variables which describe the work environment (e.g. wind speed, air temperature) and the status of WPP (e.g. power capacity, rotation speed of generator, voltage of the transformer).

For evaluation, a training data set of 232749 observations of the 10 minutes resolution was used to model the normal behavior of a WPP. The evaluation data set of 11544 observations contains 4531 reported failures and 7013 observations of normal behavior. Table 1 shows the confusion matrix [21] as a result of the evaluation. Here, true negative denotes a correct predicted normal state and true positive a correct classified failure. For this use case, the F1-score is used to analyze the system's performance in anomaly detection. Also, the runtime for the evaluation is denoted in Table 1 to compare speed performance of the different analyzed methods.

As can be seen, the presented PCA based algorithm outperforms the standardized spectral clustering. Especially a significant performance boost in computation time is achieved due to the extended N-Tree data structure.

Both, DBSCAN and Spectral Clustering, rely on complete sensor information for clustering the data set. A defect sensor leads to a maintenance action. The delay for this

	True Pos.	True Neg.	False Pos.	False Neg.	Bal. Acc.	F-Measure	elapsed Time
DBSCAN	1812	6827	186	2719	68.66%	55.50%	3s
Spectral Clustering	3832	6328	685	699	87.40%	84.71%	6637s
PCA based	3970	6517	496	561	90.27%	88.25%	68s

Table 1: Evaluation results of wind power station data.

maintenance is based on the localization of the WPP and cause missing sensor values for a certain time. To be operable in the use case of WPP such a model needs a fall back strategy in case of missing sensor values. Here, redundancy and correlation of different sensors comes in handy. By extending the PCA to a Probabilistic Principal Component Analyzes (PPCA), missing values can be estimated according to the data learned from the data set. Tipping and Bishop [22] extend a classic PCA by a probability model. This model assumes Gaussian distributed latent variables which can be inferred from the existing variables and the matrix of eigenvectors from the PCA. With the use of a PPCA, the solution for a system-level is robust enough to stay reliable even when sensors are missing. This was tested by training the model with a defect data set, containing 10% missing sensor values. While evaluating this model, also 10% of the data was damaged, simulating missing sensor values. The result of this evaluation is presented in table 1.

6 Conclusion

In this work a solution for system-level anomaly detection was presented. Three main requirements are identified and satisfied: At first a hardware concept for sensor data acquisition in the heterogeneous environment of WPPs was developed. This hardware logs existing sensor values and offers an adaptive solution to integrate new sensors on demand. Second, generic data-driven algorithms to automatically compute a system-level model out of minimal labeled, historical sensor data is presented. At last an anomaly detection method has been shown, which reaches an F-Measure of 89.02% and a balanced accuracy of 91.46%. This solution is not specialized for specific parts of a WPP and can be trained in a short period. With an extension of the standard PCA to a probabilistic PCA, the robustness of the algorithmic solution against sensor failures is ensured.

In the future, this solution will be evaluated using data from more WPPs with different working environment. Beyond the task of anomaly detection, diagnosis of the root cause of an anomaly is also a sensible functionality of a CM system. The presented solution will be extended by a root cause analysis. Such an extension can support maintenance personal to trace the detected anomaly. Another focus will be the prognosis of anomalies in a WPP. To achieve this, an appropriate algorithm will be developed to predict the future system status using the learned model of the system behavior.

Acknowledgments

Funded by the German Federal Ministry for Economic Affairs and Energy, KF2074717KM3 & KF2074719KM3

References

- [1] Global Wind Energy Council. Global wind statistics 2014. Available online at http://www.gwec.net/wp-content/uploads/2015/02/GWEC_GlobalWindStats_2014_FINAL_10.2.2015.pdf, March 2015.
- [2] European Wind Energy Association et al. Wind in power, 2012 european statistics, 2013. Available online at http://www.ewea.org/fileadmin/files/library/publications/statistics/Wind_in_power_annual_statistics_2012.pdf, March 2015.
- [3] Julia Nilsson and Lina Bertling. Maintenance management of wind power systems using condition monitoring systems life cycle cost analysis for two case studies. *Energy Conversion, IEEE Transactions on*, 22(1):223–229, 2007.
- [4] Wenxian Yang, Peter J Tavner, Christopher J Crabtree, Y Feng, and Y Qiu. Wind turbine condition monitoring: technical and commercial challenges. *Wind Energy*, 17(5):673–693, 2014.
- [5] AS Zaher and SDJ McArthur. A multi-agent fault detection system for wind turbine defect recognition and diagnosis. In *Power Tech, 2007 IEEE Lausanne*, pages 22–27. IEEE, 2007.
- [6] Li Zhen, He Zhengjia, Zi Yanyang, and Chen Xuefeng. Bearing condition monitoring based on shock pulse method and improved redundant lifting scheme. *Mathematics and computers in simulation*, 79(3):318–338, 2008.
- [7] Saad Chakkor, Mostafa Baghour, and Abderrahmane Hajraoui. Performance analysis of faults detection in wind turbine generator based on high-resolution frequency estimation methods. *arXiv preprint arXiv:1409.6883*, 2014.
- [8] E Jasiniien, R Raituis, A Voleiis, A Vladiauskas, D Mitchard, M Amos, et al. Ndt of wind turbine blades using adapted ultrasonic and radiographic techniques. *Insight-Non-Destructive Testing and Condition Monitoring*, 51(9):477–483, 2009.
- [9] Oliver Niggemann and Volker Lohweg. On the diagnosis of cyber-physical production systems: State-of-the-art and research agenda. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.
- [10] O. Bennouna, N. Heraud, and Z. Leonowicz. Condition monitoring and fault diagnosis system for offshore wind turbines. In *Environment and Electrical Engineering (EEEIC), 2012 11th International Conference on*, pages 13–17, May 2012.
- [11] Ning Fang and Peng Guo. Wind generator tower vibration fault diagnosis and monitoring based on pca. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 1924–1929, May 2013.

- [12] M. Celik, F. Dadaser-Celik, and A.S. Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pages 91–95, June 2011.
- [13] P. Baraldi, F. Di Maio, and E. Zio. Unsupervised clustering for fault diagnosis. In *Prognostics and System Health Management (PHM), 2012 IEEE Conference on*, pages 1–9, May 2012.
- [14] Karlheinz Schwarz and Im Eichbaeumle. Iec 61850, iec 61400-25 and iec 61970: Information models and information exchange for electric power systems. *Proceedings of the Distributech*, pages 1–5, 2004.
- [15] Richard A Zatorski. System and method for controlling multiple devices via general purpose input/output (gpio) hardware, June 27 2006. US Patent 7,069,365.
- [16] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.
- [17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*, 1996.
- [18] Ulrike von Luxburg. A tutorial on spectral clustering. In *Statistics and Computing*, 17 (4), 2007.
- [19] Shifei Ding, Liwen Zhang, and Yu Zhang. Research on spectral clustering algorithms and prospects. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 6, 16-18 2010.
- [20] Lei Nie, Shouguo Wu, Jianwei Wang, Longzhen Zheng, Xiangqin Lin, and Lei Rui. Continuous wavelet transform and its application to resolving and quantifying the overlapped voltammetric peaks. *Analytica chimica acta*, 450(1):185–192, 2001.
- [21] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.
- [22] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Using Incremental SAT for Testing Diagnosability of Distributed DES

Hassan IBRAHIM¹ and Philippe DAGUE¹ and Laurent SIMON²

¹LRI, Univ. Paris-Sud and CNRS, Orsay, France

hassan.ibrahim@lri.fr, philippe.dague@lri.fr

²LaBRI, Univ. Bordeaux and CNRS, Bordeaux, France

lsimon@labri.fr

Abstract

We extend in this work the existing approach to analyse diagnosability in discrete event systems (DES) using satisfiability algorithms (SAT), in order to analyse the diagnosability in distributed DES (DDES) and we test this extension. For this, we handle observable and non observable communication events at the same time. We also propose an adaptation to use incremental SAT over the existing and the extended approaches to overcome some of the limitations, especially concerning the length and the distance of the cycles that witness the non diagnosability of the fault, and improve the process of dealing with the reachability limit when scaling up to large systems.

1 Introduction

Diagnosis task is mainly using the available observations to explain the difference between the expected behavior of a system and its real behavior which may contain some faults. Many works have been done to study the automatic approaches to system fault diagnosis. They all try to deal with the main problem, i.e. the compromise between the number of possible diagnoses to the considered faults and the number of observations which must be given to make the decision. Diagnosis problem is NP-hard and one always needs to cope with an explosion in the number of system model states. Moreover, the diagnosis decision is not always certain, and thus running a diagnosis algorithm may not be accurate. For example, two sets of observations provided by different sets of sensors or at different times may lead to different diagnoses. This uncertainty raises the problem of diagnosability which is essential while designing the system model. After that, the model based diagnosis will be used in applications to explain any anomaly, with a guarantee of correctness and precision at least for anticipated faults. Diagnosability of the considered systems is a property defined to answer the question about the possibility to distinguish any possible faulty behavior in the system from any other behavior without this fault (i.e., correct or with a different fault) within a finite time after the occurrence of the fault. A fault is diagnosable if it can be surely identified from the partial observation available in a finite delay after its occurrence. A system is diagnosable if every possible fault in it is diagnosable. This property provides information before getting into finding the explanations of the fault. It also helps in designing a robust system against faults and in

positioning the sensors to manage the observation requirements. The main difficulty in diagnosability algorithms is related to the states number explosion. Another difficulty appears when checking diagnosability of a system which is actually diagnosable, i.e. the inexistence of a counterexample witnessing non diagnosability. Thus all possibilities need to be tested as for proving the non existence of a plan in a planning problem, and usually in this case some approximations are used to avoid exploring all the search space.

The paper is structured as follows. Section 2 will introduce the system transition models for centralized DES and recall the traditional definition of the diagnosability in those models and the state of the art of encoding this definition as a satisfiability problem in propositional logic. Section 3 will present our first contribution, an extension of this state of the art to DDES with observable and non observable communication events in the same model, and will give experimental results of this extension. Section 4 is devoted to our second contribution, using incremental SAT calls to overcome the limitation when the number of steps required to check diagnosability, i.e., the length of possible paths with cycles witnessing non diagnosability, is large, and will present experimental results showing how the method scales up. Section 5 will present related works and section 6 will conclude and give our perspectives for future work.

2 Using SAT in Diagnosability Analysis of Centralized Systems

We recall first the definitions of DES models we use and of diagnosability for these models.

2.1 Preliminaries

We will use finite state machines (FSM) to model systems. We define labeled transition systems following [1].

Definition 1. A **Labeled Transition System (LTS)** is a tuple $T = \langle X, \Sigma_o, \Sigma_u, \Sigma_f, \delta, s_0 \rangle$ where:

- X is a finite set of states,
- Σ_o is a finite set of observable correct events,
- Σ_u is a finite set of unobservable correct events,
- Σ_f is a finite set of unobservable faulty events,
- $\delta \subseteq X \times (\Sigma_o \cup \Sigma_u \cup \Sigma_f) \times X$ is the transition relation,
- s_0 is the initial state.

In [2] the authors used an equivalent but more compact representation than LTS for modeling systems in order to analyze their diagnosability: succinct transition systems, that exploit the regularity in the systems structures and are expressed in terms of propositional variables, which allowed them to translate more easily to a SAT problem the twin plant method proposed by [3] for checking diagnosability.

As we aim at studying the diagnosability of DDES using SAT solvers, we will follow the model of [2] who studied the same problem in centralized DES. It represents the system states by the valuations of a finite set A of Boolean state variables where valuation changes reflect the transitions between states according to the events. The set of all literals issued from A is $L = A \cup \{-a | a \in A\}$ and \mathcal{L} is the language over A that consists of all formulas that can be formed from A and the connectives \vee and \neg . We use the standard definitions of further connectives $\Phi \wedge \Psi \equiv \neg(\neg\Phi \vee \neg\Psi)$, $\Phi \rightarrow \Psi \equiv \neg\Phi \vee \Psi$ and $\Phi \leftrightarrow \Psi \equiv (\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$. The transition relation is defined to allow two or more events to take place simultaneously. Thus each event is described by a set of pairs $\langle \phi, c \rangle$ which represent its possible ways of occurrence by indicating that the event can be associated with changes $c \in 2^L$ in states that satisfy the condition $\phi \in \mathcal{L}$.

Definition 2. A **Succinct Transition System (SLTS)** is described by a tuple $T = \langle A, \Sigma_o, \Sigma_u, \Sigma_f, \delta, s_0 \rangle$ where:

- A is a finite set of state variables,
- Σ_o is a finite set of observable correct events,
- Σ_u is a finite set of unobservable correct events,
- Σ_f is a finite set of unobservable faulty events,
- $\delta : \Sigma = \Sigma_o \cup \Sigma_u \cup \Sigma_f \rightarrow 2^{\mathcal{L} \times 2^L}$ assigns to each event a set of pairs $\langle \phi, c \rangle$,
- s_0 is the initial state (a valuation of A).

It is straightforward to show that any LTS can be represented as an SLTS (one takes $\lceil \log(|X|) \rceil$ Boolean variables and represents states by different valuations of these variables; one assigns to each occurrence of an event e labeling a transition (x, e, y) a pair $\langle \phi, c \rangle$, with ϕ expressing the valuation of x and c the valuation changes between x and y). And reciprocally any SLTS can be mapped to an LTS (see Definition 2.4 in [2]).

The formal definition of diagnosability of a fault f in a centralized system modeled by (an LTS or SLTS) T was proposed by [1] as follows:

Definition 3. Diagnosability. A fault f is diagnosable in a system T iff

$$\begin{aligned} \exists k \in \mathbb{N}, \forall s^f \in L(T), \forall t \in L(T)/s^f, |t| \geq k \Rightarrow \\ \forall p \in L(T), (P(p) = P(s^f.t) \Rightarrow f \in p). \end{aligned}$$

In this formula, $L(T)$ denotes the prefix-closed language of T whose words are called trajectories, s^f any trajectory ending by the fault f , $L(T)/s$ the post-language of $L(T)$ after s , i.e., $\{t \in \Sigma^* | s.t \in L(T)\}$ and P the projection of trajectories on observable events. The above definition states that for each trajectory s^f ending with fault f in T , for each t that is an extension of s^f in T with enough events, every trajectory p in T that is equivalent to $s^f.t$ in terms of observation should contain in it f . As usual, it will be assumed

that $L(T)$ is live (i.e., for any state, there is at least one transition issued from this state) and convergent (i.e., there is no cycle made up only of unobservable events).

A system T is said to be diagnosable iff any fault $f \in \Sigma_f$ is diagnosable in T . In order to avoid exponential complexity in the number of faults during diagnosability analysis, only one fault at a time is checked for diagnosability. It will thus be assumed in the following that there exists only one fault event f ($\Sigma_f = \{f\}$), without restriction on the number of its occurrences. Diagnosability checking has been proved in [3] to be polynomial in the number $|X|$ of states for LTS, so exponential in the number $|A|$ of state variables for SLTS (actually the problem is NLOGSPACE-complete for LTS and PSPACE-complete for SLTS [4]).

2.2 SLTS Diagnosability as Satisfiability

An immediate rephrasing of the definition 3 shows that T is non diagnosable iff it exists a pair of trajectories corresponding to cycles (and thus to infinite paths), a faulty one and a correct one, sharing the same observable events. Which is equivalent to the existence of an ambiguous (i.e. made up of pairs of states respectively reachable by a faulty path and a correct path) cycle in the product of T by itself, synchronized on observable events, which is at the origin of the so called *twin plant* structure introduced in [3]. This non diagnosability test was formulated in [2] as a satisfiability problem in propositional logic. We recall below this encoding with the variables and the formulas used, where superscripts t refer to time points and (e_o^t) and (\hat{e}_o^t) refer respectively to the faulty and correct events occurrences sequences (corresponding states being described by valuations of (a^t) and (\hat{a}^t)) of a pair of trajectories witnessing non diagnosability (so sharing the same observable events represented by (e^t) and forming a cycle). The increasing of the time step corresponds to the triggering of at least one transition and the extension by an event of at least one of the two trajectories. $T = \langle A, \Sigma_u, \Sigma_o, \Sigma_f, \delta, s_0 \rangle$ being an SLTS, the propositional variables are thus:

- a^t and \hat{a}^t for all $a \in A$ and $t \in \{0, \dots, n\}$,
- e_o^t for all $e \in \Sigma_o \cup \Sigma_u \cup \Sigma_f$, $o \in \delta(e)$ and $t \in \{0, \dots, n-1\}$,
- \hat{e}_o^t for all $e \in \Sigma_o \cup \Sigma_u$, $o \in \delta(e)$ and $t \in \{0, \dots, n-1\}$,
- e^t for all $e \in \Sigma_o$ and $t \in \{0, \dots, n-1\}$.

The following formulas express the constraints that must be applied at each time step t or between t and $t+1$.

1. The event occurrence e_o^t must be possible in the current state:

$$e_o^t \rightarrow \phi^t \quad \text{for } o = \langle \phi, c \rangle \in \delta(e) \quad (2.1)$$

and its effects must hold at the next time step:

$$e_o^t \rightarrow \bigwedge_{l \in c} l^{t+1} \quad \text{for } o = \langle \phi, c \rangle \in \delta(e) \quad (2.2)$$

We have the same formulas with \hat{e}_o^t .

2. The present value (*True* or *False*) of a state variable changes to a new value (*False* or *True*, respectively) only if there is a reason for this change, i.e., because of an event that has the new value in its effects (so, change without reason is prohibited). Here is the change from

True to *False* (the change from *False* to *True* is defined similarly by interchanging a and $\neg a$):

$$(a^t \wedge \neg a^{t+1}) \rightarrow (e_{i_1 o_{j_1}}^t \vee \dots \vee e_{i_k o_{j_k}}^t) \quad (2.3)$$

where the $o_{j_i} = \langle \phi_{j_i}, c_{j_i} \rangle \in \delta(e_{i_i})$ are all the occurrences of events e_{i_i} with $\neg a \in c_{j_i}$.

We have the same formulas with \hat{a}^t and $\hat{e}_{i_i o_{j_i}}^t$.

- At most one occurrence of a given event can occur at a time and the occurrences of two different events cannot be simultaneous if they interfere (i.e., if they have two contradicting effects or if the precondition of one contradicts the effect of the other):

$$\neg(e_o^t \wedge e_{o'}^t) \quad \forall e \in \Sigma, \forall \{o, o'\} \subseteq \delta(e), o \neq o' \quad (2.4)$$

$$\neg(e_o^t \wedge e_{o'}^t) \quad \forall \{e, e'\} \subseteq \Sigma, e \neq e', \forall o \in \delta(e), \forall o' \in \delta(e') \text{ such that } o \text{ and } o' \text{ interfere} \quad (2.5)$$

We have the same formulas with \hat{e}_o^t .

- The formulas that connect the two events sequences require that observable events take place in both sequences whenever they take place (use of e^t):

$$\bigvee_{o \in \delta(e)} e_o^t \leftrightarrow e^t \text{ and } \bigvee_{o \in \delta(e)} \hat{e}_o^t \leftrightarrow e^t \quad \forall e \in \Sigma_o \quad (2.6)$$

The conjunction of all the above formulas for a given t is denoted by $\mathcal{T}(t, t+1)$.

A formula for the initial state s_0 is:

$$\mathcal{I}_0 = \bigwedge_{a \in A, s_0(a)=1} (a^0 \wedge \hat{a}^0) \wedge \bigwedge_{a \in A, s_0(a)=0} (\neg a^0 \wedge \neg \hat{a}^0) \quad (2.7)$$

At last, the following formula can be defined to encode the fact that a pair of executions is found with the same observable events and no fault in one execution (first line), but one fault in the other (second line), which are infinite (in the form of a non trivial cycle, so containing at least one observable event,¹ at step n ; third line), witnessing non diagnosability:

$$\begin{aligned} \Phi_n^T = & \mathcal{I}_0 \wedge \mathcal{T}(0, 1) \wedge \dots \wedge \mathcal{T}(n-1, n) \wedge \\ & \bigvee_{t=0}^{n-1} \bigvee_{e \in \Sigma_f} \bigvee_{o \in \delta(e)} e_o^t \wedge \\ & \bigvee_{m=0}^{n-1} \left(\bigwedge_{a \in A} ((a^n \leftrightarrow a^m) \wedge (\hat{a}^n \leftrightarrow \hat{a}^m)) \right) \wedge \bigvee_{t=m}^{n-1} \bigvee_{e \in \Sigma_o} e^t \end{aligned}$$

From this encoding in propositional logic, follows the result (theorem 3.2 of [2]) that an SLTS T is not diagnosable if and only if $\exists n \geq 1$, Φ_n^T is satisfiable. It is also equivalent to $\Phi_{2|A|}^T$ being satisfiable, as the twin plant states number is an obvious upper bound for n , but often impractically high (see in [2] some ways to deal with this problem).

3 Using SAT in Diagnosability Analysis of Distributed Systems

We extend from centralized systems to distributed systems the satisfiability framework of subsection 2.2 for testing diagnosability and we provide some experimental results.

¹This verification that the cycle found is not trivial was not done in [2]; it is why the authors had to add for each time point a formula, not needed here, guaranteeing that at least one event took place, to avoid silent loops with no state change.

3.1 DDES Modeling

In order to model DDES with SLTS, we need to extend these ones by adding communication events to each component. So we use the following definition for a distributed SLTS with k different components (sites):

Definition 4. A **Distributed Succinct Transition System** (DSLTS) with k components is described by a tuple $T = \langle A, \Sigma_o, \Sigma_u, \Sigma_f, \Sigma_c, \delta, s_0 \rangle$ where (subscripts i refer to component i):

- A is a union of disjoint finite sets $(A_i)_{1 \leq i \leq k}$ of component own state variables, $A = \bigcup_{i=1}^k A_i$,
- Σ_o is a union of disjoint finite sets of component own observable correct events, $\Sigma_o = \bigcup_{i=1}^k \Sigma_{oi}$,
- Σ_u is a union of disjoint finite sets of component own unobservable correct events, $\Sigma_u = \bigcup_{i=1}^k \Sigma_{ui}$,
- Σ_f is a union of disjoint finite sets of component own unobservable faulty events, $\Sigma_f = \bigcup_{i=1}^k \Sigma_{fi}$,
- Σ_c is a union of finite sets of (observable or unobservable) correct communication events, $\Sigma_c = \bigcup_{i=1}^k \Sigma_{ci}$, which are the only events shared by at least two different components (i.e., $\forall i, \forall c \in \Sigma_{ci}, \exists j \neq i, c \in \Sigma_{cj}$),
- $\delta = (\delta_i)$, where $\delta_i : \Sigma_i = \Sigma_{oi} \cup \Sigma_{ui} \cup \Sigma_{fi} \cup \Sigma_{ci} \rightarrow 2^{\mathcal{L}_i \times 2^{\mathcal{L}_i}}$, assigns to each event a set of pairs $\langle \phi, c \rangle$ in the propositional language of the component where it occurs (so, for communication events, in each component separately where they occur),
- $s_0 = (s_{0i})$ is the initial state (a valuation of each A_i).

In this distributed framework, synchronous communication is assumed, i.e., communication events are synchronized such that they all occur simultaneously in all components where they appear. More precisely, a transition by a communication event c may occur in a component iff a simultaneous transition by c occurs in all the other components where c appears (has at least one occurrence). In particular, all events before c in trajectories in all these components necessarily occur before all events after c in these trajectories. The global model of the system is thus nothing else than the product of the models of the components, synchronized on communication events. Notice that we allow in whole generality communication events to be, partially or totally, unobservable, so one has in general to wait further observations to know that some communication event occurred between two or more components. On the other side, assuming these communications to be faultless is not actually a limitation. If a communication process or protocol may be faulty, it has just to be modeled as a proper component with its own correct and faulty behaviors (the same that, e.g., for a wire in an electrical circuit). In this sense, communications between components are just a modeling concept, not subject to diagnosis. It will be also assumed that the observable information is global, i.e. centralized (when observable information is only local to each component, distributed diagnosability checking becomes undecidable [5]), allowing to keep definition 3 for diagnosability.

3.2 DSLTS Diagnosability as Satisfiability

Let T be a DSLTS made up of k components denoted by indexes i , $1 \leq i \leq k$. In order to express the diagnosability analysis of T as a satisfiability problem, we have to extend

the formulas of subsection 2.2 to deal with communication events between components. Let $\Sigma_c = \Sigma_{c_o} \cup \Sigma_{c_u}$ be the communication events, with $\Sigma_{c_o} = \cup_{i=1}^k \Sigma_{c_o i}$ the observable ones and $\Sigma_{c_u} = \cup_{i=1}^k \Sigma_{c_u i}$ the unobservable ones.

The idea is to treat each communication event as any other event in each of its owners and, as it has been done with events e^t for $e \in \Sigma_o$ for synchronizing observable events occurrences in the two executions, to introduce in the same way a global reference variable for each communication event at each time step, in charge of synchronizing any communication event occurrence in any of its owner with occurrences of it in all its other owners. We use one such reference variable for each trajectory, e^t and \hat{e}^t , for unobservable events $e \in \Sigma_{c_u}$, and only one for both trajectories, e^t , for observable events $e \in \Sigma_{c_o}$ as it will also in addition play the role of synchronizing observable events between trajectories exactly as the e^t for $e \in \Sigma_o$. So, we add to the previous propositional variables the new following ones:

- e_o^t, \hat{e}_o^t for all $e \in \Sigma_c, o \in \delta(e) = \cup_i \delta_i(e)$ and $t \in \{0, \dots, n-1\}$,
- e^t for all $e \in \Sigma_c, \hat{e}^t$ for all $e \in \Sigma_{c_u}$ and $t \in \{0, \dots, n-1\}$.

Formulas in $\mathcal{T}(t, t+1)$ are extended as follows.

1. Formulas (2.1), (2.2), (2.3) and (2.5) extend unchanged to e_o^t and $\hat{e}_o^t \forall e \in \Sigma_c$, expressing that a communication event must be possible and has effects in each of its owner components and that two such different events cannot be simultaneous if they interfere.
2. Formulas (2.4) extend to prevent two simultaneous occurrences of a given communication event in the same owner component, i.e. apply $\forall e \in \Sigma_c, \forall i, \forall \{o_i, o_i'\} \subseteq \delta_i(e), o_i \neq o_i'$ and the same with \hat{e} (obviously they do not apply to different owner components, by the very definition of communication events).
3. Finally, the new following formulas express the communication process itself, i.e. the synchronization of the occurrences of any communication event e in all its owners components ($S(e)$ being the set of indexes of the owners components of e) and extend also formulas (2.6) to observable communication events:

$$\bigvee_{o_i \in \delta_i(e)} e_{o_i}^t \leftrightarrow e^t \text{ and } \bigvee_{o_i \in \delta_i(e)} \hat{e}_{o_i}^t \leftrightarrow \hat{e}^t \quad \forall e \in \Sigma_{c_u} \forall i \in S(e)$$

$$\bigvee_{o_i \in \delta_i(e)} e_{o_i}^t \leftrightarrow e^t \text{ and } \bigvee_{o_i \in \delta_i(e)} \hat{e}_{o_i}^t \leftrightarrow e^t \quad \forall e \in \Sigma_{c_o} \forall i \in S(e)$$

The formula Φ_n^T is unchanged except that, in the verification that the found cycle (third line) is not trivial, any observable event can be used, so the final disjunct of events e^t is extended to all $e \in \Sigma_o \cup \Sigma_{c_o}$. We have thus the result that a DSLTS T is not diagnosable if and only if $\exists n \geq 1, \Phi_n^T$ is satisfiable.

3.3 Implementation and Experimental Testing

We have implemented the above extension in Java. We used the well designed API of the SAT solver Sat4j [6]. If more efficient solvers could have been chosen, it fitted well our clause generator written in Java and only a limited speed up can be awaited from C++ solvers (a speed up of 4, i.e. reduction of 75% of the runtime is often observed).

We have tested our tool on small examples with several communication events with multiple occurrences (three communicating components) with global communication (all components share the same event) or partial communication (only some components share the same event), as in Figure 1, which was the running example in [7].

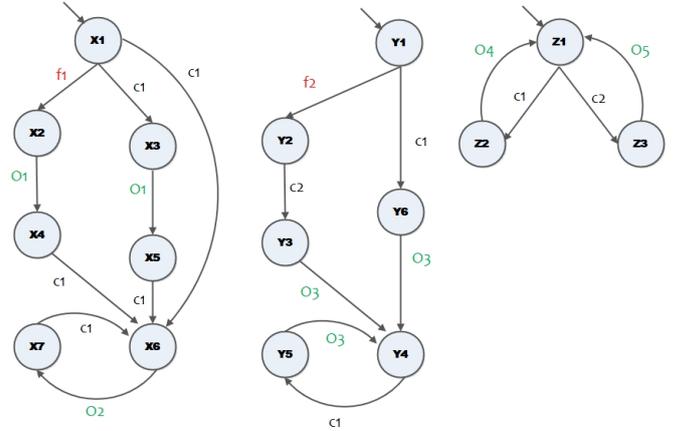


Figure 1: A DDES made up of 3 components C1, C2 and C3 from left to right. $c_{i,1 \leq i \leq 2}$ are unobservable communication events, $o_{i,1 \leq i \leq 5}$ are observable events and $f_{i,1 \leq i \leq 2}$ are faulty events.

The total number of propositional variables $VarsNum$ in the generated formula Φ_n^T after n steps is:

$$VarsNum = n \times (2|A| + 3 \sum_{i=1}^{Obs} ObOcc_i + \sum_{i=1}^{Faults} FaultOcc_i + 2 \sum_{i=1}^{Unobs} UnobOcc_i),$$

where: $|A|$ is the total number of state variables, Obs the total number of observable events, $ObOcc_i$ the total number of occurrences of the observable event e_i , $Faults$ the total number of faults, $FaultOcc_i$ the total number of occurrences of the faulty event e_i , $Unobs$ the total number of unobservable correct events, $UnobOcc_i$ the total number of occurrences of the unobservable correct event e_i .

The results are in Table 1, where the columns show the system and the fault considered (3 cases), the steps number n , the numbers of variables and clauses and the runtime.

System	Fault	Steps	SAT?	Variables	Clauses	runtime(ms)
C2	f2	4	No	106	628	27
C2	f2	5	Yes	131	783	15
C2, C3	f2	5	No	225	1157	28
C2, C3	f2	32	No	1386	7340	641
C2, C3	f2	64	No	2762	14668	1422
C2, C3	f2	128	No	5514	29324	5061
C2, C3	f2	256	No	11018	58636	18970
C2, C3	f2	512	No	22026	117260	130164
C2, C3	f2	1024	No	44042	234508	548644
C1, C2, C3	f1	8	No	576	3546	91
C1, C2, C3	f1	9	Yes	646	3987	110

Table 1: Results on the example of Figure 1.

Which means that $f2$ is not diagnosable in C2 alone while it becomes diagnosable when synchronizing C2 and C3. For this last result, we have increased the steps number until reaching $2^{2|A|}$, which is the theoretical upper bound of the twin plant states represented in the logical formula. As in general it is not always possible to reach this bound in practice, we propose in section 4 using incremental SAT to improve the management of increasing steps number. While

f_1 is not diagnosable even after synchronizing all three components together. Numbers of variables and clauses are small in comparison to what SAT solvers can handle (up to hundred thousands propositional variables and millions of clauses). These tests are mentioned as a proof of concept. However, to test the tool on larger systems and because of the absence of benchmark in the literature, we have created in subsection 4.2. an example that can be scaled up.

4 Adaptation to Incremental SAT Diagnosability Checking

We adapt satisfiability algorithms for checking diagnosability of both centralized (subsection 2.2) and distributed (subsection 3.2) DES in order to incrementally process the maximum length of paths with cycles searched for witnessing non diagnosability and we provide experimental results.

4.1 Diagnosability as Incremental Satisfiability

Two cases have to be distinguished while testing diagnosability using SAT solvers to verify the satisfiability of the logical formula Φ_n^T for a given n [2]. The first case is when we find a model for Φ_n^T , which *definitely* indicates the non diagnosability of the studied fault. The second case is when we do not find such a model: this result indicates just that the studied fault has not been found non diagnosable according to the value of n . In other words, after testing all the possible first n steps, we did not find a pair of executions of length at most n containing cycles such that one of them contains the fault and not the other and such that the two executions are equivalent in terms of observation. However, as the theoretical upper bound $n = 2^{2^{|A|}}$ which would guarantee that the fault is actually diagnosable is often in practice unreachable, such a pair may exist for a greater value of n . Testing it means increasing n and *rebuilding the logical formula* Φ_n^T then recalling the SAT solver.

Instead, we propose to adapt the formula Φ_n^T in order to be tested in an *incremental* SAT mode by multiple calls to a Conflict Driven Clause Learning (CDCL) solver. Using CDCL solvers in a specialized, incremental, mode is relatively new but already widely used [8] in many applications. In this operation mode, the solver can be called many times with different formulas. However, solvers are designed to work with similar formulas, where clauses are removed and added from calls to calls. Learnt clauses can be kept as soon as the solver can ensure that clauses used to derive them are not removed. This is generally done by adding specialized variables, called *assumptions*, to each clause that can be removed. By assuming the variable to be *False*, the clause is activated and by assuming the variable to be *True*, the clause is trivially satisfied and no longer used by the solver. What is interesting for our purpose is that the CDCL solver can save clauses learnt during the previous calls and test multiple assumptions in each new call. This means that after n steps we hope that the solver will have learnt some constraints about the behavior of the system. Although we are interested in testing the diagnosability property on a defined system, this property is independent from the system behavior which can be learnt by the solver from the previous calls.

In order to extend the clauses representation given in subsections 2.2 and 3.2 to this mode of operation, we propose to divide the formula Φ_n^T in two parts. The first part \mathcal{T}_n describes the first n steps, synchronized on the observations,

of the behavior of both trajectories (represented by the conjunction of formulas $\mathcal{T}(t, t+1)$, $0 \leq t \leq n-1$, representing the $(t+1)$ th step). The second part \mathcal{D}_n describes the diagnosability property at step n , i.e., the occurrence of a fault in the n previous steps of the faulty trajectory (given by the formula \mathcal{F}_n) and the detection of a cycle at step n (given by the formula \mathcal{C}_n). So we obtain, for $n \geq 1$:

$$\Phi_n^T = \mathcal{T}_n \wedge \mathcal{D}_n$$

$$\mathcal{T}_n = \mathcal{I}_0 \wedge \bigwedge_{t=0}^{n-1} \mathcal{T}(t, t+1) \quad \mathcal{D}_n = \mathcal{F}_n \wedge \mathcal{C}_n$$

$$\mathcal{F}_n = \bigvee_{t=0}^{n-1} \bigvee_{e \in \Sigma_f} \bigvee_{o \in \delta(e)} e^t$$

$$\mathcal{C}_n = \bigvee_{m=0}^{n-1} \left(\bigwedge_{a \in A} ((a^n \leftrightarrow a^m) \wedge (\hat{a}^n \leftrightarrow \hat{a}^m)) \right) \wedge \bigvee_{t=m}^{n-1} \bigvee_{e \in \Sigma_o} e^t$$

Add now at each step j a control variable h_j allowing to disable (when its truth value is *False*) or activate (when its truth value is *True*) the formulas \mathcal{F}_j and \mathcal{C}_j and keep at step n all these controlled formulas for $1 \leq j \leq n$. We obtain the following Ψ_n^T formula, for $n \geq 1$:

$$\Psi_n^T = \mathcal{T}_n \wedge \bigwedge_{j=1}^n \mathcal{D}_j' \quad \mathcal{D}_j' = \mathcal{F}_j' \wedge \mathcal{C}_j' \quad 1 \leq j \leq n$$

$$\mathcal{F}_j' = \neg h_j \vee \mathcal{F}_j \quad \mathcal{C}_j' = \neg h_j \vee \mathcal{C}_j \quad 1 \leq j \leq n$$

We have thus the equivalence, for all $n \geq 1$:

$$\Phi_n^T \equiv \Psi_n^T \wedge h_n \wedge \bigwedge_{j=1}^{n-1} \neg h_j$$

This allows one, for all $n \geq 1$, to replace the SAT call on Φ_n^T by a SAT call on Ψ_n^T under the control variables setting given by $H_n = \{\neg h_1, \dots, \neg h_{n-1}, h_n\}$ (indicated in a second argument of the call):

$$SAT(\Phi_n^T) = SAT(\Psi_n^T, H_n)$$

The idea is now to consider the control variables h_j as assumptions and use incremental SAT calls $IncSAT_j$ under varying assumptions, for $1 \leq j \leq n$. For this, we use the following recurrence relationships for both formulas Ψ_j^T and assumptions H_j :

$$\Psi_0^T = \mathcal{I}_0 \quad \Psi_{j+1}^T = \Psi_j^T \wedge \mathcal{T}(j, j+1) \wedge \mathcal{D}_{j+1}' \quad j \geq 0$$

$$H_1 = \{h_1\} \quad H_{j+1} = H_j \cup \{\neg h_j, h_{j+1}\} \quad j \geq 1$$

where the notation $H_j[\{ass_i\}]$ means updating in H_j assumptions h_i by their new settings ass_i , i.e., in the formula above, replacing the truth value of h_j , which was *True*, by *False*, and adding the new assumption h_{j+1} with truth value *True*. From these relationships, the unique call to SAT under given assumptions $SAT(\Psi_n^T, H_n)$ can be replaced, starting with the set of clauses \mathcal{I}_0 , by multiple calls, $0 \leq j \leq n-1$, to an incremental SAT under varying assumptions:

$$IncSAT_{j+1}(NewClauses_{j+1}, NewAssumptions_{j+1})$$

$$= IncSAT_{j+1}(\mathcal{T}(j, j+1) \wedge \mathcal{D}_{j+1}', \{\neg h_j, h_{j+1}\}) \quad (4.1)$$

If $IncSAT_j$ answers SAT, the search is stopped as non diagnosability is proved, if it answers UNSAT, then $IncSAT_{j+1}$ is called.

Notice that we used a unique assumption h_j for controlling both \mathcal{F}_j and \mathcal{C}_j as non diagnosability checking requires the presence of both a fault occurrence in the faulty trajectory and of a cycle. But the same framework allows the independent control of formulas by separate assumptions. For sake of simplicity, we also assumed we called *IncSAT* at each step, but this is not mandatory and indexes j for the successive calls can be decoupled from indexes t for steps. We should also say that, even if *IncSAT* allows us to reactivate an already disabled clause, we are sure in our case to never use this function (when h_k has been set to *False*, it always remains so) and we can thus force the solver to do a hard simplification process that removes the forgotten clauses permanently. As a result of our adaptation we will be able to scale up the size of the tested system and the distance and length of a cycle witnessing non diagnosability.

4.2 Experimental Results

We show in this subsection a comparison between our adapted version of subsection 4.1, that uses incremental SAT, and the previous versions, for centralized model (subsection 2.2 following [2]) and for distributed model (subsection 3.2). We have created the example in Figure 2 which contains $2k + 1$ components: one faulty component and two sets of k neighboring components. The faulty component has two separated paths, each one containing k different successive unobservable events c_i and ending with the same observable cycle of length 1, but only one of them contains the fault. The centralized model will be limited to this faulty component alone and thus in this case the events c_i , $1 \leq i \leq 2k$, are just unobservable events as is u . In the distributed model, these events c_i are communication events and the faulty component is considered with the other two sets of components, where each component in both sets shares one event c_i with the faulty component to ensure a number $2k$ of communications before arriving to the cycles that will witness the non diagnosability of the fault. Each set of components will be synchronized with only one path, either the faulty path or the correct one. This allows us to study the effect of the cycle distance in both models.

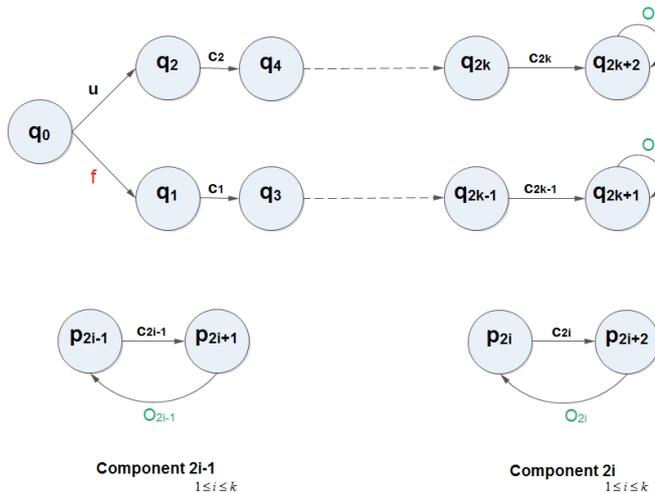


Figure 2: One faulty component that communicates with two sets of k components. Each set communicates with one path (resp. faulty and correct) in the faulty component.

The results are in Table 2 for the centralized model (for $k = 18, 28, 38, 48, 58$ and 98) and in Table 3 for the distributed

model (for $k = 3, 13, 23, 33, 43$ and 63). The length of a pair of executions with cycles witnessing the non diagnosability of f in each example is $k + 2$ and we consider the satisfiability of the formula Φ_{k+2}^T , so the number of steps required for SAT to provide the answer *Yes* is: $|\text{Steps}| = k + 2$. In order to obtain a fair comparison between *IncSAT*, which manages internally by handling assumptions the successive satisfiability checks of increasing formulas for $j = 1, \dots, k + 2$, and SAT, for which $k + 2$ successive calls are made to the solver with respective formulas Φ_n^T for $n = 1, \dots, k + 2$, the sum of the $k + 2$ runtimes of the SAT solver calls are considered in this case (last column in the tables).

Steps	Clauses	Inc. SAT(s)	SAT(s)
20	42,614	1.5	1.3
30	131,714	10.3	13.1
40	303,736	49.3	77.8
50	576,466	106	223
60	970,156	320	699
100	4,334,018	9410	13040

Table 2: Results on the faulty component of Figure 2.

Steps	Comps	Clauses	Inc. SAT(s)	SAT(s)
5	7	1,962	0.04	0.06
15	27	30,313	0.8	0.5
25	47	113,906	6.5	4.8
35	67	277,873	33.8	33.7
45	87	542,033	111	132
65	127	1,490,590	967	1090

Table 3: Results on the whole system of Figure 2.

Although these examples remain relatively simple and do not reflect any potential constraint that could be resumed by some learnt clauses (e.g. no interfering events), we can already notice the difference in runtime in favor of our incremental version in the centralized case and for the two largest values of k in the distributed case. This difference could be explained by the fact that generating all variables from the beginning for all time steps and for all events imply many meaningless clauses that would add a load on the solver in the version in [2], this load being avoided in our incremental version because of the clauses learnt by the CDCL SAT solver. From another side, we should say that generating in both versions all variables from the beginning has two main advantages: firstly, it allows the system description without unfolding it (even if this description is verbose); secondly, it allows the ordering of these variables by their time step in order to generate the constraints for only one time step and then get next steps constraints by just shifting the numbers (as we are representing the clauses in DIMACS format). One last point could help to a more efficient description of the system: in the succinct systems we represent all the occurrences of an event together, but in its SAT encoding we “unfold” this succinctness by generating for *each* occurrence n variables (for n time steps), even though logically only one of them will be assigned to *True*. We could thus mark this relation among these n copies by introducing a global cardinality constraint to express that these copies belong to only one occurrence of an event.

5 Selection of Related Works

The first introduction to the notion of diagnosability was by [1]. The authors studied diagnosability of FSM, as defined in definition 1. Their formal definition of diagnosability is the one we mentioned in definition 3. They introduced an approach to test this property by constructing a deterministic diagnoser. However, in the general case, this approach is exponential in the number of states of the system, which makes it impractical.

In order to overcome this limitation [3] introduced the *twin plant* approach, which is a special structure built by synchronizing on their observable events two identical instances of a nondeterministic fault diagnoser, and then searched for a path in this structure with an observed cycle made up of ambiguous states, i.e. states that are pairs of original states, one reached by going through a fault and the other not. Thus faults diagnosability is equivalent to the absence of such a path, called a *critical path*. This approach turns the diagnosability problem in a search for a path with a cycle in a finite automaton, and this reduces its complexity to be polynomial of degree 4 in the number of states (and exponential in the number of faults, but processing each fault separately makes its linear in the number of faults).

Let us mention here that the two previous works were interested in centralized systems with simple faults modeled as distinguished events. The first studies about fault patterns were introduced in [9] and [10] which generalize the simple fault event in a centralized DES to handle a sequence of events considered together as a fault, or handle multiple occurrences of the same fault or of different faults. More generally, a fault pattern is given as a suffix-closed rational events language (so by a complete deterministic automaton with a stable subset of final states).

The first work that addressed diagnosability analysis in DDES was [7]. A DDES is modeled as a set of communicating FSM. Each FSM has its own events set, communication events being the only ones shared by at least two different FSM. In [7] was introduced an incremental diagnosability test which avoids to build the twin plant for the whole distributed system if not needed. Thus one starts by building a local twin plant for the faulty component to test the existence of a local critical path. If such a path exists one builds the local twin checkers of the neighboring components. Local twin checker is a structure similar to local twin plant, i.e., where each path in it represents a pair of behaviors with the same observations, except that there is no fault information in it since it is constructed from non-faulty component. After constructing local twin checkers, one tries to solve the ambiguity resulting from the existence of a critical path in the local twin plant. This is done by synchronizing *on their communication events* this local twin plant with the local twin checker of one neighboring component. In other words, one tries to distinguish the faulty path from the correct one by exploiting the observable events in the neighboring components, because these events occurrences that are consistent with the occurrences of the communication events could solve the ambiguity. The process is repeated until the diagnosability is answered, so only in the worst case has the whole system to be visited. Another important contribution in this work was to delete the unambiguous parts after each synchronization on the communication events, reducing thus the amount of information transferred to next check (if needed). The approach assumed simple

faults.

The work by [11] has optimized the construction of local twin plants, by exploiting the fact that one distinguishes two behaviors (faulty and correct) and one synchronizes at two levels (observations first and communications later). It improved the construction of the twin plants proposed by [7] by exploiting the different identifiers given to the communication events at the observation synchronization level (depending on which instance, left or right, they belong to) to assign them directly to the two behaviors studied (left copy assigned to the faulty behavior, right copy to the correct one). This helped in deleting the redundant information, then in abstracting the amount of information to be transferred later to next steps if the diagnosability was not answered. The generalization to fault patterns in DDES was introduced by [12].

After the reduction of diagnosability problem to a path finding problem by [3], it became transferable to a satisfiability problem like it is the case for planning problems [13]. This was done by [2] which formulated the diagnosability problem (in its twin plant version) into a SAT problem, assuming a centralized DES with simple fault events. The authors represented the studied transition system by a succinct representation (cf. definition 2). This allows both a compact representation of the system states and a maximum amount of non interfering events to be fired simultaneously. Thus, they represented the system states by the valuation of a set of Boolean state variables ($\lceil \log(q) \rceil$ state variables for q states) and the interference relation between two events according to the consistency among their effects and preconditions, one versus the other. They distinguished between an occurrence of an event in the faulty sequence or in the correct sequence by introducing two versions of it and constructed the logical formula expressing states transitions for each possible step in the system. Each step may contain simultaneous events that belong to faulty and correct sequences but must synchronize the occurrence of observable events whenever they take place. For a given bound n of paths length, they made the conjunct of these formulas for n steps and added the logical formula that represents the occurrence of the fault in the faulty sequence and the occurrence of a cycle in both sequences. The satisfiability of the obtained formula is equivalent to finding a critical path, i.e. to the non diagnosability of the fault (see subsection 2.2 for a summary of this approach). Although this approach allows one to test diagnosability in large systems, it has a limitation which is that we cannot dynamically increase n to ensure reaching more states while scaling up the size of the system where the cycles that witness non diagnosability can be very long. However the authors notice that we are not always forced to test all reachable states in many cases where an approximation for the reachable states can be applied, but without explaining explicitly how such an approximation can be found.

6 Conclusion and Future Works

By extending the state of the art works for centralized DES, we have expressed diagnosability analysis of DDES as a satisfiability problem by building a propositional formula whose satisfiability, witnessing non diagnosability, can be checked by SAT solvers. We allow both observable and non observable communication events in our model. Our expression of these communication events, which avoids

merging all their owner components, helps in reducing the number of clauses used to represent them and this reduction is proportional to the number of their occurrences. We have also proposed an adaptation of the logical formula in order to use incremental SAT calls helping managing the scaling up of the distance and the length of the intended cycles witnessing non diagnosability and thus the size of the tested system. Thus we exploited the clauses learnt about the system behavior in the previous calls. This approach is more practical and more efficient for complex systems than existing ones, as it avoids starting from scratch at each call.

We are now considering the extension of this work to fault patterns diagnosability [12]. We will use the same approach to express predictability analysis [14] as a satisfiability problem, for DES and DDES [15] and both for simple fault events and fault patterns [16]. Although our representation can be easily extended to deal with local observations (i.e., observable events in one component are observed only by this component), we know that in general diagnosability checking becomes then undecidable, e.g. when communication events are unobservable (obviously it remains decidable when these events are observable in all their owners) [5]. A future work will be to study decidable cases of diagnosability checking in DDES with local observations, e.g. assuming some well chosen communication events being observable. Another natural question is to study if the methods used in [7] and refined in [11] to check diagnosability in DDES in an incremental way in terms of the system components could be transposed as guiding strategies for some component incremental SAT based approach for testing diagnosability in DDES. Transposing in SAT these methods, based on building a local twin plant and local twin checkers for gaining efficiency with regards to a global checking, seems difficult. Basically, at any step k , corresponding to considering a subsystem made up of k components, these methods build all critical paths witnessing non diagnosability at the level of this subsystem and the incremental step, when adding a $(k + 1)^{th}$ neighboring component, consists in checking the consistency of these pairs with the observations in the new component: only those pairs which can be consistently extended are kept, if any. In addition, in [11], only useful and abstracted information is kept from one step to the next one. With SAT, only one critical pair witnessing non diagnosability of the subsystem (i.e., a model for the formula) will be built. If it is not consistent, and thus disappears, when adding the $(k + 1)^{th}$ component, diagnosability is not proven for all that: other critical pairs in the subsystem, not completely computed at step k , may exist and be extendible to step $(k + 1)$. So, they have to be computed now, which limits the incremental characteristic of the approach. In the same way, abstracting some information is difficult to achieve with SAT. So, there is no evidence a priori that efficiency gain could be obtained by trying to develop a component incremental SAT based approach for testing DDES diagnosability.

References

- [1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamotheen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.
- [2] J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 532–537, 2007.
- [3] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.
- [4] J. Rintanen. Diagnosers and diagnosability of succinct transition systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 538–544, 2007.
- [5] L. Ye and P. Dague. Undecidable case and decidable case of joint diagnosability in distributed discrete event systems. *International Journal On Advances in Systems and Measurements*, 6(3 and 4):287–299, 2013.
- [6] D. Le Berre and A. Parrain. The sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.
- [7] Y. Pencolé. Diagnosability analysis of distributed discrete event systems. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, 2004.
- [8] A. Nadel and V. Ryvchin. Efficient SAT solving under assumptions. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT'12)*, 2012.
- [9] T. Jérón, H. Marchand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, 2006.
- [10] S. Genc and S. Lafortune. Diagnosis of patterns in partially-observed discrete-event systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 422–427. IEEE, 2006.
- [11] L. Ye and P. Dague. An optimized algorithm for diagnosability of component-based systems. In *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES'10)*, 2010.
- [12] L. Ye, Y. Yan, and P. Dague. Diagnosability for patterns in distributed discrete event systems. In *Proceedings of the 21st International Workshop on Principles of Diagnosis (DX'10)*, 2010.
- [13] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.
- [14] S. Genc and S. Lafortune. Predictability of Event Occurrences in Partially-observed Discrete-event Systems. *Automatica*, 45(2):301–311, 2009.
- [15] L. Ye, P. Dague, and F. Nouioua. Predictability Analysis of Distributed Discrete Event Systems. In *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC-13)*, pages 5009–5015. IEEE., 2013.
- [16] T. Jérón, H. Marchand, S. Genc, and S. Lafortune. Predictability of Sequence Patterns in Discrete Event Systems. In *Proceedings of the 17th World Congress*, pages 537–453. IFAC., 2008.

Improving Fault Isolation and Identification for Hybrid Systems with Hybrid Possible Conflicts

Anibal Bregon and **Carlos J. Alonso-González** and **Belarmino Pulido**
 Departamento de Informática, Universidad de Valladolid, Valladolid, 47011, Spain
 e-mail: {anibal,calonso,belar}@infor.uva.es

Abstract

Model-based fault isolation and identification in hybrid systems is computationally expensive or even unfeasible for complex systems due to the presence of uncertainty concerning the actual state, and also due to the presence of both discrete and parametric faults coupled with changing modes in the system. In this work we improve fault isolation and identification performance for hybrid systems diagnosis using Hybrid Possible Conflicts. The Hybrid Bond Graph modeling approach makes feasible to track system behavior without enumerating the complete set of system modes. Hybrid Possible Conflicts focus the analysis on potential mode changes on those subsystems whose behavior deviates from expected. Moreover, using information derived from the Hybrid Bond Graph model, we can cope with both discrete and parametric faults in a unique framework.

Fault detection with Hybrid Possible Conflicts relied upon an statistical test to decide when a significant deviation in the residual occurs. Fault detection time was later used to start the fault isolation and identification stages. In this work we propose to analyze the evolution of the residual signal using CUSUM to find a more accurate estimation of the time of fault occurrence, which allows to improve both the potential new modes tracking and the parametric fault identification. Moreover, we extend our previous proposal for fault identification in continuous systems to cope with fault identification along a set of mode changes while performing parameter identification. We have tested these ideas in a four-tank hybrid system with satisfactory results.

1 Introduction

Complex hybrid systems are present in a broad range of engineering applications, such as mechanical systems, electrical circuits, or embedded computation systems. The behavior of these systems is made up of continuous and discrete event dynamics. The main sources of hybrid behavior are discrete actuators, like discrete valves or switches in fluid or

electrical systems, respectively. These changes in the continuous behavior increase the difficulties for accurate and timely online fault diagnosis. Our focus in this paper is on developing efficient model-based methodologies for online fault isolation and identification in complex hybrid systems.

Both the DX and the FDI communities have approached hybrid systems modeling and diagnosis during the last 20 years. They have used different modeling proposals [1; 2; 3], and have approached diagnosis either as hybrid state estimation [2] or as online state tracking [4; 5; 6], or a combination of both methods [7]. The main difficulties in any approach is to estimate the current state or set of states, and to diagnose that set of feasible states. Both tasks are computationally expensive or even unfeasible for complex systems. Several approaches have been proposed in the DX field to tackle these problems [4; 6].

In this work we have selected the hybrid system modeling based on Hybrid Bond Graphs (HBGs) [1; 6], together with consistency-based diagnosis using Possible Conflicts (PCs) [8]. HBGs are an extension of Bond Graphs (BG) [9], which models the discrete changes as ideal switching junctions that can be set to ON or OFF according to an automaton. In [10] we presented Hybrid Possible Conflicts (HPCs) as an extension of Possible Conflicts using HBGs to track hybrid systems behavior. Later, the HPCs approach was extended to integrate fault diagnosis of both parametric and discrete faults using HPCs [11] in a unique framework.

In order to achieve efficient fault identification, it is very important to determine the time of fault occurrence as accurately and quickly as possible. But there is a required trade-off between fast and reliable fault detection. In our approach we relied upon an statistical test to decide when a residual deviates from the current mode, and used this time to start the fault isolation and identification stages, however, the fault detection instant can be delayed from the fault occurrence time and this has some problems (e.g., that the fault identification process is delayed, or that we have to assume that we know the value of the state variables at the beginning of the identification process). In this work we propose to analyze the evolution of the residual signal using the CUSUM algorithm [12; 13] to find a more accurate estimation of the time of fault occurrence, both for potential new modes tracking and for parametric fault identification. Moreover, we extend our previous proposals for fault identification [14; 15] to cope with fault identification along a set of mode changes while performing the parameter identification.

The rest of the paper is organized as follows. Section 2 presents the case study used along the paper and introduces the Hybrid Bond Graph (HBG) modeling technique. Section 3 summarizes the Hybrid Possible Conflicts (HPCs) background, while section 4 explains the unified framework for both discrete and parametric faults. Section 5 introduces some concepts related to the CUSUM algorithm required in our approach. Section 6 explains our approach for fault identification. Section 7 introduces some results obtained applying our proposal on our case study. Finally, Section 8 draws some conclusions.

2 Case Study

The hybrid four-tank system in Figure 1 will be used to show some concepts and to present some results in this work. The system has an input flow which can be sent to tank 1, to tank 3 or to both tanks. Next to tank 1 there is tank 2, once the liquid in tank 1 reaches a level of h it starts to fill also tank 2. The lower part of the system has the same configuration, tank 4 is next to tank 3 connected by a pipe at a distance h above the base of the tanks.

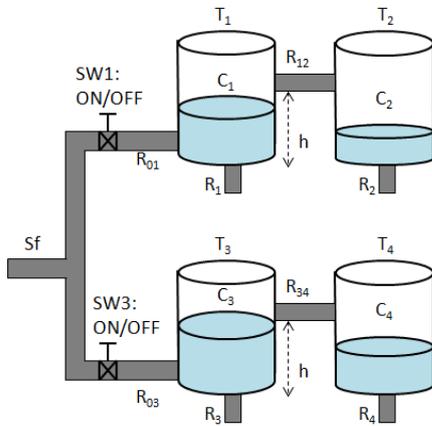


Figure 1: Schematics of the four-tank system

The methodology chosen to model the system in this work is Hybrid Bond Graph (HBG), which is an extension of Bond Graphs (BGs). BGs are defined as a domain-independent energy-based topological modeling language for physical systems [9]. Several types of primitive elements are used to build BGs: storage elements (capacitances, C , and inductances, I), dissipative elements (resistors, R) and elements to transform energy (transformers, TF , and gyrators, GY). There are also effort and flow sources (Se and Sf), which are used to define interactions between the system and the environment. Elements in a BG are connected by 0 or 1 junctions (representing ideal parallel or series connections between components). Each bond has associated two variables (effort and flow). The power is defined as $effort \times flow$ for each bond. The SCAP algorithm [16] is used to assign causality automatically to the BG.

To model hybrid systems using BGs we need to use some kind of connections which allow changes in their state. Hybrid Bond Graphs (HBGs) [1] extend BGs by including those connections. They are idealized switching junctions that allow mode changes in the system. If a switching junction is set to *ON*, it behaves as a regular junction. When it changes to *OFF*, all bonds incident on the junction are deactivated forcing 0 flow (or effort) for 1 (or 0) junctions.

A finite state machine *control specification* (CSPEC) implements those junctions. Transitions between the CSPEC states can be triggered by endogenous or exogenous variables, called guards. CSPECs capture controlled and autonomous changes as described in [17]. Figure 2 shows the HBG model of the four-tank system in Figure 1.

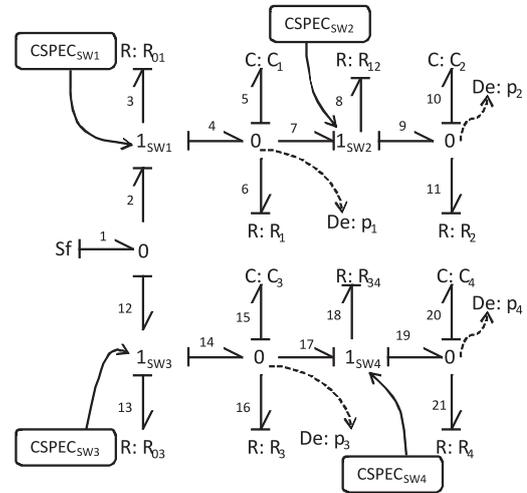


Figure 2: Bond graph model of the plant.

The system has four switching junctions: SW_1 , SW_2 , SW_3 and SW_4 . SW_1 and SW_3 are controlled *ON/OFF* transitions, while SW_2 and SW_4 are autonomous transitions. Both kinds of transitions are represented using a finite state machine. Figure 3 shows: a) the automaton associated with switching junction SW_1 and b) the automaton representing the autonomous transition in SW_2 . Since the system is symmetric, automata for SW_3 and SW_4 are equivalent to the ones shown in Figure 3.

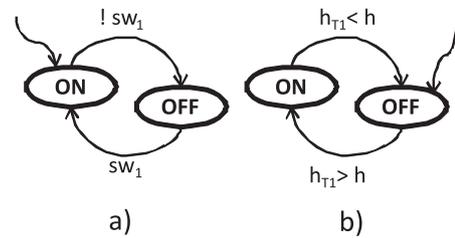


Figure 3: a) Automaton associated with the *ON/OFF* switching junction SW_1 ; b) Automaton representing the autonomous transition in SW_2 .

3 Hybrid Possible Conflicts background

Consistency-based diagnosis of continuous systems using Possible Conflicts (PCs) [8] is based upon a dependency-compilation technique from the DX community. PCs are computed offline, finding minimal structurally overdetermined subsets of equations with sufficient analytical redundancy to generate fault hypotheses from observed measurement deviations. Only structural and causal information about the system description is required. This information can be obtained from a set of algebraic and/or differential equations, or can be automatically derived from bond graph models [18; 19]. Once the set of PCs is found, they can be implemented as simulation, state-observers or gray-box

models for tracking online actual system behavior [20], or for online fault identification [14].

The PCs approach has been recently extended to cope with hybrid system dynamics, and the set of PCs for hybrid systems were called Hybrid Possible Conflicts (HPCs) [10]. HPCs rely upon the Hybrid Bond-Graph modeling formalism [1], whose main advantage is that the set of possible modes in the system do not need to be enumerated. Moreover, HBGs are capable to track online hybrid system behavior, performing online causality reassignment in the system model by means of the HSCAP algorithm [17]. Using HPCs we make even more efficient the HSCAP algorithm, because causality needs only to be revised within the subsystem defined for each HPC, and these changes are local to the switching junction affected by the mode change.

For the four-tank system we have found four HPCs. Each one of them estimates one of the measured variables (p_1 , p_2 , p_3 , or p_4). Figure 4 shows the BG fragments of these four HPCs. In this example, the four HPCs were computed assuming that all switching junctions are set to *ON*.

As mentioned before, when any of these junctions is switched to *OFF*, causality in the system needs to be re-assigned, but the HPCs generation process does not need to be restarted again [10]. The decomposition of a hybrid system model obtained from HPCs is unique, and after a mode change some portions of some HPCs can disappear (or even the entire HPC), but no additional HPC appears. It is proved in [10] that once PCs of the system have been generated considering all switching junctions set to *ON* mode, turning a switch from *ON* to *OFF* or viceversa, no genuine new HPCs will ever appear.

Regarding fault profiles, our current proposal works with single fault, and abrupt fault assumptions. Abrupt faults are modeled as an instantaneous change in a parameter, whose magnitude does not change afterwards (can be modeled as a step function).

Regarding parametric faults, fault isolation is performed by means of the Reduced Qualitative Fault Signature Matrix (RQFSM). Table 1 shows the RQFSM for the mode where each switch is set to *ON*. For a given mode, the RQFSM can be computed online from the TCG associated to an HPC [1]. In this table there is a row for each fault considered. And there is a column for each HPC. The entry in the table represent the Qualitative Fault Signature of the fault in the HPC residual, as computed in TRANSCEND [1]. The “reduced” tag means that the Qualitative Fault Signature is computed within the subsystem delimited by a HPC, and not for the whole set of measurements [18]. Once fault detection is performed, we can use this information to reject those faults whose residual evolution does not match the qualitative signatures in this table.

We also consider discrete faults, i.e. faults in discrete actuators, as commanded mode switches which do not perform the correct action. In our case study, there are four faulty situations to be considered, where SW_i denotes the switching junction i of the system.

1. $SW_i = 11$: SW_i stuck ON (1).
2. $SW_i = 00$: SW_i stuck OFF (0).
3. $SW_i = 01$: Autonomous switch ON (SW_i is OFF (0) and it switches to ON itself (1)).
4. $SW_i = 10$: Autonomous switch OFF (SW_i is ON (1) and it switches to OFF itself (0)).

Table 1: Reduced Qualitative Fault Signature Matrix.

	<i>HPC1</i>	<i>HPC2</i>	<i>HPC3</i>	<i>HPC4</i>
C_1^+	-+			
C_2^+		--		
C_3^+			-+	
C_4^+				--
R_{01}^+	0-		0+	
R_{03}^+	0+		0-	
R_1^+	0+			
R_2^+		0+		
R_3^+			0+	
R_4^+				0+
R_{12}^+	0-	0-		
R_{34}^+			0+	0-

The relation between the HPCs and their related switching junctions can be seen in Table 2, which is called Hybrid Fault Signature Matrix (HFSM). This information can be used in the unified framework for discrete and parametric fault isolation and identification [11].

Table 2: Hybrid Fault Signature Matrix (HFSM) showing the relations between switching junctions and each HPC.

	<i>HPC1</i>	<i>HPC2</i>	<i>HPC3</i>	<i>HPC4</i>
1_{SW_1}	1		1	
1_{SW_2}	1	1		
1_{SW_3}	1		1	
1_{SW_4}			1	1

Discrete faults usually introduce high non-linearities in the system outputs, that should be easily detected if magnitudes related to the failing switch were measured, generating almost instantaneous detection for discrete faults. In this case, exoneration could be applied. But even if those measurements are not available we can still use the qualitative signature of the effects of the discrete faults in the HPC residuals. With this information we can build the so-called Hybrid Qualitative Fault Signature Matrix (HQFSM) that can also be used for exoneration purposes in the fault isolation stage. In our system we can build the following HQFSM for *HPC1* and *HPC3*, which are linked to commanded switches SW_1 and SW_3 , which are the potential source of discrete faults in our system. We do not show SW_2 and SW_4 in the table since they introduce hybrid dynamics in the system, but they can not be the source of a discrete fault.

Table 3: Hybrid Qualitative Fault Signature Matrix.

	<i>HPC1</i>	<i>HPC3</i>
$1_{SW_1}(11)$	+	-
$1_{SW_1}(00)$	-	+
$1_{SW_1}(01)$	+	-
$1_{SW_1}(10)$	-	+
$1_{SW_3}(11)$	-	+
$1_{SW_3}(00)$	+	-
$1_{SW_3}(01)$	-	+
$1_{SW_3}(10)$	+	-

Next section presents our diagnosis framework for hybrid systems using HPCs.

4 Hybrid Systems Diagnosis using HPCs

As we mentioned before, tracking of hybrid systems can be performed using Hybrid PCs [10]. Initially, the set of HPCs is built assuming all switching junctions are set to *ON*.

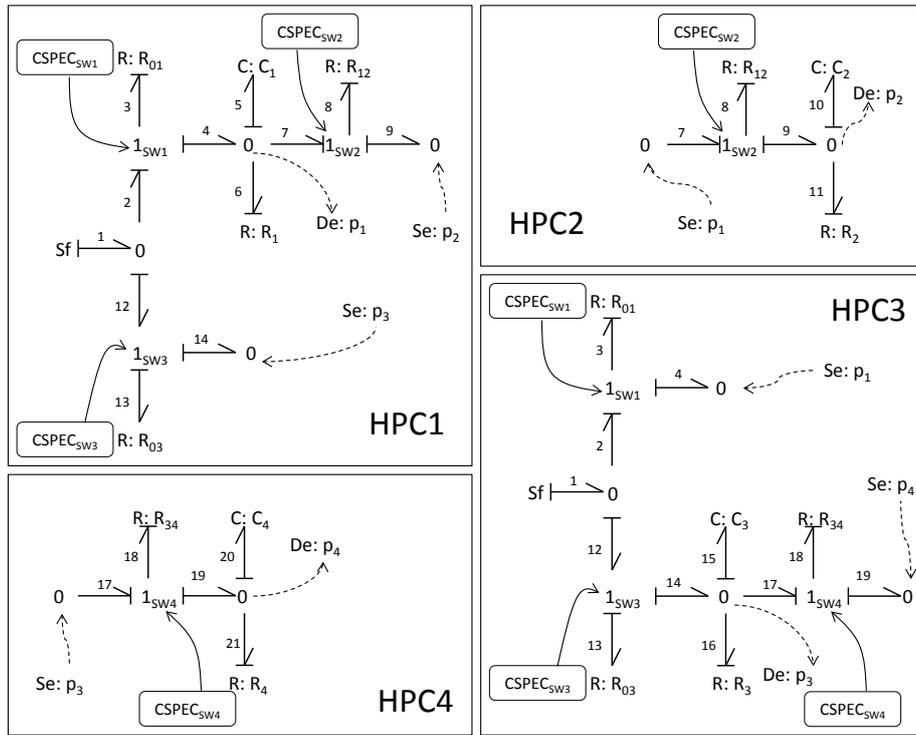


Figure 4: Bond graphs of the four PCs found for the four-tank system.

Afterwards, the set of models for the HPCs for the actual mode are efficiently built, and they start tracking the system. Whenever a mode change, commanded or autonomous, is detected, a new set of models for the HPCs is computed on-line.

In case a fault occurs, one or more HPC residuals will trigger. Significant deviations in the residuals are found using the statistical Z-test. Based on the activated residuals for the set of HPCs in the current mode, the structural information in the HQFSM (Table 3), and the RQFSM (Table 1), we build the current set of fault candidates. This set can contain both discrete and parametric faults. Since discrete faults generally have a bigger and potentially more dangerous influence in the system behavior, in our framework we consider discrete faults as preferred candidates before considering the parametric ones. If there is no discrete fault as candidate, then we directly go to the fault identification as described in Section 6.

At this point we run the CUSUM algorithm (described in Section 5) to approximately determine the time of fault occurrence. Once this is done, we create a new simulation model using the HPCs, and starting at the fault time determined by the CUSUM, we begin tracking the system behavior in each one of the hypothesized mode changes (the HQFSM and the qualitative value of the HPC residuals are used to reject those modes that are inconsistent with expected deviations in the HQFSM). If the hypothesized mode is the correct one, the residual for that mode will go to zero after a relatively small period of time (this is possible, as we will show later, thanks to the accurate estimation of the fault time provided by the CUSUM). If the hypothesized mode is not correct, the residual will keep deviating from zero, and after an empirically determined time window without converging, the discrete fault candidate will be discarded. If

only one mode has the residual close to zero, this is the new system mode.

If the residual for each hypothesized new mode does not converge to zero, discrete faults (as mode changes) are discarded and we focus on parametric faults, starting the identification stage. As mentioned before, qualitative fault signatures in the RQFSM can be used to reject those parametric faults non consistent with current observations thus focusing even further the fault identification stage.

Finally, once the set of parametric fault candidates is refined through the RQFSM, we perform fault identification for the set of remaining parametric fault candidates. Fault identification is done with hybrid parameter estimators, which are presented in Section 6.

5 Time of Fault Estimation using CUSUM

In the previous section we have presented our fault isolation approach of discrete faults by hypothesizing the faults compatible with the Hybrid Qualitative Fault Signature Matrix and filtering out those faults whose models do not converge. Divergence of non-current models is usually easy to check when we are dealing with discrete faults. However, the convergence of the current model may be slow if initial values of the state variables of the model are not known or our initial guess is far from the actual value. We are assuming that we are able to track the system dynamic before the occurrence of a fault. In other words, we are assuming that we know -or we are able to estimate- the state variables before the time of fault occurrence. Hence, in order to speed up the convergence of the current model, it is important to have a good estimation of that time.

The cumulative sum algorithm, CUSUM, introduced by [12] and discussed in detail in [13] and elsewhere, is an op-

timal fault detection algorithm that can also provide a estimation of the time of fault occurrence t_0 , as we will detail later. Nevertheless, it makes the strong assumption that the signal we are tracking changes its mean value from a constant initial mean μ_0 to a final constant mean μ_1 .

On the other hand, the Z-test [21] is a sub-optimal fault detection algorithm compared to CUSUM, but it makes no assumptions concerning the properties of the new mean value. Particularly, it does not require this to be constant.

In order to have a robust fault detection mechanism and a good approximation of the fault time, we have opted for combining both tests. We use Z-test to perform fault detection and, afterwards, we estimate the fault time using CUSUM.

CUSUM was designed to detect abrupt changes in the mean of stochastic signals. In the simple case of a Gaussian residual, $res(i)$, of constant variance σ^2 , constant and known initial mean μ_0 and constant and known final mean μ_1 , the decision signal, S_k , is $S_k = \sum_{i=1}^k s_i =$

$\sum_{i=1}^k \frac{\mu_1 - \mu_0}{\sigma^2} (res(i) - \frac{\mu_0 + \mu_1}{2})$. Hence, for a window of N samples with a change in mean at $1 \leq t_0 \leq N$, S_k decreases at the constant rate $\mu = \frac{\mu_1 - \mu_0}{2}$ for $k < t_0$ and increases by μ for $t_0 \leq k$. It can be shown [13] that the change time t_0 can be estimated as $\hat{t}_0 = \arg \min_k S_k$.

When μ_1 is unknown, it can be set to the residual corresponding to the smallest fault to be detected, typically some units of the residual noise deviation, σ . This can be done without increasing the fault positive alarm rate because we use Z-test to perform fault detection, and we only use this CUSUM variant to estimate the time of fault occurrence, t_0 . We have also tried estimating μ as the empirical mean of the residual, with similar results. In all the cases we have tested, the estimated time of fault occurrence, \hat{t}_0 , computed by CUSUM, is smaller than the detection time provided by Z-test.

6 Fault Identification with HPCs

Once all the discrete fault candidates have been discarded, we have to do fault identification for the set of isolated parametric faults. In previous work [14] we proposed to use minimal parameter estimators computed from PCs to generate parameterized estimators. However, that approach is not applicable for hybrid systems fault identification since we can have mode changes during the identification process. As a solution, we propose an extension of our minimal parameterized estimators which are computed directly from HPCs, thus being able to handle mode changes during the identification process.

The fault identification process is done by the following steps: (i) model decomposition by offline computation of the set of HPCs from the hybrid bond graph model; (ii) offline computation and selection of the better hybrid estimator for each fault candidate; (iii) after the fault isolation process, online quantitative parameter estimation procedure over the hybrid estimators related with the set of isolated fault candidates; and (iv) decision procedure to select the faulty candidate.

Using HPCs we can derive the structure of a hybrid parameterized estimator, e_{hpc_k} , for a hybrid system. The parameterized estimator e_{hpc_k} can be used as a hybrid estimator as stated in the following proposition:

Proposition 1. A HPC, HPC_k , along with its set of input variables, u_{hpc_k} , the commanded signals of the switching junctions, sw_{hpc_k} , and initial value of the parameter to identify, θ_f , can be used as a parameter estimator using $\hat{y}_{hpc_k} = e_{hpc_k}(u_{hpc_k}, \theta_f, sw_{hpc_k}(t))$, where the measured variable estimated by the HPC, y_{hpc_k} , is solved in terms of the remaining measured variables.

Each estimator is uniquely related to one HPC, hence it contains minimal redundancy required for parameter estimation. In this case, each HPC has an executable model that can be used for simulation purposes. For the four-tank system we have obtained four hybrid parameter estimators shown in table 4, one for each HPC.

Estimator	Related PC	Parameters	Inputs	Output
e_1	HPC_1	$R_{01}, R_{03}, R_{12}, R_1, C_1$	S_f, p_2, p_3	p_1
e_2	HPC_2	R_{12}, R_2, C_2	p_1	p_2
e_3	HPC_3	$R_{01}, R_{03}, R_{34}, R_3, C_3$	S_f, p_1, p_4	p_3
e_4	HPC_4	R_{34}, R_4, C_4	p_3	p_4

Table 4: Hybrid parameter estimators found for the four-tank system, and their related HPCs.

The basic idea is to use the estimator e_{hpc_k} to compute estimations for \hat{y}_{hpc_k} with different values of the parameter θ_f , so that we can find a value of the parameter that minimizes the least squares (LS) error between the estimation \hat{y}_{hpc_k} and the measured value y_{hpc_k} .

Fig. 5 shows the parameter estimation process using the hybrid estimators. A parameterized estimator, e_{hpc_k} , uses the inputs of the system, u_{hpc_k} , and a parameter value, θ_f , to generate an estimation of the output, \hat{y}_{hpc_k} . This estimated output is compared against the observed output, y_{hpc_k} , by the quadratic error calculator block. This block computes the quadratic error between \hat{y}_{hpc_k} and y_{hpc_k} for the fault candidate f , E_f^2 . Then, the iteration engine block, that contains a nonlinear optimization algorithm, finds the minimum of the error surface $E_f^2(\theta_f)$, by iteratively invoking the estimator with different parameter values. The value of the parameter and its minimum LS error will be the output of the parameter estimation block (and the input for the decision procedure block).

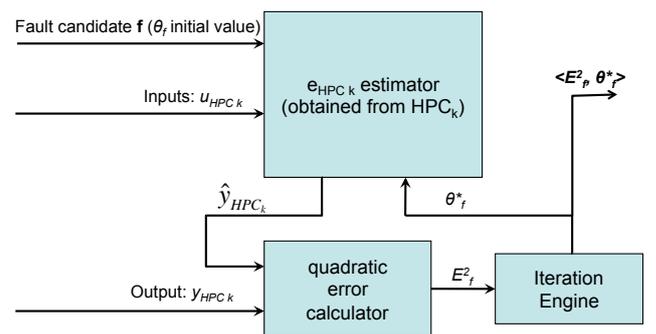


Figure 5: Parameter estimation using the hybrid estimators from HPCs.

7 Results

To test the validity of the approach, we implemented the four hybrid HPCs for the four-tank system, with its corresponding estimators, and run different simulation experiments.

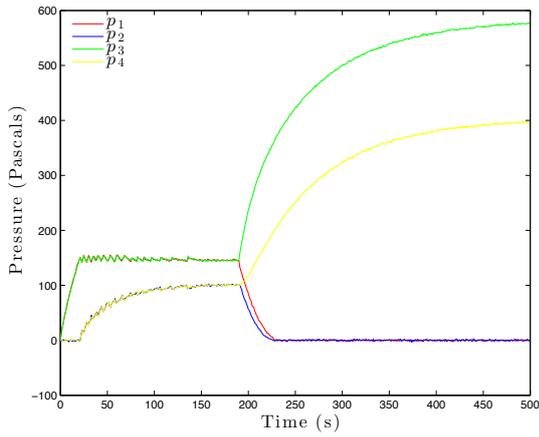


Figure 6: Measured pressures in the four tanks when a fault in SW_1 is introduced at $t = 190$ s.

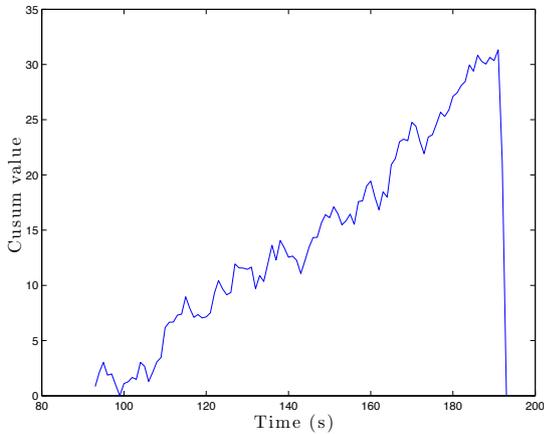


Figure 7: CUSUM output for a fault in SW_1 .

In the first experiment, we assume that the water tanks are initially empty, and start to fill in at constant rate. Hence, the initial configuration of the system is SW_1 and SW_3 set to *ON*, and SW_2 and SW_4 set to *OFF*. Tanks 1 and 3 start to fill in, and approximately at time 20 s level in both tanks reach the height of the connecting pipes and tanks 2 and 4 start to fill in. At time 190 s, a fault occurs in the controlled junction SW_1 , which switches off (see Fig. 6 for the measured pressures in the four tanks for this experiment).

Four seconds after the fault is introduced, at $t = 194$ s, both HPC_1 and HPC_3 trigger, and consequently both SW_1 or SW_3 are initially considered as discrete fault candidates. At this point, the CUSUM algorithm is run, determining that the fault has occurred at $t = 191$ s. In this case study we use a CUSUM window of size 100. Figure 7 shows the output of the CUSUM algorithm where the absolute maximum represents the approximate time (due to noise in the system) of fault occurrence.

Once the point of fault occurrence has been determined at $t = 191$ s, the diagnosis framework takes the values of the simulation at such time instant and launches two parallel diagnosis experiments, one for each hypothesized fault candidate, i.e., $SW_1(10)$ and $SW_3(10)$. Figs. 8 and 9 show the

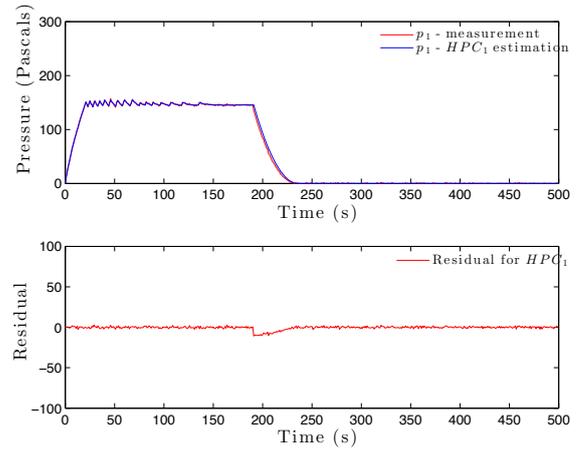


Figure 8: Estimation and residual for HPC_1 (using CUSUM) when a fault in SW_1 occurs and the hypothesized fault is SW_1 .

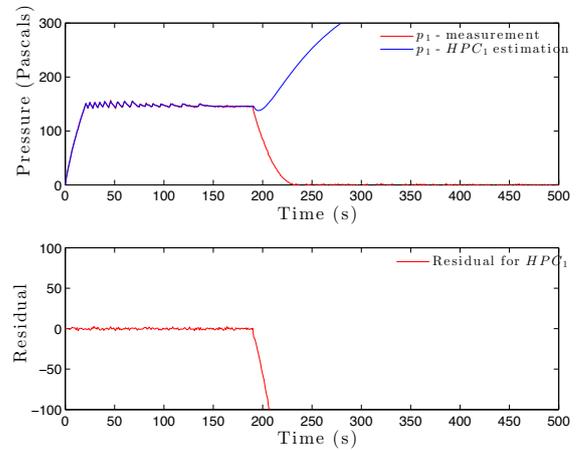


Figure 9: Estimation and residual for HPC_1 (using CUSUM) when a fault in SW_1 occurs and the hypothesized fault is SW_3 .

estimation and the residual for HPC_1 when the hypothesized faults are $SW_1(10)$ and $SW_3(10)$, respectively (we do not show the result for HPC_3 since are similar to the results obtained for HPC_1). Looking at the results, it is obvious that the residual converges to zero when a fault in $SW_1(10)$ is hypothesized, while the residual when $SW_3(10)$ is hypothesized does not converge. Hence, $SW_1(10)$ is confirmed as the fault. This confirmation is done by continuously analyzing residual signals with the Z-test. Please note that, since the CUSUM algorithm gives a good approximation of the point of failure, the residual is able to converge very quickly when the true fault is hypothesized. For comparison purposes, Fig. 10 shows the estimation and residual for HPC_1 when CUSUM is not used to re-initialize the simulation (for the hypothesized fault SW_1). By comparing this figure with Fig. 8 it is clear that using CUSUM allows the HPC to converge faster.

As a second diagnosis experiment, we start off from the same situation of the previous experiment, but in this case, we introduce a small parametric fault and after a short while,

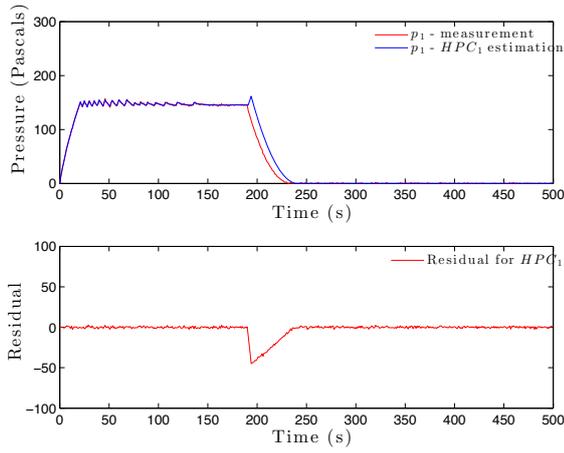


Figure 10: Estimation and residual for HPC_1 (without using CUSUM to re-initialize the simulation) when a fault in SW_1 occurs and the hypothesized fault is SW_1 .

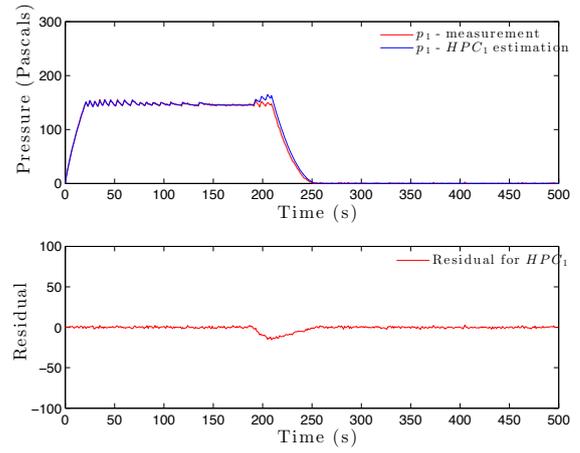


Figure 12: Estimation and residual for HPC_1 when a fault in R_{01} occurs and then SW_1 is set to OFF mode.

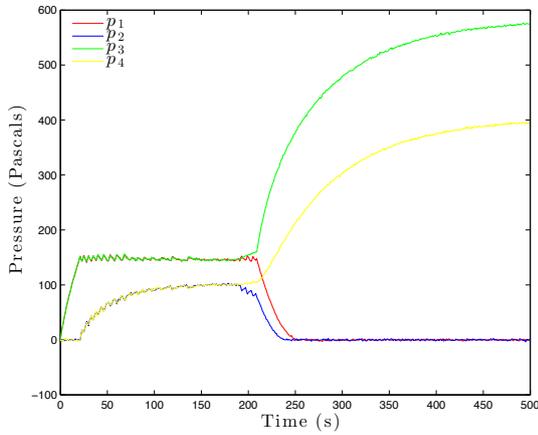


Figure 11: Measured pressures in the four tanks when a fault in R_{01} is introduced at $t = 190$ s and the switching junction SW_1 is turned off at $t = 210$ s.

a discrete change. Specifically, a 20% blockage in the input pipe of tank 1, R_{01} , is introduced at $t = 190$ s, and then SW_1 is commanded to switch OFF at $t = 210$ s (Fig. 11 shows the measured pressures in the four tanks for this experiment).

For this experiment, both HPC_1 and HPC_3 trigger at $t = 198$ s (as an example, see Fig. 12 with the estimation and residual for HPC_1), and consequently both $SW_1(10)$ and $SW_3(10)$ are initially considered as discrete fault candidates. However, in this scenario, after running the CUSUM (see Fig. 13 for the CUSUM output), which estimated the fault time at $t = 191$ s, and the diagnosis experiments for both fault candidates, none of the residuals was able to converge within a reasonable, empirically determined, amount of time, thus concluding that a parametric fault has occurred. At this point, the fault identification process is triggered for R_{01} , which is the only parametric fault candidates (R_{03} is discarded due to the qualitative sign in the residuals). The estimated value for parameter R_{01} was 0.1937, i.e., a 19.37% blockage in the pipe. Please note that the estimator

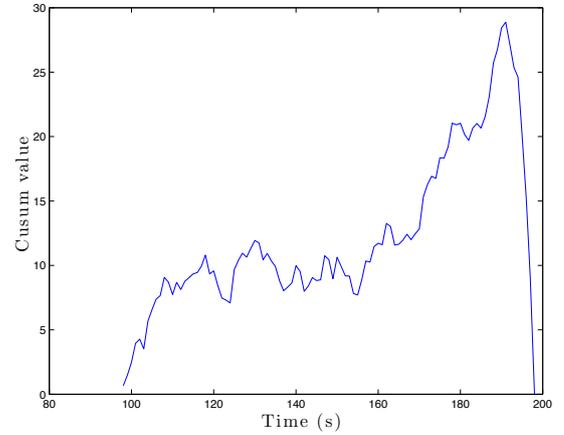


Figure 13: CUSUM output for a fault in R_{01} .

used a total of 60 seconds of data starting from $t = 191$ s, hence, the estimator was capable of correctly estimating the value of the faulty parameter even if the system transitions from one mode to another during the estimation process.

We run several experiments with different mode configurations and different faults, varying the size, time of fault occurrence (in some of them by introducing faults immediately after the mode change). Results for all these situations were equivalent to the examples shown in this section.

8 Conclusions

In this work we have presented an approach for hybrid systems fault identification using Hybrid Possible Conflicts. Using HBGs we can generate minimal estimators that can be used for fault identification just considering the possible mode changes within the estimators. Additionally, we have proposed the integration of the CUSUM algorithm to accurately determine the time of fault occurrence. A more accurate estimation of the fault instant allows to quickly isolate discrete faults, and to obtain a better approximation of the values of the state variables, which are needed as initial values for the fault identification.

Diagnosis results using a four-tank system showed that the proposed approach can be successfully used for fault identification of hybrid systems.

In future work, we will test the approach in more complex systems with real data, and will propose a distributed approach for hybrid systems fault diagnosis.

Acknowledgments

This work has been funded by the Spanish MINECO DPI2013-45414-R grant.

References

- [1] P. Mosterman and G. Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. on Sys., Man, and Cyber. Part A*, 29(6):554–565, 1999.
- [2] M.W. Hofbaur and B.C. Williams. Hybrid estimation of complex systems. *IEEE Trans. on Sys., Man, and Cyber. Part B*, 34(5):2178–2191, oct. 2004.
- [3] E. Benazera and L. Travé-Massuyès. Set-theoretic estimation of hybrid system configurations. *IEEE Trans. on Sys. Man Cyber. Part B*, 39:1277–1291, October 2009.
- [4] S. Narasimhan and L. Brownston. Hyde - a general framework for stochastic and hybrid model-based diagnosis. In *Proc. of the 18th Int. WS on Pples. of Diagnosis, DX07*, pages 186–193, Nashville, TN, USA, May 29-31 2007.
- [5] Mehdi Bayouhd, Louise Travé-Massuyès, and Xavier Olive. Fault detection and diagnosis; hybrid systems modeling and control; discrete event systems modeling and control. In *Proc. of the Int. Conference on Control, Automation and Systems, ICCAS08*, pages 7265–7270, Seoul, Korea, October 2008.
- [6] S. Narasimhan and G. Biswas. Model-Based Diagnosis of Hybrid Systems. *IEEE Trans. on Sys., Man and Cyber., Part A*, 37(3):348–361, May 2007.
- [7] Th. Rienmüller, M. Bayouhd, M.W. Hofbaur, and L. Travé-Massuyès. Hybrid Estimation through Synergic Mode-Set Focusing. In *Proc. of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS09*, pages 1480–1485, Barcelona, Spain, 2009.
- [8] B. Pulido and C. Alonso-González. Possible Conflicts: a compilation technique for consistency-based diagnosis. *IEEE Trans. on Sys., Man, and Cyber. Part B: Cybernetics*, 34(5):2192–2206, Octubre 2004.
- [9] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics: Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2006.
- [10] A. Bregon, C. Alonso, G. Biswas, B. Pulido, and N. Moya. Fault diagnosis in hybrid systems using possible conflicts. In *Proc. of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS12*, Mexico City, Mexico, 2012.
- [11] N. Moya, A. Bregon, CJ Alonso-González, and B. Pulido. A Common Framework for Fault Diagnosis of Parametric and Discrete Faults Using Possible Conflicts. In *Advances in Artificial Intelligence, CAEPIA 2013*, volume 8109 of *Lecture Notes in Computer Science*, pages 239–249. Springer-Verlag Berlin, 2013.
- [12] E.S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.
- [13] M. Basseville and I.V. Nikiforov. *Detection of Abrupt Changes: Theory and Applications*. Prentice Hall, 1993.
- [14] A. Bregon, G. Biswas, and B. Pulido. A Decomposition Method for Nonlinear Parameter Estimation in TRANSCEND. *IEEE Trans. Syst. Man, Cyber. Part A*, 42(3):751–763, 2012.
- [15] N. Moya, B. Pulido, CJ Alonso-González, A. Bregon, and D. Rubio. Automatic Generation of Dynamic Bayesian Networks for Hybrid Systems Fault Diagnosis. In *Proc. of the 23rd Int. WS on Pples. of Diagnosis, DX12*, Great Malvern, UK, Jul-Aug 2012.
- [16] D.C. Karnopp, R.C. Rosenberg, and D.L. Margolis. *System Dynamics, A Unified Approach*. 3rd ed., John Wiley & Sons, 2000.
- [17] I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Efficient simulation of hybrid systems: A hybrid bond graph approach. *SIMULATION: Transactions of the Society for Modeling and Simulation International*, (6):467–498, June 2011.
- [18] A. Bregon, B. Pulido, G. Biswas, and X. Koutsoukos. Generating Possible Conflicts from Bond Graphs Using Temporal Causal Graphs. In *Proc. of the 23rd European Conference on Modelling and Simulation, ECMS09*, pages 675–682, Madrid, Spain, 2009.
- [19] A. Bregon, G. Biswas, B. Pulido, C. Alonso-González, and H. Khorasgani. A Common Framework for Compilation Techniques Applied to Diagnosis of Linear Dynamic Systems. *IEEE Trans. on Sys., Man, and Cyber.: Systems*, 44(7):863–873, 2013.
- [20] A. Bregon, C. Alonso-González, and B. Pulido. Integration of simulation and state observers for on line fault detection of nonlinear continuous systems. *IEEE Trans. on Syst., Man, and Cyb.: Systems*, 44(12):1553–1568, 2014.
- [21] G. Biswas, G. Simon, N. Mahadevan, S. Narasimhan, J. Ramirez, and G. Karsai. A robust method for hybrid diagnosis of complex systems. In *Proceeding of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS 2003*, pages 1125–1130, Washington D.C., USA, June 2003.

State estimation and fault detection using box particle filtering with stochastic measurements

Joaquim Blesa¹, Françoise Le Gall², Carine Jaubert^{2,3} and Louise Travé-Massuyès²

¹Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas, 4-6, 08028 Barcelona, Spain

e-mail: joaquim.bleesa@upc.edu

²CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

Univ de Toulouse, LAAS, F-31400 Toulouse, France

e-mail: legall,cjaubert,louise@laas.fr

³Univ de Toulouse, UPS, LAAS, F-31400 Toulouse

Abstract

In this paper, we propose a box particle filtering algorithm for state estimation in nonlinear systems whose model assumes two types of uncertainties: stochastic noise in the measurements and bounded errors affecting the system dynamics. These assumptions respond to situations frequently encountered in practice. The proposed method includes a new way to weight the box particles as well as a new resampling procedure based on repartitioning the box enclosing the updated state. The proposed box particle filtering algorithm is applied in a fault detection schema illustrated by a sensor network target tracking example.

1 Introduction

For various engineering applications, system state estimation plays a crucial role. Kalman filtering (KF) has been widely used in the case of stochastic linear systems. The Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are KF's extensions for nonlinear systems. These methods assume unimodal, Gaussian distributions. On the other hand, Particle Filtering (PF) is a sequential Monte Carlo Bayesian estimator which can be used in the case of non-Gaussian noise distributions. Particles are punctual states associated with weights whose likelihoods are defined by a statistical model of the observation error. The efficiency and accuracy of PF depend on the number of particles used in the estimation and propagation at each iteration. If the number of required particles is too large, a real implementation is unsuitable and this is the main drawback of PF. Several methods have been proposed to overcome these shortcomings, mainly based on variants of the resampling stage or different ways to weight the particles ([1]).

Recently, a new approach based on *box particles* was proposed by [2; 3]. The Box Particle Filter handles box states and bounded errors. It uses interval analysis in the state update stage and constraint satisfaction techniques to perform measurement update. The set of box particles is interpreted as a mixture of uniform pdf's [4]. Using box particles has been shown to control quite efficiently the number of required particles, hence reducing the computational cost and providing good results in several experiments.

In this paper, we take into account the box particle filtering ideas but consider that measurements are tainted by

stochastic noise instead of bounded noise. The errors affecting the system dynamics are kept bounded because this type uncertainty really corresponds to many practical situations, for example tolerances on parameter values. Combining these two types of uncertainties following the seminal ideas of [5] and [6] within a particle filter schema is the main issue driving the paper. This issue is different from the one addressed in [7] in which the focus is put on Bernoulli filters able to deal with data association uncertainty. The proposed method includes a new way to weight the box particles as well as a new resampling procedure based on repartitioning the box enclosing the updated state.

The paper is organized as follows. Section 2 describes the problem formulation. A summary of the Bayesian filtering is presented and the box-particle approach is introduced. The main steps of this approach are developed in section 3. Section 4 and 5 are devoted to the repartitioning of the boxes and the computation of the weight of the box particles in order to control the number of boxes. In section 6 the box particle filter is used for state estimation and fault detection; the results obtained with the proposed method for a target tracking in a sensor network are presented in section 7. Conclusion and future work are overviewed in the last section.

2 Problem formulation

We consider nonlinear dynamic systems represented by discrete time state-space models relating the state $\mathbf{x}(k)$ to the measured variables $\mathbf{y}(k)$

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k)) \quad (1)$$

$$\mathbf{y}(k) = h(\mathbf{x}(k)) + \mathbf{e}(k), k = 0, 1, \dots \quad (2)$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$ and $h : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ are nonlinear functions, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ is the system input, $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ is the system output, $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the state-space vector, $\mathbf{e}(k) \in \mathbb{R}^{n_y}$ is a stochastic additive error that includes the measurement noise and discretization error and is specified by its known pdf p_e . $\mathbf{v}(k) \in \mathbb{R}^{n_x}$ is the process noise.

In this work the process noise is assumed bounded $|v_i(k)| \leq \sigma_i$ with $i = 1, \dots, n_x$, i.e $p_v \sim \mathcal{U}(\mathcal{V})$, where $\mathcal{V} = [-\sigma_1, \sigma_1] \times \dots \times [-\sigma_{n_x}, \sigma_{n_x}]$.

2.1 Bayesian filtering

Given a vector of available measurements at instant k : $\mathbf{Y}(k) = \{\mathbf{y}(i), i = 1, \dots, k\}$, $\mathbf{Y}(0) = \mathbf{y}(0)$, the Bayesian

solution to compute the posterior distribution $p(\mathbf{x}(k)|\mathbf{Y}(k))$ of the state vector at instant $k + 1$, given past observations $\mathbf{Y}(k)$ is given by (Gustafsson 2002):

$$p(\mathbf{x}(k+1)|\mathbf{Y}(k)) = \int_{\mathbb{R}^{n_x}} p(\mathbf{x}(k+1)|\mathbf{x}(k))p(\mathbf{x}(k)|\mathbf{Y}(k))dx(k) \quad (3)$$

where the posterior distribution $p(\mathbf{x}(k)|\mathbf{Y}(k))$ can be computed by

$$p(\mathbf{x}(k)|\mathbf{Y}(k)) = \frac{1}{\alpha(k)}p(\mathbf{y}(k)|\mathbf{x}(k))p(\mathbf{x}(k)|\mathbf{Y}(k-1)) \quad (4)$$

where $\alpha(k)$ is a normalization constant, $p(\mathbf{y}(k)|\mathbf{x}(k))$ is the likelihood function that can be computed from (2) as:

$$p(\mathbf{y}(k)|\mathbf{x}(k)) = p_e(\mathbf{y}(k) - h(\mathbf{x}(k))) \quad (5)$$

and $p(\mathbf{x}(k)|\mathbf{Y}(k-1))$ is the prior distribution.

Equations (5), (4) and (3) can be computed recursively given the initial value of $p(\mathbf{x}(k)|\mathbf{Y}(k-1))$ for $k = 0$ denoted as $p(\mathbf{x}(0))$ that represents the prior knowledge about the initial state.

2.2 Objective

Considering the assumptions of our problem, we adopt a particle filtering schema which is well-known for solving numerically complex dynamic estimation problems involving nonlinearities. However, we propose to use box particles and to base our method on the interval framework. Box particle filters have been demonstrated efficient, in particular to reduce the number of particles that must be considered to reach a reasonable level of approximation [2].

Let's consider the current state estimate $\mathcal{X}(k)$ as a set, denoted by $\{\mathcal{X}(k)\}$, that is approximated by N_k disjoint boxes

$$[\mathbf{x}(k)]^i \quad i = 1, \dots, N_k \quad (6)$$

where $[\mathbf{x}(k)]^i = [\underline{\mathbf{x}(k)}^i, \overline{\mathbf{x}(k)}^i]$, with $\underline{\mathbf{x}(k)}^i, \overline{\mathbf{x}(k)}^i \in \mathbb{R}^{n_x}$. The width of every box is smaller or equal to a given accuracy for every component, i.e

$$\overline{x_j(k)}^i - \underline{x_j(k)}^i \leq \delta_j \quad i = 1, \dots, N_k, \quad j = 1, \dots, n_x \quad (7)$$

where δ_j is the predetermined minimum accuracy for every component j .

Moreover, every box $[\mathbf{x}(k)]^i$ is given a prior probability denoted as

$$P([\mathbf{x}(k)]^i|\mathbf{Y}(k-1)) \quad i = 1, \dots, N_k \quad (8)$$

with

$$\sum_{i=1}^{N_k} P([\mathbf{x}(k)]^i|\mathbf{Y}(k-1)) \geq \gamma \quad (9)$$

where $\gamma \in [0, 1]$ is a confidence threshold.

Then, given a new output measurement $\mathbf{y}(k)$, the problem that we consider in this paper is:

- to compute the state estimate $\mathcal{X}(k+1)$,
- to decide about the number N_{k+1} of disjoint boxes of the approximation of $\mathcal{X}(k+1)$, each with accuracy smaller or equal to δ_j ,

- to provide the prior probabilities associated to the particles of the new state estimation set

$$P([\mathbf{x}(k+1)]^i|\mathbf{Y}(k)) \quad i = 1, \dots, N_{k+1} \quad (10)$$

3 Interval Bayesian formulation

This section deals with the evaluation of the Bayesian solution of the state estimation problem considering bounded state boxes (6).

3.1 Measurement update

Whereas each particle is defined as a box by (6), the measurement is tainted with stochastic uncertainty defined by the pdf p_e . The weight $w(k)^i$ associated to a box particle is updated by the posterior probability $P([\mathbf{x}(k)]^i|\mathbf{Y}(k))$:

$$\begin{aligned} w(k)^i &= \frac{1}{\Lambda(k)}P([\mathbf{x}(k)]^i|\mathbf{Y}(k-1))p_e(\mathbf{y}(k) - h([\mathbf{x}(k)]^i)) \\ &= \frac{1}{\Lambda(k)}P([\mathbf{x}(k)]^i|\mathbf{Y}(k-1)) \int_{x(k) \in [\mathbf{x}(k)]^i} p_e(\mathbf{y}(k) - h(x(k))) dx(k) \end{aligned} \quad (11)$$

$$i = 1, \dots, N_k$$

where the normalization constant $\Lambda(k)$ is given by

$$\Lambda(k) = \sum_{i=1}^{N_k} P([\mathbf{x}(k)]^i|\mathbf{Y}(k-1)) \int_{x(k) \in [\mathbf{x}(k)]^i} p_e(\mathbf{y}(k) - h(x(k))) dx(k) \quad (12)$$

then

$$\sum_{i=1}^{N_k} w(k)^i = 1 \quad (13)$$

The deduction of the measurement update equation (11) from the particle filtering update equation (4) is detailed in the Appendix for $n_x = 1$, without the loss of generality. The principle of the proof is that the point particles are grouped into particle groups inside boxes, then the posterior probability of a box can be approximated by the sum of posterior probabilities of the point particles when the number of these particles tends to infinity.

3.2 State update

This step is similar to the state update state as in [2] and [3]. Hence, we have:

$$p(\mathbf{x}(k+1)|\mathbf{Y}(k)) \approx \sum_{i=1}^{N_k} w(k)^i \mathcal{U}_{[f]([\mathbf{x}(k)]^i, \mathbf{u}(k), [\mathbf{v}(k)])} \quad (14)$$

The interval boxes $[\mathbf{x}(k+1)|\mathbf{x}(k)]^i$ are computed from (1) using interval analysis as follows,

$$[\mathbf{x}(k+1)|\mathbf{x}(k)]^i \approx [f]([\mathbf{x}(k)]^i, \mathbf{u}(k), [\mathbf{v}(k)]) \quad (15)$$

The update interval boxes inherit the weights $w(k)^i$ of their mother boxes $[\mathbf{x}(k)]^i \quad i = 1, \dots, N_k$.

4 Resampling

Once the updated boxes $[\mathbf{x}(k+1)|\mathbf{x}(k)]^i$ and their associated weights $w(k)^i$ have been computed, the objective is to compute a new set of disjoint boxes. This corresponds to the resampling step of the conventional particle filter.

4.1 Repartitioning

We assume that the new boxes are of the same size, that they cover the whole space defined by the union of the updated boxes $[\mathbf{x}(k+1)|\mathbf{x}(k)]^i$ $i = 1, \dots, N_k$, and that their weight is proportional to the weight of the former boxes.

For this purpose, a support box set \mathcal{Z} is computed as the minimum box such that

$$\mathcal{Z} \supseteq \bigcup_{i=1}^{N_k} [\mathbf{x}(k+1)|\mathbf{x}(k)]^i. \quad (16)$$

\mathcal{Z} is partitioned into M disjoint boxes of the same size

$$[\mathbf{z}]^i \quad i = 1, \dots, M \quad (17)$$

where $[\mathbf{z}]^i = [\underline{\mathbf{z}}^i, \overline{\mathbf{z}}^i]$, $\underline{\mathbf{z}}^i, \overline{\mathbf{z}}^i \in \mathbb{R}^{n_x}$, and

$$\overline{z}_j^i - \underline{z}_j^i = \varepsilon_j \quad i = 1, \dots, M \quad j = 1, \dots, n_x. \quad (18)$$

The box component widths are computed as

$$\varepsilon_j = \frac{\overline{Z}_j - \underline{Z}_j}{m_j} \quad j = 1, \dots, n_x \quad (19)$$

where m_j is the number of intervals along dimension j computed as

$$m_j = \lceil \frac{\overline{Z}_j - \underline{Z}_j}{\delta_j} \rceil \quad j = 1, \dots, n_x \quad (20)$$

where $\lceil \cdot \rceil$ indicates the ceiling function and δ_j the minimum accuracy for every state component j defined in Section 2.2. In this way, we guarantee that

$$\varepsilon_j \leq \delta_j \quad j = 1, \dots, n_x \quad (21)$$

Finally, the number M of boxes of the uniform grid partition is given by

$$M = \prod_{j=1}^{n_x} m_j \quad (22)$$

Once the new boxes $[\mathbf{z}]^i$ have been computed, the weight of the new boxes w_z^i can be computed as

$$w_z^i = \sum_{j=1}^{N_k} \left(\frac{\prod_{l=1}^{n_x} |[x_l(k+1)|x_l(k)]^j \cap [z_l]^i|}{\prod_{l=1}^{n_x} |[x_l(k+1)|x_l(k)]^j|} w(k)^j \right) \quad (23)$$

$$i = 1, \dots, M$$

where $[v_l]^i$ refers to the l -th component of the vector $[\mathbf{v}]^i$ and the interval width $\overline{x}_l - \underline{x}_l$ is denoted by $|[x_l]|$ for more compactness. The new weights fulfill

$$\sum_{i=1}^M w_z^i = \sum_{i=1}^{N_k} w(k)^i = 1 \quad (24)$$

The new weights w_z^i in (4.1) can be computed efficiently using Algorithm 1. This algorithm searches the number N_{inter} of boxes of \mathcal{Z} that intersect every $[\mathbf{x}(k+1)|\mathbf{x}(k)]^j$. Then, the weight $w(k)^j$ is distributed proportionally to the volume of the intersection between the updated boxes $[\mathbf{x}(k+1)|\mathbf{x}(k)]^j$ and each of the N_{inter} boxes of \mathcal{Z} that have a non-empty intersection.

Algorithm 1 Weights of the new boxes.

Algorithm Weights-new-boxes ($\mathcal{Z}, [\mathbf{x}(k+1)|\mathbf{x}(k)]^1, \dots, [\mathbf{x}(k+1)|\mathbf{x}(k)]^{N_k}, w(k)^1, \dots, w(k)^{N_k}$)
 $w_z^i \leftarrow 0 \quad i = 1, \dots, M$
for $j = 1, \dots, N_k$ **do**
 $[N_{inter}, \mathbf{V}_{inter}] = \text{intersec}([\mathbf{x}(k+1)|\mathbf{x}(k)]^j, \mathcal{Z})$
for $h = 1, \dots, N_{inter}$ **do**
 $i = \mathbf{V}_{inter}(h)$
 $w_z^i = w_z^i + \frac{\prod_{l=1}^{n_x} |[x_l(k+1)|x_l(k)]^j \cap [z_l]^i|}{\prod_{l=1}^{n_x} |[x_l(k+1)|x_l(k)]^j|} w(k)^j$
end for
end for
Return (w_z^1, \dots, w_z^M)
endAlgorithm

4.2 Controlling the number of boxes

Once the new disjoint boxes and their associated weights have been computed, the associated weights can be used to select the set of boxes that are worth pushing forward through the next iteration. This is performed by selecting the boxes with highest weights and discarding the others. In order to fulfill the confidence threshold criterion (9) proposed in Section 2.2, Algorithm 2 is proposed. The set W_z of weights w_z^i associated to the boxes $[\mathbf{z}]^i$ is defined as

$$W_z = \{w_z^1, \dots, w_z^M\}. \quad (25)$$

Given a desired confidence threshold γ , the M disjoint boxes $[\mathbf{z}]^i$ that compose the uniform grid partition of \mathcal{Z} and vector W_z with the associated weights, Algorithm 2 determines the minimum number N_{k+1} of boxes $[\mathbf{z}]^i$ with highest weights w_z^i that fulfill

$$\sum_{i=1}^{N_{k+1}} w_z^i \geq \gamma \quad (26)$$

The new state estimate $\mathcal{X}(k+1)$ is approximated by this set of N_{k+1} boxes and their prior probability by

$$P([\mathbf{x}(k+1)]^i | \mathbf{Y}(k)) \approx W_{k+1}^i \quad i = 1, \dots, N_{k+1}. \quad (27)$$

where W_{k+1}^i are the N_{k+1} highest weights of W_z associated with the disjoint boxes $[\mathbf{x}(k+1)]^i$, $i = 1, \dots, N_{k+1}$, that approximate $\mathcal{X}(k+1)$. W_{k+1}^i can be referred as the *a priori* weights.

Algorithm 2 State update at step $k+1$ with confidence threshold γ .

Algorithm State-update($[\mathbf{z}]^1, \dots, [\mathbf{z}]^M, W_z, \gamma$)
 $\gamma_c \leftarrow 0, \{\mathcal{X}(k+1)\} \leftarrow \{\emptyset\}, W_{k+1} \leftarrow \{\emptyset\}, N_{k+1} \leftarrow 0$
while $\gamma_c < \gamma$ **do**
 $[value, pos] = \max(W_z)$
 $\text{addbox}(\mathcal{X}(k+1), [\mathbf{z}]^{pos})$
 $\text{addelement}(W_{k+1}, value)$
 $\gamma_c = \gamma_c + value$
 $W_z(pos) \leftarrow 0$
 $N_{k+1} \leftarrow N_{k+1} + 1$
endwhile
Return ($\mathcal{X}(k+1), W_{k+1}, N_{k+1}$)
endAlgorithm

This algorithm generates a set of state boxes $\{\mathcal{X}(k+1)\}$ a list of weights W_{k+1}^i , a cumulative weight variable γ_c , and a cardinality variable N_{k+1} . At the beginning of the algorithm, the state boxes and weight list are initialized as empty sets and cumulative weight and cardinality variable are initialized to zero. The loop "while" operates as a sorting, eliminating the boxes with smallest weights so that the cumulative sum of the boxes with largest weights is greater or equal to the threshold γ . If the state space is not bounded, the threshold $0 < \gamma < 1$ does not guarantee a bounded number of boxes in a worst-case scenario in which the measurements do not emphasize some particles against others. In this case, a maximum number of particles N_{max} should be imposed.

5 State estimation and fault detection

5.1 State estimation

Once the set of N_{k+1} disjoint boxes $[\mathbf{x}(k+1)]^i$, $i = 1, \dots, N_{k+1}$, that approximate $\mathcal{X}(k+1)$ and their associated *a priori* weights W_{k+1}^i have been computed, their measurement updated weights $w(k+1)^i$ are obtained using (11). Then, according to [2], the state at instant $k+1$ is approximated by

$$\hat{\mathbf{x}}(k+1) = \sum_{i=1}^{N_{k+1}} w(k+1)^i \mathbf{x}_0^i(k+1) \quad (28)$$

where $\mathbf{x}_0^i(k+1)$ is the center of the particle box $[\mathbf{x}(k+1)]^i$.

Algorithm 3 summarizes the whole state estimation procedure.

Algorithm 3 State estimation

Algorithm State estimation

Initialize $\mathcal{X}(0)$, N_0 and $P([\mathbf{x}(k)]^i | \mathbf{Y}(k-1))_{k=0, i=1 \dots N_0}$

for $k = 1, \dots, \text{end do}$

Obtain Input/Output data $\{\mathbf{u}(k), \mathbf{y}(k)\}$

Measurement update

compute $\Lambda(k)$ using Eq. (12)

compute $w(k)^i$ using Eq.(11) $i = 1 \dots N_0$

State estimation

compute $\hat{\mathbf{x}}(k)$ using (28)

State update

compute $[\mathbf{x}(k+1)]^i | \mathbf{x}(k)^i$ $i = 1 \dots N_0$ using (15)

compute \mathcal{Z} that fulfils (16)

compute disjoint boxes $[\mathbf{z}]^i$ $i = 1, \dots, M$ of (17)

compute weights w_z^i using Algorithm 1

compute new state estimation using Algorithm 2

N_{k+1} disjoint boxes that approximate $\mathcal{X}(k+1)$

Prior probabilities given by weights W_{k+1}

end for

endAlgorithm

5.2 Fault detection

In our framework, fault detection can be formulated as detecting inconsistencies based on the state estimation. To do so, we propose the two following indicators:

- Abrupt changes in the state estimation provided by (28) from instant $k-1$ to instant k , i.e. abnormal high values of $\sqrt{(\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1))(\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1))^T}$

- Abnormal low sum of the unnormalized posterior probability of all the particles at instant k , which means that all the particles have been penalized by the current measurements. This abnormality can be checked by thresholding $\Lambda(k)$ defined in (12).

If enough representative fault free data are available, the indicators defined above can be determined by means of thresholds computed with these data. For example, the threshold that defines the abnormal abrupt change in state estimation can be computed as

$$\Delta \hat{\mathbf{x}}^{max} = \beta_1 \max_{i=2, \dots, L} \sqrt{(\hat{\mathbf{x}}(i) - \hat{\mathbf{x}}(i-1))(\hat{\mathbf{x}}(i) - \hat{\mathbf{x}}(i-1))^T} \quad (29)$$

where L is the length of the fault free scenario and $\beta_1 > 1$ a tuning parameter. Then the fault detection test consists in checking at each instant k if

$$\sqrt{(\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1))(\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1))^T} > \Delta \hat{\mathbf{x}}^{max} \quad (30)$$

In a similar way, threshold Λ^{min} that defines the minimum expected unnormalized posterior probability can be computed as

$$\Lambda^{min} = \beta_2 \min_{i=2, \dots, L} (\Lambda(i)) \quad (31)$$

where $\Lambda(i)$ is determined using (12) and $0 < \beta_2 < 1$ is a tuning parameter. Then the fault detection test consists in checking at each instant k if

$$\Lambda(k) < \Lambda^{min} \quad (32)$$

6 Application example

In this section a target tracking in a sensor network example presented in [8] is used to illustrate the state estimation method presented above. The problem consists of three sensors and one target moving in the horizontal plane. Each sensor can measure distance to the target, and by combining these a position fix can be computed. Fig. 1 depicts a scenario with a trajectory and a certain combination of sensor locations (S_1 , S_2 and S_3).

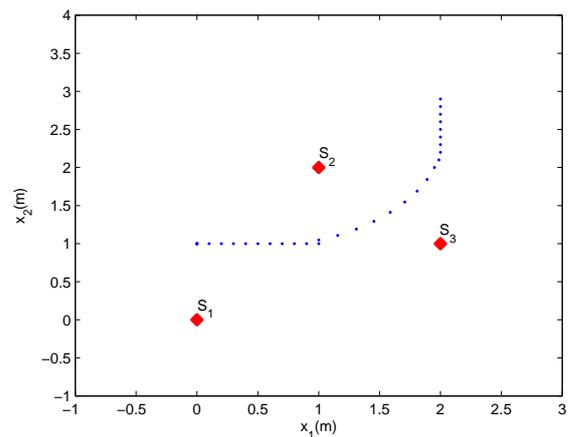


Figure 1: Target true trajectory and sensor positions in the bounded horizontal plane

The behaviour of the system can be described by the following discrete time state-space model:

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} + T_s \begin{pmatrix} v_1(k) \\ v_2(k) \end{pmatrix} \quad (33)$$

$$\begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \end{pmatrix} = \begin{pmatrix} \sqrt{(x_1(k) - S_{1,1})^2 + (x_2(k) - S_{1,2})^2} \\ \sqrt{(x_1(k) - S_{2,1})^2 + (x_2(k) - S_{2,2})^2} \\ \sqrt{(x_1(k) - S_{3,1})^2 + (x_2(k) - S_{3,2})^2} \end{pmatrix} + \begin{pmatrix} e_1(k) \\ e_2(k) \\ e_3(k) \end{pmatrix}$$

where $x_1(k)$ and $x_2(k)$ are the object coordinates bounded by $-1 \leq x_1(k) \leq 3$ and $-1 \leq x_2(k) \leq 4 \forall k \geq 0$. $T_s = 0.5s$ is the sampling time, $v_1(k)$ and $v_2(k)$ are the speed components of the target that are unknown but considered bounded by the maximum speed $\sigma_v = 0.4m/s$ ($|v_1(k)| \leq \sigma_v$ and $|v_2(k)| \leq \sigma_v$). $y_1(k)$, $y_2(k)$ and $y_3(k)$ are the distances measured by the sensors. $S_{i,j}$ denotes the component j of the location of sensor i . $e_1(k)$, $e_2(k)$ and $e_3(k)$ are the stochastic measurement additive errors $p_{e_i} \sim N(0, \sigma_i)$ with $\sigma_1 = \sigma_2 = \sigma_3 = \sqrt{0.05}m$.

Fig. 2 shows the evolution of the real sensor distances and measurements in the target trajectory scenario depicted in Fig. 1.

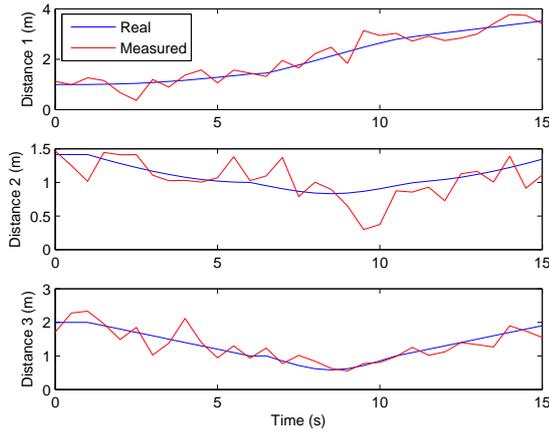


Figure 2: Real and measured distances from the target to the sensors

In order to apply the state estimation methodology presented above, a minimum accuracy $\delta_1 = \delta_2 = \delta = 0.2m$ has been selected for both components. No a priori information has been used in the initial state. Then, a uniform grid of disjoint boxes with the same weights and component widths $\varepsilon_1 = \varepsilon_2 = \delta$ that covers all the bounded coordinates $-1 \leq x_1 \leq 3$ and $-1 \leq x_2 \leq 4$ has been chosen as initial state $\mathcal{X}(0)$. Posterior probabilities of the boxes have been approximated by weights $w(k)^i$ computed using the new sensor distances measurements in (4.1). State update has been computed considering speed bounds in (33). The new boxes have been rearranged considering the minimum accuracy δ and their associated weights have been computed using (4.1). Finally, Algorithm 2 with threshold $\gamma = 1$ has been applied to reduce the number of boxes.

Figs. 3 and 4 depict the box weights and their contours using measurement $y_1(1)$ (up) and all the measurements at

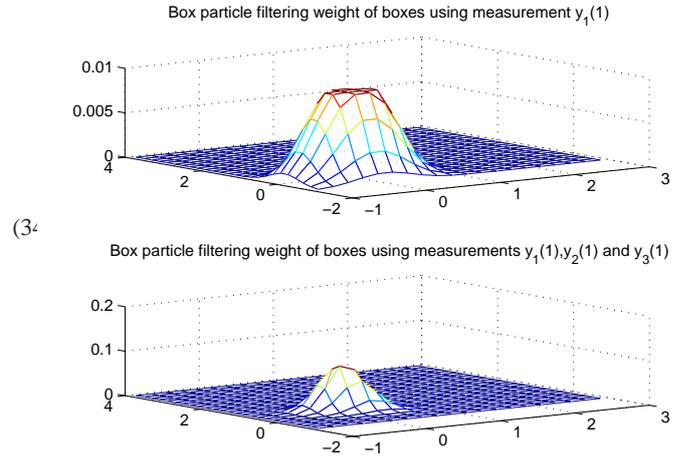


Figure 3: Box weights using measurement $y_1(k)$ (up) and measurements $(y_1(k), y_2(k), y_3(k))^T$ (down)

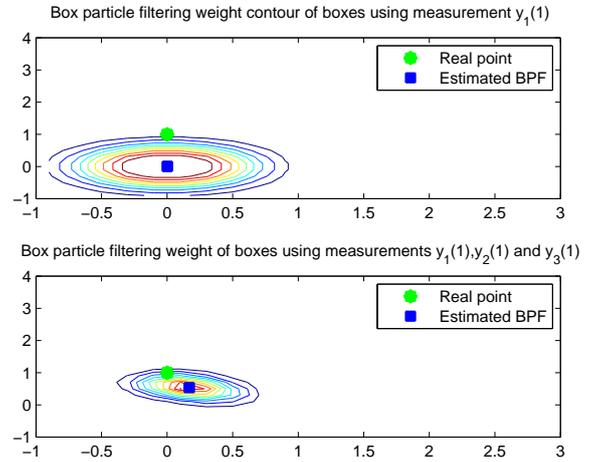


Figure 4: Box weight contours using measurement $y_1(k)$ (up) and measurements $(y_1(k), y_2(k), y_3(k))^T$ (down)

instant $k = 1$ ($y_1(1), y_2(1)$ and $y_3(1)$) (down). Fig. 5 depicts the box weights and their contours using the measurements at hand at instant $k = 2$.

The real trajectory and the one estimated using (28) are shown in Fig. 6.

Finally, different additive sensor faults have been simulated and satisfactory results of the fault detection tests (30) and (32) have been obtained for faults bigger than $0.5m$ using thresholds $\Delta \hat{x}^{max}$ and Λ^{min} computed with (29) and (31) with $L = 3200$, $\beta_1 = 1.1$ and $\beta_2 = 0.9$.

Fig. 7 shows the real trajectory and the one estimated using (28) when an additive fault of $+0.5m$ affects sensor S_1 at time $k = 22$. The behaviour of fault detection tests (30) and (32) is depicted in Fig. 8. As seen in this figure, both thresholds are violated at time instant $k = 22$ and therefore the fault is detected at this time instant.

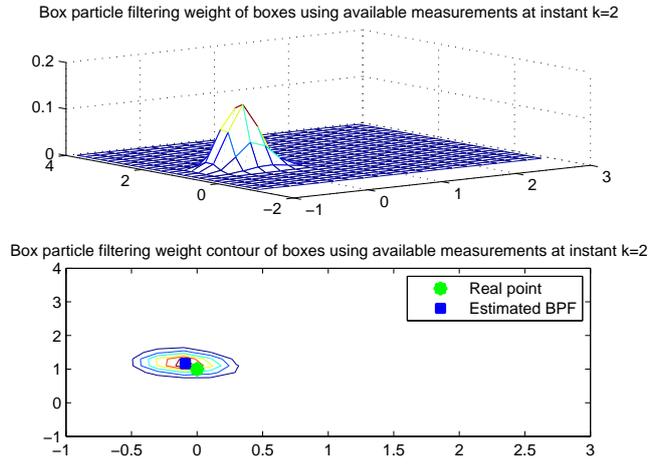


Figure 5: Box weights (up) and Box weights contours (down) at instant $k = 2$

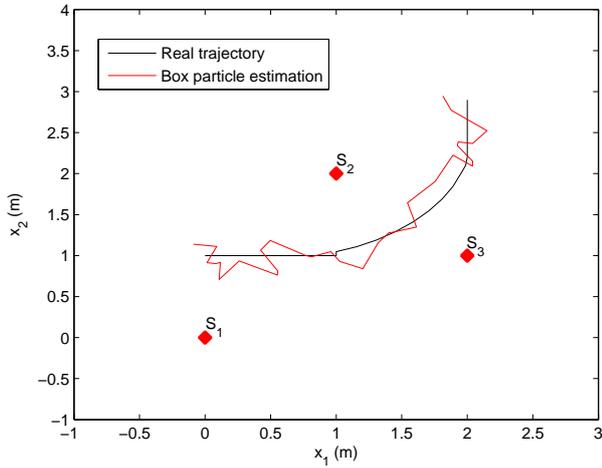


Figure 6: Trajectories

7 Conclusion and perspectives

A Box particle algorithm has been proposed for estimation and fault detection in the case of nonlinear systems with stochastic and bounded uncertainties. Using this method in the case of a target tracking sensor networks illustrates its feasibility. It has been shown how the measurement update state for the box particle is derived from the particle case. However convergence and stability of this filter have to be proved. Resampling unfortunately drops information and waives guaranteed results that characterize interval analysis based solutions. However without resampling the particle filter suffers from sample depletion. This is the reason why resampling is a critical issue in particle filtering (Gustafsson 2002). This approach has to be compared to other PF variants which reduce the number of particles [2] and further investigations concerning resampling are required, in particular if we want to take better benefit of the interval based approach.

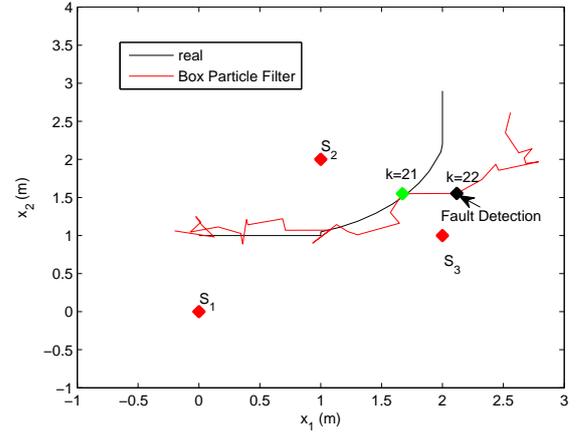


Figure 7: Trajectories in fault scenario

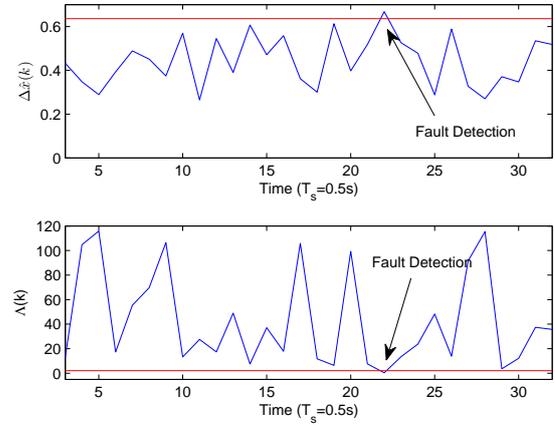


Figure 8: Fault indicators and thresholds in the fault scenario

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Technology through the Project ECOCIS (Ref. DPI2013-48243-C2-1-R) and Project HARRICIS (Ref. DPI2014-58104-R).

A Demonstration of Measurement update: "From particles to boxes"

A.1 Particle filtering

Consider the particles $\{x(k)^j\}_{j=1}^N$ uniformly distributed in $x(k)^j \in [\underline{x}(k), \overline{x}(k)] \forall j = 1, \dots, N$ where $\underline{x}(k), \overline{x}(k) \in \mathbb{R}$. Then according to [1] the relative posterior probability for each particle is approximated by

$$P(\mathbf{x}(k)^j | \mathbf{Y}(k)) \approx \frac{1}{c(k)} P(\mathbf{x}(k)^j | \mathbf{Y}(k-1)) p_e(y(k) - h(x(k)^j)) \quad (35)$$

with

$$c(k) = \sum_{j=1}^N P(\mathbf{x}(k)^j | \mathbf{Y}(k)) \quad (36)$$

A.2 Grouping particles

If we group the N particles in N_g groups of ΔN elements

$$\{x(k)^j\}_{j=1}^N = \bigcup_{i=1}^{N_g} \{x(k)^l\}_{l=1+(i-1)\Delta N}^{i\Delta N} \quad (37)$$

with $N_g = \frac{N}{\Delta N}$

If we select the groups of points in such a way that

$$\{x(k)^l\}_{l=1+(i-1)\Delta N}^{i\Delta N} \in [x(k)]^i \quad \forall i = 1, \dots, N_g \quad (38)$$

where

$$[x(k)]^i = [\underline{x(k)} + (i-1)\Delta L, \overline{x(k)} + i\Delta L] \quad (39)$$

with

$$\Delta L = \frac{\overline{x(k)} - \underline{x(k)}}{N_g} \quad (40)$$

If the number of particles $N \rightarrow \infty$ and therefore $\Delta N \rightarrow \infty$

$$P([\mathbf{x}(k)]^i | \mathbf{Y}(k)) \approx \sum_{j=1+(i-1)\Delta N}^{i\Delta N} P(\mathbf{x}(k)^j | \mathbf{Y}(k)) \quad (41)$$

according to (35)

$$P([\mathbf{x}(k)]^i | \mathbf{Y}(k)) \approx \frac{\sum_{j=1+(i-1)\Delta N}^{i\Delta N} P(\mathbf{x}(k)^j | \mathbf{Y}(k-1)) p_e(y(k) - h(x(k)^j))}{\sum_{l=1}^{N_g} \sum_{j=1+(l-1)\Delta N}^{l\Delta N} P(\mathbf{x}(k)^j | \mathbf{Y}(k-1)) p_e(y(k) - h(x(k)^j))} \quad (42)$$

If we consider the particles in the same group i have the same prior probabilities, then:

$$\frac{P([\mathbf{x}(k)]^i | \mathbf{Y}(k-1))}{\Delta N} \quad \forall j = 1 + (i-1)\Delta N, \dots, i\Delta N \quad (43)$$

and (42) leads to

$$P([\mathbf{x}(k)]^i | \mathbf{Y}(k)) \approx \frac{P([\mathbf{x}(k)]^i | \mathbf{Y}(k-1)) \sum_{j=1+(i-1)\Delta N}^{i\Delta N} p_e(y(k) - h(x(k)^j))}{\sum_{l=1}^{N_g} (P([\mathbf{x}(k)]^l | \mathbf{Y}(k-1)) \sum_{j=1+(l-1)\Delta N}^{l\Delta N} p_e(y(k) - h(x(k)^j)))} \quad (44)$$

If the N particles are uniformly distributed in the interval $[\underline{x(k)}, \overline{x(k)}]$, i.e

$$x(k)^j - x(k)^{j-1} = \Delta x(k) \quad \forall j = 2, \dots, N \quad (45)$$

where

$$\Delta x(k) = \frac{\overline{x(k)} - \underline{x(k)}}{N} = \frac{\Delta L}{\Delta N} \quad (46)$$

Then

$$\sum_{j=1+(i-1)\Delta N}^{i\Delta N} p_e(y(k) - h(x(k)^j)) \Delta x(k) \approx \int_{(1+(i-1)\Delta N)\Delta x(k)}^{(i\Delta N)\Delta x(k)} p_e(y(k) - h(x(k))) dx(k) \approx \int_{x(k) \in [x(k)]^i} p_e(y(k) - h(x(k))) dx(k) \quad (47)$$

Finally, multiplying the numerator and denominator of equation (44) by Δx , we obtain the particle box measurement update equation

$$P([\mathbf{x}(k)]^i | \mathbf{Y}(k)) \approx \frac{P([\mathbf{x}(k)]^i | \mathbf{Y}(k-1)) \int_{x(k) \in [x(k)]^i} p_e(y(k) - h(x(k))) dx(k)}{\sum_{l=1}^{N_g} (P([\mathbf{x}(k)]^l | \mathbf{Y}(k-1)) \int_{x(k) \in [x(k)]^l} p_e(y(k) - h(x(k))) dx(k))} \quad (48)$$

that corresponds to the equation (11) with

$$\Lambda(k) = \sum_{l=1}^{N_g} (P([\mathbf{x}(k)]^l | \mathbf{Y}(k-1)) \int_{x(k) \in [x(k)]^l} p_e(y(k) - h(x(k))) dx(k)) \quad (49)$$

References

- [1] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, 2002.
- [2] F. Abdallah, A. Gning, and P. Bonnifait. Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44(3):807–815, 2008.
- [3] A. Doucet, N. De Freitas, and N. Gordon. *An introduction to sequential Monte Carlo methods*. Springer, 2001.
- [4] A. Gning, L. Mihaylova, and F. Abdallah. Mixture of uniform probability density functions for non linear state estimation using interval analysis. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8. IEEE, 2010.
- [5] R.M. Fernández-Cantí, S. Tornil-Sin, J. Blesa, and V. Puig. Nonlinear set-membership identification and fault detection using a bayesian framework: Application to the wind turbine benchmark. In *Proceedings of the IEEE Conference on Decision and Control*, pages 496–501, 2013.
- [6] J. Xiong, C. Jauberthie, L. Travé-Massuyès, and F. Le Gall. Fault detection using interval kalman filtering enhanced by constraint propagation. In *Proceedings of the IEEE Conference on Decision and Control*, pages 490–495, 2013.
- [7] A. Gning, B. Ristic, and L. Mihaylova. Bernoulli particle/box-particle filters for detection and tracking in the presence of triple measurement uncertainty. *IEEE Transactions on Signal Processing*, 60(5):2138–2151, 2012.
- [8] F. Gustafsson. *Statistical sensor fusion*. Studentlitteratur, Lund, 2010.

Minimal Structurally Overdetermined Sets Selection for Distributed Fault Detection

Hamed Khorasgani¹ Gautam Biswas¹ and Daniel Jung²

¹Institute of Software Integrated Systems, Vanderbilt University, USA

e-mail: {hamed.g.khorasgani,gautam.biswas}@vanderbilt.edu

²Dept. of Electrical Engineering, Linkoping University, Sweden

e-mail: daner@isy.liu.se

Abstract

This paper discusses a distributed diagnosis approach, where each subsystem diagnoser operates independently without a coordinator that combines local results and generates the correct global diagnosis. In addition, the distributed diagnosis algorithm is designed to minimize communication between the subsystems. A Minimal Structurally Overdetermined (MSO) set selection approach is developed as a Binary Integer Linear Programming (BILP) optimization problem for subsystem diagnoser design. For cases, where a complete global model of the system may not be available, we develop a heuristic approach, where individual subsystem diagnosers are designed incrementally, starting with the local system MSOs and progressively extending the local set to include MSOs from the immediate neighbors of the subsystem. The inclusion of additional neighbors continues till the MSO set ensures correct global diagnosis results. A multi-tank system is used to demonstrate and validate the proposed methods.

1 Introduction

The Minimal Structurally Overdetermined (MSO) sets approach has been used extensively for designing model based fault detection and isolation (FDI) schemes for complex systems [Krysander *et al.*, 2008a; Krysander *et al.*, 2008b; Svard *et al.*, 2012]. However, for large complex systems such as aircraft and other transportation systems, manufacturing processes, supply chain and distribution networks, and power generation and the power grid it is becoming imperative to develop distributed approaches to monitoring and diagnosis to overcome the need for complete global models, while also addressing computational complexity and reliability problems for the diagnosers [Leger *et al.*, 1999; Shum *et al.*, 1988; Deb *et al.*, 1998; Lanigan *et al.*, 2011].

Unlike centralized approaches, distributed approaches are more reliable because they avoid single points of failure. In addition, they can reduce the problems of noise, corruption, and losses that can occur when transmitting signals from individual subsystems to a centralized fault diagnosis unit. Measurement noise and signal corruption can significantly affect diagnoser robustness and accuracy [Ferrari *et al.*, 2012]. Transmission delays not only increase detection time, but can also affect the order of detection,

which can further affect diagnostic accuracy. Detection time is important for the safe and reliable operation of safety-critical systems. Faster fault detection and isolation enables accompanying fault tolerant control units to react in a timely manner, thus reducing damage and down time of systems [Roychoudhury *et al.*, 2009; Daigle *et al.*, 2007; Duarte Jr and Nanya, 1998; Rish *et al.*, 2005; Bregon *et al.*, 2014]. The computational intractability of building centralized diagnosers for the large systems is another important reason to develop distributed solutions for FDI problems.

In this paper, we formulate the distributed minimal structurally overdetermined set selection as a binary integer linear programming (BILP) problem [Wolsey, 1998]. The approach efficiently picks a minimal number of measurements from a subsystem and its neighboring subsystems to develop a local diagnoser for each subsystem of the larger, complex dynamic system. We start with an efficient algorithm designed by [Krysander *et al.*, 2008a] for finding minimally overdetermined sets of constraints to generate the minimal structurally overdetermined (MSO) sets for designing the diagnoser. Other researchers have employed binary integer programming and binary linear integer programming for optimal sensor placement for fault detection and isolation [Sarrate *et al.*, 2007; Rosich *et al.*, 2009]. In this paper, we utilize BILP for distributed MSO selection to facilitate an efficient distributed diagnosis approach.

Our method is designed in a way that the subsystem diagnosers, once designed can operate independently with no communication with the other subsystem diagnosers (other than a minimal number of shared measurements), but still provide globally correct diagnosis results. Unlike [Lafortune, 2007; Debouk *et al.*, 2000; Indra *et al.*, 2012] this method does not require the use of a centralized coordinator during on-line operations. Therefore, we avoid the single point-of-failure problem of centralized diagnosers. Our method assumes the availability of a global system model from which the set of MSOs for the system can be derived. The independent subsystem diagnosers are designed to minimize the sharing of measurements across subsystems, thus decreasing the cost, and increasing the reliability of the overall system diagnosis.

However, global models of a complex system are hard to construct and may not be readily available. Subsystems are often provided by different manufacturers, who are not willing to pass along all of the intellectual property associated with the subsystem to the system integrator. Therefore, to avoid the unrealistic assumption that the complete model of the complex system is available for subsystem diagnoser de-

sign, we propose a second algorithm that constructs the individual subsystem diagnosers without assuming the availability of a global model. The modified algorithm is computationally more efficient, but we cannot guarantee that the shared measurements between the subsystems is minimal globally (i.e., across the entire system).

The rest of this paper is organized as follows. The background material, definitions and the running example, a four-tank system, are presented in Section 2. The distributed diagnosis problem formulation is presented in Section 3. Algorithm 1 for distributed MSO set selection is described in Section 4. The heuristic modifications to Algorithm 1 given the global model is not available is presented in Section 5 as the incremental algorithm. Section 6 discusses the contributions of the paper in relation to previous work, and presents the conclusion of the paper.

2 Background

This section introduces the basic concepts associated with MSO set selection for structural diagnosis of dynamic systems. The system model S is defined as follows.

Definition 1 (System model). *A system model S is a four-tuple: (V, M, E, F) , where V is the set of variables, M is the set of measurements, E is the set of equations and F is the set of system faults.*

We use a configured four tank system, shown in Figure 1, as a running example throughout this paper to describe the problem, and to illustrate the algorithms for distributed MSO set selection. We assume each tank, and the outlet pipe to its right, constitute a subsystem. Therefore, this system has four subsystems. Two of the subsystems, 1 and 3, also have inflows into their tanks. We assume the subsystems are disjoint, i.e., they have no overlapping components. Associated with each subsystem are a set of measurements that are shown as encircled variables in the figure.

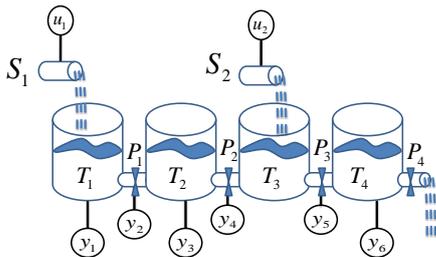


Figure 1: Running example: Four Tank System.

More generally, we assume the system, S has n pre-defined subsystems, S_1, S_2, \dots, S_n . Each subsystem model is defined as:

Definition 2 (Subsystem model). *A subsystem model of system model S , S_i ($1 \leq i \leq k$) is also a four-tuple: (V_i, M_i, E_i, F_i) , where $V_i \subseteq V$, $M_i \subseteq M$, $E_i \subseteq E$ and $F_i \subseteq F$. Also, $S_1 \cup S_2 \cup \dots \cup S_k = S$.*

For illustration, the first subsystem in our running example is described by the following set of equations:

$$\begin{aligned} e_1 : \dot{p}_1 &= \frac{1}{C_{T1} + f_1} (q_{in1} - q_1) & e_4 : q_{in1} &= u_1 \\ e_2 : q_1 &= \frac{p_1 - p_2}{R_{P1} + f_2} & e_5 : p_1 &= y_1 \\ e_3 : p_1 &= \int \dot{p}_1 dt & e_6 : q_1 &= y_2. \end{aligned} \quad (1)$$

Therefore, $E_1 = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ defines the set of equations, $V_1 = \{\dot{p}_1, p_1, p_2, q_{in1}, q_1\}$ defines the set of variables, $M_1 = \{u_1, y_1, y_2\}$ defines the set of subsystem measurements, and $F_1 = \{f_1, f_2\}$ defines the set of faults associated with this subsystem model.

Similarly, the second subsystem model is defined by the following equations:

$$\begin{aligned} e_7 : \dot{p}_2 &= \frac{1}{C_{T2} + f_3} (q_1 - q_2) & e_{10} : p_2 &= y_3 \\ e_8 : q_2 &= \frac{p_2 - p_3}{R_{P2} + f_4} & e_{11} : q_2 &= y_4. \\ e_9 : p_2 &= \int \dot{p}_2 dt \end{aligned} \quad (2)$$

For this subsystem the set of equations is $E_2 = \{e_7, e_8, e_9, e_{10}, e_{11}\}$, the set of variable is $V_2 = \{\dot{p}_2, p_2, p_3, q_1, q_2\}$, the set of measurements is $M_2 = \{y_2, y_4\}$, and $F_2 = \{f_3, f_4\}$ is the set of faults.

In this paper, we assume there are no overlapping components among the subsystems. However, the subsystems may share variables at their interface. For example, the liquid flowrate at outlet pipe of subsystem $q_i = q_{i'}$, the liquid flowrate at input to connected tank $i + 1$.

Definition 3 (First Order Connected Subsystems). *Two subsystems, S_i and S_j are defined to be first order connected if and only if they have at least one shared variable.*

In the running example, subsystems S_1 and S_2 are first order connected and their shared variables are $V_1 \cap V_2 = \{p_2, q_1\}$. The two other subsystems in the running example are:

$$\begin{aligned} e_{12} : \dot{p}_3 &= \frac{1}{C_{T3}} (q_{in2} + q_2 - q_3) & e_{15} : q_{in2} &= u_2 \\ e_{13} : q_3 &= \frac{p_3 - p_4}{R_{P3} + f_5} & e_{16} : q_3 &= y_5. \\ e_{14} : p_3 &= \int \dot{p}_3 dt \end{aligned} \quad (3)$$

$$\begin{aligned} e_{17} : \dot{p}_4 &= \frac{1}{C_{T4} + f_6} (q_3 - q_4) & e_{19} : p_4 &= \int \dot{p}_4 dt \\ e_{18} : q_4 &= \frac{p_4}{R_{P4}} & e_{20} : p_4 &= y_6. \end{aligned} \quad (4)$$

In more general terms, i th order connected subsystem models are defined as follows.

Definition 4 (i th Order Connected Subsystems). *Two subsystems, S_k and S_j are defined to be i th order connected if and only if there exists a subsystem model S_m that is $(i-1)$ th order connected to S_k , and is first-order connected to S_j , or S_m is $(i-1)$ th order connected to S_j , and is first-order connected to S_k .*

For example in the four tank system, S_1 and S_3 are second order connected because both of them are first order connected to S_2 .

In this paper, we use MSO sets [Krysander *et al.*, 2008b] as the primary conceptual approach for fault detection and isolation. The formal definitions of Structurally Overdetermined (SO) and MSO sets are:

Definition 5. (*Structural Overdetermined Set*) Consider a set of equations and its associated variables, measurements, and faults: (E, V, M, F) . This set of equations is structurally overdetermined (SO) if the cardinality of the set $\{E\}$ is greater than the cardinality of set $\{V\}$, i.e. $|E| > |V|$.

Definition 6. (*Minimal Structurally Overdetermined Set*) A set of over determined equations is minimal structurally overdetermined (MSO) if it has no subset of structurally overdetermined equations.

Consider subsystem S_1 of the four tank system in equation (1). Using the software developed by [Krysander *et al.*, 2008a], we can compute the only minimal structurally overdetermined set in this subsystem as $MSO_{11} = (E_{11}, V_{11}, M_{11}, F_{11})$, where $E_{11} = \{e_1, e_3, e_4, e_5, e_6\}$, $V_{11} = \{\dot{p}_1, p_1, q_{in1}, q_1\}$, $M_{11} = \{u_1, y_1, y_2\}$ and $F_{11} = \{f_1\}$. For the sake of brevity and simplification we simply say a specific equation, variable, measurement, or fault is a member of a MSO in the rest of the paper. For example, we say $f_1 \in MSO_{11}$.

MSOs represent the redundancies in the system and can form the basis for fault detection and isolation. Global and Local fault detectability are defined as:

Definition 7. (*Globally detectable fault*) A fault $f \in F$ is globally detectable in system S if there is a minimal structurally overdetermined set MSO_i in the system, such that $f \in MSO_i$.

Definition 8. (*Locally detectable fault*) A fault $f \in F_i$ is locally detectable in subsystem S_i if there is a minimal structurally overdetermined set MSO_i in the subsystem that $f \in MSO_i$.

Consider Definition 8 and equation (1). Fault f_1 is locally detectable because $f_1 \in MSO_{11}$ but f_2 is not locally detectable since there is no MSO in this subsystem that includes f_2 . To detect f_2 locally, the diagnosis subsystem needs to include additional measurements. Global and Local fault isolability are defined as:

Definition 9. (*Globally isolable fault*) A fault $f_i \in F$ is globally isolable from fault $f_j \in F$ if there exists a minimal structurally overdetermined set MSO_i in the system S , such that $f_i \in MSO_i$ and $f_j \notin MSO_i$.

Definition 10. (*Locally isolable fault*) A fault $f_i \in F_i$ is locally isolable from fault $f_j \in F$ if there exists a minimal structurally overdetermined set MSO_i in subsystem S_i , such that $f_i \in MSO_i$ and $f_j \notin MSO_i$.

Note that if a fault f_i is locally detectable in a subsystem S_i , it is globally detectable too, and if a fault f_i is locally isolable from a fault f_j , it is globally isolable from f_j as well. The problem of MSO selection is presented as a binary integer linear programming (BILP) problem in this paper. BILP is a special case of the integer linear programming problem (ILP), where the unknowns to be solved for are binary variables.¹

¹See definition in Wikipedia: https://en.wikipedia.org/wiki/Integer_programming.

Definition 11. (*Binary integer linear programming problem (BILP)*) A Binary integer linear programming problem is a special case of an integer linear programming (ILP) optimization problem in which some or all the unknown variables to be solved for are required to be binary, and the constraints in the problem and the objective function, like ILP, are linear.

The mathematical formulation of BILP is as follows.

$$\begin{aligned} \min c^T x \\ Ax \leq b \\ \exists x_b \subset x \\ \forall x_k \in x_b \Rightarrow x_k \in \{0, 1\}, \end{aligned} \quad (5)$$

where vector c is the cost weights and matrix A and vector b define linear constraints, x represents the variables, and x_b represents the binary variables [Wolsey and Nemhauser, 2014].

3 Problem Formulation

Designing a set of distributed diagnosers that together have the same diagnosability as a centralized diagnoser is the focus of our work in this paper. In the ideal case, each subsystem includes sufficient redundancies, such that its set of MSOs is sufficient to detect and isolate all of its faults, F_i uniquely and unambiguously. In that case, we can associate an independent diagnoser D_i with each subsystem S_i ; $1 \leq i \leq k$, and each diagnoser operates with no centralized control, and no exchange of information with other diagnosers. If the independence among diagnosers does not hold, then the subsystems need to communicate some of their measurements to other subsystems to detect and isolate the faults. To address this problem in an efficient way, we derive an integrated approach to select a set of MSOs for each subsystem that guarantee full diagnosability and minimum exchange of measurements among subsystems.

Given subsystems, S_i ; $1 \leq i \leq k$, with a set of local fault candidates, F_i , such that $\bigcup_{i=1}^k F_i = F$. We may need to augment each subsystem with additional measurements that are typically acquired from the (nearest) neighbors of the subsystem, such that all of the faults associated with the extended model of this subsystem are detectable and isolable. In the worst case, all of the measurements from another subsystem may have to be included to make the current subsystem diagnosable. When such a situation occurs, we say the two subsystems are merged and represented by a common diagnoser, therefore, the total number of independent distributed diagnosers may be less than k .

Each MSO is sensitive to a set of faults and, therefore can be used to detect them and isolate them from the other faults in the system. For each subsystem S_i , our goal is to find a minimal set of MSOs that provide maximum detectability and isolability to that subsystem. A set of MSOs is minimal if there is no subset of MSOs that provides the same detectability and isolability. To achieve distributed fault diagnosis, we also want each subsystem to use the minimum number of measurements from the other subsystems. In other words, we want to minimize communication or the amount of data (measurements) to be transmitted between the subsystems. More formally, the problem for designing a diagnoser for a particular subsystem S_i can be described as follows:

Consider $M\mathcal{S}\mathcal{O} = \{M\mathcal{S}\mathcal{O}_1, M\mathcal{S}\mathcal{O}_2, \dots, M\mathcal{S}\mathcal{O}_r\}$ as the set of possible MSOs for the subsystem S_i . We need to develop an algorithm to select a minimal subset of $M\mathcal{S}\mathcal{O}$ that guarantees maximal structural detectability and isolability for faults F_i associated with the subsystem, and include a minimum number of measurements from the other subsystems in the system to assure the equivalence of local and global diagnosability, i.e.,

$$\begin{aligned} & \forall S_i; \quad 1 \leq i \leq k \\ & \text{Select } M\mathcal{S}\mathcal{O}_{S_i} \subset M\mathcal{S}\mathcal{O} \\ & \text{s.t. } \quad \min_{M_o \subseteq M} |M_o| \\ & \quad D_i(M_i \cup M_o) = D_i(M), \\ & \quad I_i(M_i \cup M_o) = I_i(M), \end{aligned} \quad (6)$$

where M_o represents the set of measurement we need to communicate to the subsystem S_i along with the set of measurements, M_i associated with the subsystem S_i . M represents the set of all measurements in the system. For a given set of measurements, X , $D_i(X)$ represents the set of detectable faults in F_i , and $I_i(X)$ represents the set of isolable faults in F_i from the system faults, F .

In the next section we formulate the problem as a BILP problem. Formulating the problem as a BILP, enables us to use a number of well-developed tools like branch and bound algorithms [Land and Doig, 1960] and branch and cut algorithms [Mitchell, 2002] to solve the problem. However, much like integer linear programming, the general BILP solution is exponential.

4 MSOs Selection for Distributed Fault Detection Using Global Model

In this section, we present our algorithm to select a minimal set of residuals for each subsystem of a system whose global model is available as a set of equations. In the next section, we modify this algorithm to make it applicable to much larger systems, where a compiled global model is not available.

For the situation in which the global model is known, M in equation (6) is the set of all system measurements. Assume we have l measurements in the system: $M = \{m_1, m_2, \dots, m_l\}$. The measurements imply redundancies in the system model that form the basis for generating MSOs. Let us assume we can generate r MSOs given M : $M\mathcal{S}\mathcal{O} = \{M\mathcal{S}\mathcal{O}_1, M\mathcal{S}\mathcal{O}_2, \dots, M\mathcal{S}\mathcal{O}_r\}$. Our goal is to design an algorithm that selects $M\mathcal{S}\mathcal{O}_i \subseteq M\mathcal{S}\mathcal{O}$ in a way that we add a minimum number of measurements $M_o \subseteq M$, $M_i \cap M_o = \emptyset$, i.e., measurements from the system not belonging to subsystem i , to a subsystem to make all its faults globally diagnosable. Note that this is equivalent to the set covering problem and, therefore, any algorithm for finding the minimal measurements is exponential, in general. In the past, we have adopted heuristic search methods for solving this problem. Our approach for designing subsystem diagnosers used the Temporal Causal Graph (TCG) approach [Roychoudhury *et al.*, 2009]. In this paper, we formulate the search for minimal sensors as a BILP problem. The general formulation of BILP is presented in (5), and there are several tools available for solving this problem.²

²For example, see <http://www.mathworks.com/help/optim/ug/>

To formulate the problem (6) as a BILP problem we define a binary variable $x(k)$: $1 \leq k \leq l$, for measurement m_k in the system as follows:

$$x(k) = \begin{cases} 1 & \text{if } m_k \in M_i \cup M_o \\ 0 & \text{if } m_k \notin M_i \cup M_o, \end{cases} \quad (7)$$

where M_o is the answer to problem (6). We also define $x(k+l)$: $1 \leq k \leq r$, for MSO $M\mathcal{S}\mathcal{O}_k$ in the system as follows.

$$x(k+l) = \begin{cases} 1 & \text{if } M\mathcal{S}\mathcal{O}_k \in M\mathcal{S}\mathcal{O}_i \\ 0 & \text{if } M\mathcal{S}\mathcal{O}_k \notin M\mathcal{S}\mathcal{O}_i. \end{cases} \quad (8)$$

To minimize the number of measurements from the other subsystems, we develop the following cost function c as:

$$c(k) = \begin{cases} 0 & \text{if } m_k \in M_i \\ 1 & \text{if } m_k \in M \setminus M_i \\ 0 & \text{if } l < k \leq l+r, \end{cases} \quad (9)$$

where l is the number system measurements and r is the number of MSOs in the system. Using the algorithm proposed in [Krysander *et al.*, 2008a] 165 MSOs are generated for the running example, the four tank system. Since there are 8 measurements in the system c is a vector with 173 elements for this example.

Consider subsystem S_i with local faults F_i and the set of system faults, F . Each local fault $f_j \in F_i$ has to be locally detectable. Given definition 8, we can guarantee local detectability of all the faults $f_j \in F_i$ with the following constraints in the optimization problem (5).

$$A(j, k) = \begin{cases} 0 & \text{if } k < l \\ -1 & \text{if } f_j \in M\mathcal{S}\mathcal{O}_{k-l} \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

Note that l is the number of measurements in the system. By considering $b(j) = -1$ for $1 \leq j \leq g$, where g is the number of faults in F_i , we make sure that we have selected at least one MSO to detect each fault.

To address isolability requirement we follow the same procedure. To isolate $f_j \in F_i$ from any other fault in system, i.e., $f_h \in F$ we need to have:

$$A(j+g, k) = \begin{cases} 0 & k < l \\ -1 & f_j \in M\mathcal{S}\mathcal{O}_{k-l}, f_h \notin M\mathcal{S}\mathcal{O}_{k-l} \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

Setting $b(j) = -1$ for $g < j \leq g * h$, where h is the number of faults in the system, $h = |F|$, we make sure that there is at least one MSO to isolate each of the subsystem faults from the other faults in the system.

In addition to the constraints that guarantee maximum detectability and isolability for the distributed diagnosis system, we need a set of constraints that capture the relationship between the measurements and MSOs in the distributed diagnosis system. Using a MSO is equivalent to using the measurements that are included in the MSO, and we need to include this in the optimization problem. For example, consider $M\mathcal{S}\mathcal{O}_{11}$, it has three measurements $M_{11} = \{u_1, y_1, y_2\}$. Using $M\mathcal{S}\mathcal{O}_{11}$ in a local diagnosis subsystem means we need to communicate these measurement streams to that subsystem to achieve global diagnosability for the

mixed-integer-linear-programming-algorithms.html in the MatlabTMlinear integer programming toolbox.

faults that belong to that subsystem. The following equation represents this constraint.

$$-x(1) - x(2) - x(3) + |M_1|x(7) \leq 0, \quad (12)$$

where $|M_1| = 3$ is the cardinality number of M_1 and $x(1)$, $x(2)$, $x(3)$ and $x(7)$ are binary variables that are 1 if we use u_1 , y_1 , y_2 and MSO_{11} in the diagnosis system and are zero otherwise. This constraint implies that if we use MSO_1 : $x(7) = 1$, its associated measurements are used by the subsystem too: $x(1) = x(2) = x(3) = 1$.

Equation (13) represents these set of constraints in A matrix.

$$A(j + g * h, k) = \begin{cases} -1 & \text{if } m_k \in MSO_j \\ |M_j| & \text{if } k = j + |M| \\ 0 & \text{otherwise,} \end{cases} \quad (13)$$

where $|M_j|$ is the cardinality number of set of measurements in MSO_j and $|M|$ is the cardinality number of set of all the measurements in the system. Setting $b(j) = 0$ for $g * h < j \leq g * h + n$, where n is the number of MSOs in the system. The optimization problem takes into account the relationship between measurements and MSOs. For the running example we generated 165 MSOs, there are also 3 measurements in the subsystem 1, and 8 measurements for the entire system. Similarly, subsystem 1 has two faults of interest, and the goal is to be able to isolate them from any of the 6 faults in the complete system. Therefore, to solve the optimization problem (5) for subsystem 1, matrix A has 177 rows (equal to the number of constraints: 2 constraints to guarantee the local detectability of f_1 and f_2 , 10 constraints to guarantee the local isolability of f_1 and f_2 from the other faults, and 165 constraints to capture the relationship between the MSOs and the measurements) and 173 columns (equal to the number of binary variables: 8 for the measurements and 165 for the MSOs) and b is a vector with 177 elements (equal to the number of constraints).

Table 1 shows the set of measurements that we need to add for each of the subsystem diagnosers to achieve maximum possible detectability and isolability using our proposed algorithm. To find the optimum measurements, we solved the optimization problem (5) for each subsystem.

Table 1: Set of augmented measurements to each subsystem model

Subsystem	Set of augmented measurements
S_1	y_3
S_2	u_2, y_2, y_6
S_3	y_4, y_6
S_4	y_5

Considering the expanded measurement set the schematic of the four tank system with the four distributed diagnosers is shown in Figure 2. The figure shows the complete set of measurements required by the four subsystem diagnosers to achieve global detectability and isolability for the set of faults they contain. For example subsystem 1 includes three measurements $M_1 = \{u_1, y_1, y_2\}$, and to achieve global diagnosability for its faults, y_3 must be communicated to its diagnoser from subsystem 2. Subsystem 2 is the only subsystem that shares a variable with a second order connected

subsystem, all the other subsystems only need to communicate with their first order connected subsystems. Note that communicated measurements typically will incur additional cost and may lower reliability of the system diagnoser. But keeping them to a minimum (see results in Table 1) reduces that cost and uncertainty, while maintaining global diagnosability.

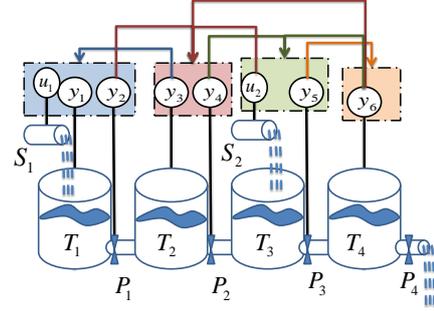


Figure 2: Distributed diagnosis subsystems.

A common way to validate a distributed fault detection and isolation approach is to compare the result with the maximum global detectability and isolability. Adopting the exoneration assumption, Table 2 shows the detectability and isolability performance of the centralized approach. An X in the table shows that the fault in the row and the fault in the column are not isolable from each other. An X in the first column (NF) means the fault in the corresponding row is not isolable from NF (No Fault) or simply it is not detectable. Table 2 shows that with a centralized approach we can detect and isolate all the faults.

Table 2: Fault isolability table for running example using centralized approach

	NF	f_1	f_2	f_3	f_4	f_5	f_6
f_1	X						
f_2			X				
f_3				X			
f_4					X		
f_5						X	
f_6							X

However, Table 3 shows that using the original subsystems for distributed diagnosis does not provide the same results as the centralized global diagnoser.

Table 3: Fault isolability table for running example using distributed approach for the original subsystems

	NF	f_1	f_2	f_3	f_4	f_5	f_6
f_1	X						
f_2	X	X	X	X	X	X	X
f_3	X	X	X	X	X	X	X
f_4	X	X	X	X	X	X	X
f_5	X	X	X	X	X	X	X
f_6	X	X	X	X	X	X	X

In fact, only f_1 can be detected and isolated from the other faults. Using the augmented subsystems in Table 1

Table 4: Fault isolability table for running example using distributed approach for the augmented subsystems

NF	f_1	f_2	f_3	f_4	f_5	f_6
f_1	X					
f_2		X				
f_3			X			
f_4				X		
f_5					X	
f_6						X

(Figure 2) we achieve the same performance as the global diagnoser as shown in Table 4.

This demonstrates that the distributed approach can achieve the same performance with the centralized approach for fault detection and isolation in the running example. In general, the worst case scenario for a system with strongly connected subsystems (i.e., all subsystems are connected to each other) will typically require a large number of measurements from other subsystems to be communicated to each subsystem diagnoser. In those situations, subsystem diagnosers just get rid of the single point of failure, but each subsystem diagnoser may require a large number of measurements to be communicated to it from all of the other subsystems.

In our case study, the four tank system model included 165 MSOs, which means for each subsystem there was 2^{165} different MSO candidate sets. This creates a very large search space (in general the search space is exponential in the number of MSOs, and generating all MSOs is in itself an exponential problem. This justifies the formulation of the problem as a BILP problem that provides efficient tools, like the *bintprog* function in MatlabTM (see earlier footnote), to solve it. However, given the exponential nature of the solution, this method will not scale up for larger systems, even if the subsystem diagnoser design is performed off-line. In addition to the computational complexity, the availability of global models for large, complex systems is unlikely because of the issues discussed in Section 1. To overcome this problem, we sacrifice minimality of the solution to some extent, and propose an incremental algorithm for designing the subsystem diagnosers.

5 MSOs Selection for Distributed Fault Detection Using Neighboring Subsystems

The proposed approach in the previous section used the global model of the system to generate the residuals, and then derived the subsystem diagnosers using the BILP algorithm run on the global MSO set. In this section, we achieve global diagnosability of a subsystem diagnoser by incrementally adding a minimum number of measurements from the neighbors of this subsystem till the global diagnosability property is established. The algorithm starts with the set of equations for the subsystem whose diagnoser is being designed, and if global diagnosability is not achieved using this model, it expands to include equation sets that correspond to the models of its immediate neighbors. If global diagnosability is achieved, the algorithm terminates, otherwise the algorithm expands to use the next higher order of neighbors and repeats the search for minimal MSOs to achieve complete diagnosability. The process of including successively higher order neighbors is shown in Figure 3.

In the worst case, this process continues, till the complete set of system equations are required to generate all possible MSOs, and establish global diagnosability for the subsystem. Therefore, it is guaranteed that the method has the same diagnosability performance as the best centralized diagnoser for the same set of measurements. Algorithm 1 de-

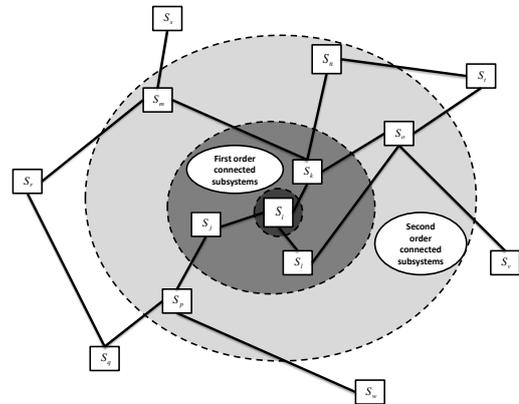


Figure 3: Expanding the search environment to the higher order connected subsystems.

scribes the algorithm for our proposed method.

Algorithm 1 Incremental Algorithm

- 1: **for each** $S_i \in S$ **do**
 - 2: $SS = S_i$
 - 3: $j = 0$
 - 4: **while** $D_i(SS) \neq D_i(S)$ or $I_i(SS) \neq I_i(S)$ **do**
 - 5: $j = j + 1$
 - 6: $SS = SS \cup (j\text{th order connected subsystems of } S_i)$
 - 7: Generate all the MSOs for SS
 - 8: Use equation (9) to compute cost function for SS
 - 9: Use equations (10), (11), and (13) to generate A matrix for SS
 - 10: Generate vector b for SS
 - 11: Use *bintprog*(c, A, b) to solve the problem and compute $D_i(SS)$ and $I_i(SS)$
-

Consider the running example. To design the diagnosis system for the first subsystem, we start with its set of equations and we can only generate one MSO which is not enough to detect subsystem faults and isolate them from the system faults. We then augment the subsystem model with the model from its nearest neighbor subsystem 2, and generate the set of MSOs for the augmented model. The total number of MSOs for the augmented subsystem (Subsystem 1 + subsystem 2) is 11 which leads to 2^{11} MSO set candidates which is much smaller than 2^{165} candidates. Solving the optimization problem presented in this section gives the same result with the global method for this subsystem, but the computation time is reduced significantly. Using the same approach for every subsystem, the set of measurements that we need to transfer to each subsystem of the running example are presented in Table 5.

Figure 4 shows that for the four tank case study, all the subsystems share variables with their first order connected subsystems. This provides a practical advantage to this al-

Table 5: Set of augmented measurements to each subsystem model

Subsystem	Set of augmented measurements
S_1	y_3
S_2	u_2, y_2, y_5
S_3	y_4, y_6
S_4	y_5

gorithm because usually the subsystems with shared variables are physically closer to each other (corresponding to our definition of nearest neighbors) and, therefore, we do not need to transfer data over long distances, which, as discussed earlier, can be costly and error-prone.

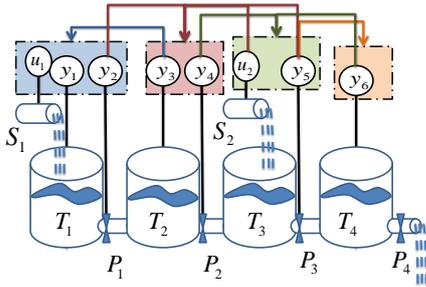


Figure 4: Distributed diagnosis subsystems using incremental algorithm.

Table 6 shows that this distributed diagnosis system provides the same diagnosability performance as the centralized diagnosis method.

Table 6: Fault isolability table for running example using the incremental algorithm

NF	f_1	f_2	f_3	f_4	f_5	f_6
f_1	X					
f_2		X				
f_3			X			
f_4				X		
f_5					X	
f_6						X

The proposed algorithm provides the maximum possible detectability and isolability that can be achieved. The advantage of this algorithm is that not only we do not need a global model for detecting and isolating the faults, but also we do not use the global model in the design process of the supervisory system. This makes the approach suitable for large, complex systems, such as aircraft and power plants where the global systems models are likely to be unavailable or unknown.

6 Discussion and Conclusions

A distributed approach to the problem of fault detection and isolation is presented in this paper. We proposed two algorithms for MSOs selection for the distributed diagnosis. The proposed algorithms provide the maximum possible detectability and isolability that can be achieved for a system given a set of measurements. The first algorithm also guarantees that the subsystems share the minimum number

of measurements, implying that we minimize the communication of measurement streams across subsystems of the global system. This is important because sending the data to other subsystems is costly in large scale systems. On the other hand, the second algorithm does not need to use the global model in the design process of the supervisory system. This makes the algorithm more practical, specially for the complex systems. However, the second algorithm does not guarantee that the number of shared variables among the subsystems are globally minimum.

Unlike previous work, such as [Bregon *et al.*, 2014; Daigle *et al.*, 2007] this method directly works with MSOs generated from subsystem and system equations, and therefore, does not need to use the temporal response and event ordering in the diagnosis, all of which are derived properties, and, therefore, require additional computation. Using a purely structural approach, reduces the overall diagnosability of the system for the given set of measurements. However, it also reduces the number of assumptions we need to make about the fault characteristics, order of events in the diagnoses subsystems (which can be error-prone), and we do not have to analyze in detail the subsystem dynamics.

Moreover, in the incremental algorithm we do not need to have the full global model to design the individual subsystem diagnosers. This is an important practical contribution of this paper in comparison to our previous work (e.g., [Roychoudhury *et al.*, 2009]). Requiring the global model may render the approach to be impractical for the large-scale complex systems, such as aircraft and power plants where the global systems models are likely to be unavailable or unknown.

Finally, in the proposed methods, we generate the MSOs first to design our subsystem diagnosers. The total number of MSOs is exponential in terms of the system measurements. This increases the computational cost of the problem. To make our diagnoser derivation process more efficient, we used BILP framework. On the other hand, having all the MSOs beforehand, makes robustness analysis [Khorasgani *et al.*, 2014a; Khorasgani *et al.*, 2014b] possible for robust distributed MSOs selection. In future work, we will consider noise and uncertainty in the system and will extend the proposed method to robust distributed fault detection and isolation.

References

- [Bregon *et al.*, 2014] Anibal Bregon, Matthew Daigle, Indranil Roychoudhury, Gautam Biswas, Xenofon Koutsoukos, and Belarmino Pulido. An event-based distributed diagnosis framework using structural model decomposition. *Artificial Intelligence*, 210:1–35, 2014.
- [Daigle *et al.*, 2007] M. J. Daigle, X. D. Koutsoukos, and G. Biswas. Distributed diagnosis in formations of mobile robots. *Robotics, IEEE Transactions*, 23(2):353–369, 2007.
- [Deb *et al.*, 1998] S. Deb, A. Mathur, P. K. Willett, and K. R. Pattipati. Decentralized real-time monitoring and diagnosis. In *Systems, Man, and Cybernetics. IEEE International Conference*, 3:2998–3003, 1998.
- [Debouk *et al.*, 2000] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete*

Event Dynamic System: Theory and Applications, 10(1-2):33–86, 2000.

- [Duarte Jr and Nanya, 1998] E. P Duarte Jr and T. Nanya. A hierarchical adaptive distributed system-level diagnosis algorithm. *Computers, IEEE Transactions*, 47(1):34–45, 1998.
- [Ferrari et al., 2012] Riccardo MG Ferrari, Thomas Parisini, and Marios M Polycarpou. Distributed fault detection and isolation of large-scale discrete-time nonlinear systems: An adaptive approximation approach. *Automatic Control, IEEE Transactions on*, 57(2):275–290, 2012.
- [Indra et al., 2012] Saurabh Indra, L. Trave-Massuyes, and Elodie Chanthery. Decentralized diagnosis with isolation on request for spacecraft. Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (Safeprocess), Mexico City, 2012.
- [Khorasgani et al., 2014a] Hamed Khorasgani, Daniel E. Jung, Gautam Biswas, Erik Frisk, and Mattias Krysander. Off-line robust residual selection using sensitivity analysis. International Workshop on Principles of Diagnosis (DX-14), Graz, Austria, 2014.
- [Khorasgani et al., 2014b] Hamed Khorasgani, Daniel E. Jung, Gautam Biswas, Erik Frisk, and Mattias Krysander. Robust residual selection for fault detection. Decision and Control (CDC), IEEE 53rd Annual Conference on, 2014.
- [Krysander et al., 2008a] Mattias Krysander, Jan Åslund, and Mattias Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 38(1), 2008.
- [Krysander et al., 2008b] Mattias Krysander, Jan Aslund, and Mattias Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):197–206, 2008.
- [Lafortune, 2007] S. Lafortune. On decentralized and distributed control of partially-observed discrete event systems. in advances in control theory and applications. *Springer Berlin Heidelberg*, pages 171–184, 2007.
- [Land and Doig, 1960] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.
- [Lanigan et al., 2011] Patrick E. Lanigan, Soila Kavulya, and Priya Narasimhan. Diagnosis in automotive systems: A survey. *Technical Report CMU-PDL-11-110, Carnegie Mellon University PDL*, 2011.
- [Leger et al., 1999] J. B. Leger, B. Iung, A. Ferro De Beca, and J. Pinoteau. An innovative approach for new distributed maintenance system: application to hydro power plants of the remafex project. *Computers in industry*, 38(2):131–148, 1999.
- [Mitchell, 2002] John E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, pages 65–77, 2002.
- [Rish et al., 2005] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K Hernandez. Adaptive diagnosis in distributed systems. *Neural Networks, IEEE Transactions*, 16(5):1088–1109, 2005.
- [Rosich et al., 2009] Albert Rosich, Ramon Sarrate, and Fatiha Nejari. Optimal sensor placement for fdi using binary integer programming. International Workshop on Principles of Diagnosis, 2009.
- [Roychoudhury et al., 2009] Indranil Roychoudhury, Gautam Biswas, and Xenofon Koutsoukos. Designing distributed diagnosers for complex continuous systems. *Automation Science and Engineering, IEEE Transactions on*, 6(2):277–290, 2009.
- [Sarrate et al., 2007] Ramon Sarrate, Puig Vicenc, Escobet Teresa, and Rosich Albert. Optimal sensor placement for model-based fault detection and isolation. pages 2584–2589. 46th IEEE Conference In Decision and Control, 2007.
- [Shum et al., 1988] S. K. Shum, J. F. Davis, W. F. Punch III, and B. Chandrasekaran. An expert system approach to malfunction diagnosis in chemical plants. *Computers and chemical engineering*, 12(1):27–36, 1988.
- [Svard et al., 2012] C. Svard, M. Nyberg, and J. Stoustrup. Automated design of an fdi system for the wind turbine benchmark. *JCSE Journal of Control Science and Engineering*, 2012.
- [Wolsey and Nemhauser, 2014] Laurence A. Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley and Sons, 2014.
- [Wolsey, 1998] Laurence A. Wolsey. *Integer programming*, volume 42. Wiley, New York, 1998.

Condition-based Monitoring and Prognosis in an Error-Bounded Framework

L. Travé-Massuyès^{1,2} and R. Pons^{1,2} and P. Ribot^{1,3} and Y. Pencole^{1,2} and C. Jaubertie^{1,3}

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

²Univ de Toulouse, LAAS, F-31400 Toulouse, France

³Univ de Toulouse, Université Paul Sabatier, LAAS, F-31062 Toulouse, France

e-mail: louise@laas.fr, renaud.pons@gmail.com, pribot,ypencole,cjaubert@laas.fr

Abstract

Condition-based maintenance is recognized as a better health management strategy than regularly planned inspections as used nowadays by most companies. In practice, it is however difficult to implement because it means being able to predict the time to go before a failure occurs. This prediction relies on knowing the current health status of the system's components and on predicting how components age. This paper demonstrates the applicability of interval-based tools in integrated health management architectures, hence proposing an alternative to the standard statistical approach¹.

Keywords: Interval analysis, diagnosis, prognosis.

1 Introduction

Nowadays system's availability is a key ingredient of economical competitiveness. A typical example is the civil aircraft industry for which the unavailability of passenger carriers generate great costs and considerable economical losses. Technical inspections are generally planned on regular bases. If a component fails and needs to be replaced between two successive inspections, the plane is taken to a standstill and the company has to re-schedule the aircraft fleet, implying money loss during this unplanned immobilization. This is why condition-based maintenance is a preferable strategy that means predicting at inspection time the time to go before a new failure occurs. In this case the aircraft company can replace the part whose failure is estimated during the current inspection and then prevent an extra immobilization of the plane. This strategy not only saves a lot of money but also increases reliability and safety.

Integrated systems health management architectures performing condition-based monitoring naturally couple fault diagnosis and prognosis mechanisms [1; 2; 3; 4]. Diagnosis is used to assess the current state of the system and is used to initialize a prediction mechanism based on ageing models that aims to estimate the *remaining useful life* (RUL). In the prognosis process several sources of uncertainty can be identified, in particular the ageing models and the future

stress conditions. These uncertainties are commonly taken into account through appropriate assumptions about noise and model error distributions, which are difficult to acquire. An alternative approach is to frame the problem in a set-membership framework and make use of recent advances in the field of interval analysis and interval constraint propagation.

This paper demonstrates the applicability of interval-based tools — briefly introduced in Section 2 — in integrated health management architectures, providing an interesting alternative to the standard statistical approach [5; 6]. It proposes a two stages *set-membership* (SM) *condition-based monitoring* method whose principle is presented in Section 3. The first stage is diagnosis that provides an estimation of the system's health status. It takes the form of SM parameter estimation using *Focused Recursive Partitioning* (FRP) and is the subject of Section 4. The second stage concerns prognosis in the form of the estimation of the remaining system's lifespan. It is based on the use of a *damaging table* and is detailed in Section 5. The case study of a shock absorber is used to illustrate the method and is presented in Section 6 before the concluding Section 7.

2 Interval analysis

Interval analysis was originally introduced to obtain guaranteed results from floating point algorithms [7] and it was then extended to validated numerics [8]. A *guaranteed result* first means that the solution set encloses the actual solution. It also means that the algorithm is able to conclude about the existence or not of a solution in limited time or number of iterations [9].

2.1 Interval

A real interval $x = [\underline{x}, \bar{x}]$ is a closed and connected subset of \mathbb{R} where \underline{x} and \bar{x} represent the lower and upper bound of x , respectively. \underline{x} and \bar{x} are real numbers. The *width* of an interval x is defined by $w(x) = \bar{x} - \underline{x}$, and its *midpoint* by $mid(x) = (\bar{x} + \underline{x})/2$. If $w(x) = 0$, then x is degenerated and reduced to a real number. x is defined as positive (resp. negative), *i.e.* $x \geq 0$ (resp. $x \leq 0$), if $\underline{x} \geq 0$ (resp. $\bar{x} \leq 0$).

The set of all real intervals of \mathbb{R} is denoted \mathbb{IR} . Two intervals x_1 and x_2 are equal if and only if $\underline{x}_1 = \underline{x}_2$ and $\bar{x}_1 = \bar{x}_2$. Real arithmetic operations have been extended to intervals [8]:

$$\circ \in \{+, -, *, /\}, x_1 \circ x_2 = \{x \circ y \mid x \in x_1, y \in x_2\}.$$

An *interval vector* or *box* \mathbf{x} is a vector with interval components. An *interval matrix* is a matrix with interval components. The set of n -dimensional real interval vectors is

¹This work was supported by the CORALIE Project IA 2012–01–06 of the Council for Research in French Civil Aeronautics (CORAC), WP1 "Contrôle Santé", and by the French National Research Agency (ANR) in the framework of the project ANR-11-INSE-006 (MAGIC-SPS).

denoted by \mathbb{R}^n and the set of $n \times m$ real interval matrices is denoted by $\mathbb{R}^{n \times m}$. The width $w(\cdot)$ of an interval vector (or of an interval matrix) is the maximum of the widths of its interval components. The midpoint $mid(\cdot)$ of an interval vector (resp. an interval matrix) is a vector (resp. a matrix) composed of the midpoints of its interval components.

Classical operations for interval vectors (resp. interval matrices) are direct extensions of the same operations for real vectors (resp. real matrices) [8].

2.2 Inclusion function

Given \mathbf{x} a box of \mathbb{R}^n and a function f from \mathbb{R}^n to \mathbb{R}^m , an *inclusion function* of f aims at getting a box containing the image of \mathbf{x} by f . The *range* of f over \mathbf{x} is given by:

$$f(\mathbf{x}) = \{f(\nu) \mid \nu \in \mathbf{x}\},$$

where ν is a real vector of the same dimension as \mathbf{x} . Then, the interval function $[f]$ from \mathbb{R}^n to \mathbb{R}^m is an *inclusion function* for f if:

$$\forall \mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}) \subset [f](\mathbf{x}).$$

An inclusion function of f can be obtained by replacing each occurrence of a real variable by its corresponding interval and by replacing each standard function by its interval evaluation. Such a function is called the *natural inclusion function*. A function f generally has several inclusion functions, which depend on the syntax of f .

2.3 Notations

Throughout the paper and unless explicitly mentioned, variables are assumed to take values in \mathbb{R}^d , where d is the dimension of the variable. Exception is made for overlined and underlined variables that are assumed to take values in \mathbb{R}^d , where d is the dimension of the variable. Bold symbols are used to denote multi-dimensional variables (vector or matrices).

3 Principle of the Set-Membership Health Management Method

3.1 Method Architecture

The architecture of the preventive maintenance method is shown in Fig. 1. The method relies on two modules:

- A *diagnosis* module that uses the system measured inputs and outputs to compute an estimation of the system's health status; this is performed by estimating the value of the system parameter vector θ by the means of a *behavioral Model* Σ of the system.
- A *prognosis* module that predicts the parameter evolution over time by using a *Damaging Model* Δ and computes the *Remaining Useful Life* or RUL of the underlying subsystems.

The global model representing the progressive evolution of the system over time, *e.g.* the *Ageing Model*, is obtained by putting together the *behavioral model* Σ and the *damaging model* Δ . The behavioral model takes the form of a state space model, *i.e.* a state equation modeling the dynamics of the system state vector, and an observation equation that links the state variables to the observed variables:

$$\Sigma : \begin{cases} \dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t), \theta(t), \mathbf{u}(t)) \\ \mathbf{y}(t) = h(t, \mathbf{x}(t), \theta(t), \mathbf{u}(t)) \end{cases} \quad (1)$$

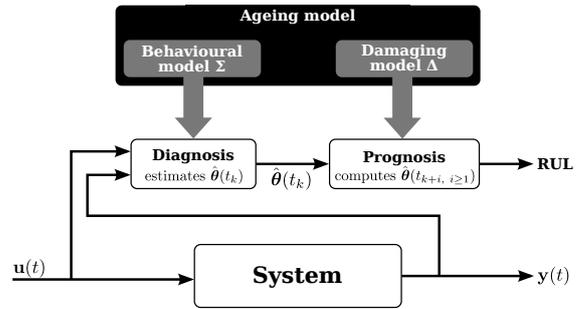


Figure 1 – Health management architecture.

where

$\mathbf{x}(t)$ is the state vector of the system of dimension n_x ,

$\mathbf{u}(t)$ is the input vector of dimension n_u ,

$\mathbf{y}(t)$ is the output vector of dimension n_y ,

$\theta(t)$ is the parameters vector of dimension n_θ .

Σ represents the system's nominal behavior as it is supposed to act when its parameters have not yet suffered any ageing.

The damaging model Δ represents the dynamics of the behavioral model parameters. It is described by the dynamic state equation (2) where the equation states are the system parameters. The equation models how the parameters evolve over time because of the wearing, leakage, etc.:

$$\Delta : \dot{\theta}(t) = g(t, \theta(t), \mathbf{w}, \mathbf{x}(t)) \quad (2)$$

where \mathbf{w} is a wearing parameter vector of dimension n_θ .

3.2 Unit Cycles

Predicting the evolution of the system's behavior requires to know *a priori* how the system will be solicited either by the control system or by external causes (*e.g.* environmental conditions, temperature, humidity, etc.) This knowledge is generally difficult to obtain. In our approach, we make assumptions about the future solicitations of the system by determining the most usual way the system is intended to be used and we define the notion of *unit cycles*. A unit cycle \mathcal{C} is defined as a solicitation that repeats in time and that leads to a behavioral sequence that is known to impact system's ageing.

For example, in the case of a pneumatic valve from the Space Shuttle cryogenic refueling system, [2] defines a unit cycle as the opening of the valve, the filling of the tank and the valve closing when the tank is full. In the case of an aircraft, an unit cycle may be chosen to be a flight: it starts with the plane take-off, a cruising stage and landing.

One may simultaneously use unit cycles at different time scales, depending on the dynamics of the system and its subsystems. As an example, one may define a "global" unit cycle for a bus as being the journey from the starting station to the terminus, and another unit cycle for the subsystem "bus doors" as being the opening and the closing of the doors at each station.

4 SM Diagnosis

Diagnosis is achieved through SM parameter estimation. This problem assumes that measured outputs $\mathbf{y}_m(t_i)$ generated by the real system on a time horizon $t_i = t_0, \dots, t_H$ of length $H \times \delta$, where δ is the sampling period, are corrupted by bounded-error terms that may originate from the system

parameters varying within specified bounds, bounded noise, or sensor precision. The $\mathbf{y}_m(t_i)$'s are hence interval vectors of \mathbb{R}^{n_y} . The SM parameter estimation problem for the system Σ is formulated as finding the set $\Theta \subseteq \mathbb{R}^{n_\theta}$ of real parameter vectors such that the arising outputs $\mathbf{y}(t_i, \theta) \in \mathbb{R}^{n_y}$ hit all the output data sets, *i.e.*:

$$\theta \in \Theta \Leftrightarrow \mathbf{y}(t_i, \theta) \in \mathbf{y}_m(t_i), \forall t_i \in \{t_0, \dots, t_H\}.$$

Θ is called the *feasible parameter set* (FPS). SM parameter estimation problems are generally solved with a branch-and-bound algorithm like SIVIA [10] that enumerates candidate box solutions thanks to a rooted tree and assumes the full parameter space as the root. At every node, the set of $\hat{\mathbf{y}}(t_i), t_i = t_0, \dots, t_H$, arising from the considered box parameter vector $[\theta]^*$, *i.e.* solution of Σ for any real $\theta \in [\theta]^*$, is checked for consistency against the measurements and labelled feasible, unfeasible or undetermined. Unfeasible candidates are rejected while undetermined candidates are split and checked in turn until the set precision of the candidate solutions is below a given threshold ε provided by the user.

Such algorithms return an overestimation of the FPS given by the convex union of the candidates that have been labelled feasible and undetermined [11]. Interestingly, the convex union may consist of one set or more, which means that the systems does not need to be identifiable in the classical sense [12].

When considering a SIVIA-based algorithm for dynamical systems like Σ , a critical step is the determination of the inclusion function for the state vector $\hat{\mathbf{x}}(t_i)$ at instants $t_i = t_0, \dots, t_H$, arising from a given candidate parameter vector $[\theta]^*$, from which the $[\hat{\mathbf{y}}(t_i)], t_i = t_0, \dots, t_H$ can be computed using the observation equation of Σ . This step relies on set-membership integration for which we have chosen the interval Taylor series integration scheme implemented in the VNODE-LP solver [13]. Although quite well optimized [14], it is well-known that this method is computationally stable only for $[\theta]^*$ of very small size. SIVIA-like parameter estimation algorithms are hence particularly inefficient as they enumerate candidate parameter subspaces starting with the full parameter space. This is why we propose the FRP schema presented in the next section.

4.1 Principle of FRP-based SM Parameter Estimation

The principle of the FRP method is based on partitioning the parameter search space $\mathcal{S}(\theta)$. Each part of the partition represents a candidate parameter vector $[\theta]_j$ for which SM integration of the state equation of Σ provides a conservative numerical enclosure $\hat{\mathbf{x}}(t_i)_j, t_i = t_0, \dots, t_H$. The output vector can now be estimated as:

$$\hat{\mathbf{y}}(t_i)_j = h(t, \hat{\mathbf{x}}_j, [\theta]_j, \mathbf{u}_m), \forall t_i \in \{t_0, \dots, t_H\} \quad (3)$$

We then keep track of the parameters vectors for which the $\hat{\mathbf{y}}(t_i)_j$'s are consistent with the measurements, for all $t_i = t_0, \dots, t_H$, *i.e.* the unfeasible ones are discarded. Computing the convex hull then provides us with a minimal and maximal value for the admissible parameter vectors.

The consistency test is defined as testing the intersection of the estimated output vector with the measurements:

$$\text{If } \exists t_i \in \{t_0, \dots, t_H\} \text{ s.t. } \hat{\mathbf{y}}(t_i)_j \cap \mathbf{y}_m(t_i) = \emptyset, \quad (4)$$

then there is no consistency between the estimation and the measured input $\mathbf{u}_m(t)$ and output $\mathbf{y}_m(t)$ with the tested parameter vector $[\theta]_j$. The parameters box $[\theta]_j$ is unfeasible

and hence rejected.

$$\text{If } \hat{\mathbf{y}}(t_i)_j \subseteq \mathbf{y}_m(t_i), \forall t_i \in \{t_0, \dots, t_H\}, \quad (5)$$

then $[\theta]_j$ is a parameter vector for which the estimation is consistent with the measurements. The box is added to the list of the solution parameter boxes:

$$\mathbb{P} = \mathbb{P} \cup [\theta]_j. \quad (6)$$

If none of the two previous conditions is true, *i.e.*:

$$[\hat{\mathbf{y}}(t_i)_j] \cap [\mathbf{y}_m(t_i)] \neq \emptyset, \forall t_i \in \{t_0, \dots, t_H\}, \quad (7)$$

it means that the parameter box $[\theta]_j$ is undetermined and that it *partially* contains solutions. The box is also added to \mathbb{P} . Two different labels allow us to keep track of the boxes that are feasible or undetermined. The convex union of these boxes provides the estimation $\hat{\theta}$ that encloses the feasible parameter set Θ , *i.e.* $\hat{\theta} \supseteq \Theta$.

The quality of the enclosure depends on the size of the boxes of the partition, in other words on the partition precision. A way to improve the enclosure is to proceed with a partition of the obtained solution $\hat{\theta}$ and run another round of consistency tests over the new boxes, and so on recursively. The process of iterating the partition ends when the gain in precision is low with respect to the SM integration and consistency tests computational cost. The method is detailed for a one dimension parameter vector in the next section.

The *estimation precision* $\omega(P)$ obtained for a given partition P can be evaluated by the following percentage:

$$\omega(P) = \left| \text{mid}(\hat{\theta}) \right| ./ \left(\left| \text{mid}(\hat{\theta}) \right| + w(\hat{\theta})/2 \right) \quad (8)$$

where $./$ denotes the division of two vectors term by term.

Given two partitions P_i and P_j , one can evaluate the precision gain as:

$$G(P_j/P_i) = \omega(P_j) ./ \omega(P_i). \quad (9)$$

4.2 Parameter search space

The domain value of the parameter vector θ is given by $\Omega(\theta) = [\text{inf}(\theta_{bol}, \theta_{eol}), \text{sup}(\theta_{bol}, \theta_{eol})]$, where inf and sup denote the operators inf and sup applied term by term and θ_{bol} and θ_{eol} are the real vectors whose components are given by $\theta_{k,bol}$ and $\theta_{k,eol}$ for each parameter $\theta_k, k = 1, \dots, n_\theta$:

- $\theta_{k,bol}$ (*bol*: “beginning-of-life”) is the factory setting defined by the design specification,
- $\theta_{k,eol}$ (*eol*: “end-of-life”) is the maximal/minimal admissible value, *i.e.* the value above/below which the component is considered to have failed and the function is no longer guaranteed.

During the system's life, the impact of ageing results in the parameter vector value evolving in $\Omega(\theta)$. Depending on the impact of ageing, its value may decrease or increase with time:

$$\theta(t_i) = \alpha \theta(t_j), t_i \geq t_j \quad (10)$$

where α is an n_θ dimensional real vector whose components α_k are greater or lower than 1 depending on the impact of ageing on the change direction of the parameter.

We assume a health management strategy, which means that diagnosis (and prognosis) is performed according to a given inspection planning at some chronologically ordered

times of the system's life T_0, \dots, T_F . The parameter vector search space depends on the time and is hence denoted by $\mathcal{S}(\theta)_{T_i} = [\underline{\mathcal{S}}(\theta)_{T_i}, \overline{\mathcal{S}}(\theta)_{T_i}]$, where $T_i \in \{T_0, \dots, T_F\}$. Initially, for the first inspection time T_0 , we set $\mathcal{S}(\theta)_{T_0} = \Omega(\theta)$. Diagnosis then returns the estimated parameter value $\hat{\theta}(T_0)$. For the next inspection time, $\mathcal{S}(\theta)$ is updated by taking the parameter value estimation into account as follows:

- if $\alpha_k > 1$, $\theta_{k,bol}$ is replaced by $\inf(\hat{\theta}_k(T_0), \overline{\theta}_k(T_0))$,
- if $\alpha_k < 1$, $\theta_{k,bol}$ is replaced by $\sup(\hat{\theta}_k(T_0), \overline{\theta}_k(T_0))$,

and one of the bounds of the components of $\mathcal{S}(\theta)_{T_1}$ remains equal to $\theta_{k,eol}$.

In the general case, when considering the inspection time T_i , $\mathcal{S}(\theta)_{T_i}$ is hence obtained with $\hat{\theta}(T_{i-1})$ as follows:

- if $\alpha_k > 1$, $\inf(\hat{\theta}_k(T_{i-1}), \overline{\theta}_k(T_{i-1}))$ is replaced by $\inf(\hat{\theta}_k(T_i), \overline{\theta}_k(T_i))$,
- if $\alpha_k < 1$, $\sup(\hat{\theta}_k(T_{i-1}), \overline{\theta}_k(T_{i-1}))$ is replaced by $\sup(\hat{\theta}_k(T_i), \overline{\theta}_k(T_i))$.

4.3 FRP Parameter Estimation for a Single Parameter

In this section, we consider one single parameter θ whose evolution is monotonically increasing. As an example, let's state that θ is a bearing friction coefficient that grows with the bearing wearing and the clogging of the environment. In the general case, this kind of knowledge must be brought by an expert of the system and/or the manufacturer.

Let us consider the first inspection time and the initial search space $\mathcal{S}(\theta)_{T_0}$ given by the domain value of the parameter $\Omega(\theta) = [\theta_{bol}, \theta_{eol}]$. The search space is partitioned into boxes, in our case intervals (cf. Fig. 2).

The dynamic equation of Σ is integrated on the time window $t_i = t_0, \dots, t_H$, where $t_H = T_0$, as many times as the number of intervals in the partition P_1 . The number of intervals is defined by the partition factor $\epsilon(P_1)$, which equals $1/15$ in our example (cf. Fig. 2). We start with $[\theta]_1 = [\theta_{bol}, \theta_{bol} + pw]$, where $pw = \epsilon(P_1)w(\mathcal{S}(\theta)_{T_0})$ is the width of the partition intervals, then proceed with the subsequent intervals $[\theta]_j$. For each interval, we get an estimation of the state vector at times $t_i = t_0, \dots, t_H$, denoted as $\hat{\mathbf{x}}(t_0 \dots t_N)_j$, and obtain $\hat{\mathbf{y}}(t_0 \dots t_N)_j$ thanks to the observation equation (1). This latter is tested for consistency against the measurements $\mathbf{y}_m(t_0 \dots t_N)$.

Depending on the output of the tests (4), (5), and (7), the parameter interval $[\theta]$ is rejected or added to the solution as feasible or undetermined (red-colored, green-colored, and yellow-colored parts, respectively, in Fig. 2).

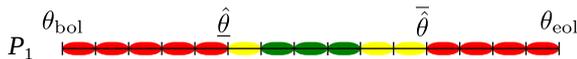


Figure 2 – Partition P_1 and test results for this partition.

The convex union of feasible and undetermined intervals provides a *guaranteed estimation* $\hat{\theta} = [\hat{\theta}, \hat{\theta}]$ of the admissible values for θ . We iterate the process by creating a new partition P_2 of $[\hat{\theta}, \hat{\theta}]$ with a precision $\epsilon(P_2) = 1/10$ (cf. Fig. 3).

We proceed as above for each interval of P_2 in order to refine the bounds of $[\hat{\theta}, \hat{\theta}]$ and find a more precise enclosure of feasible parameter solution (see Fig. 3).

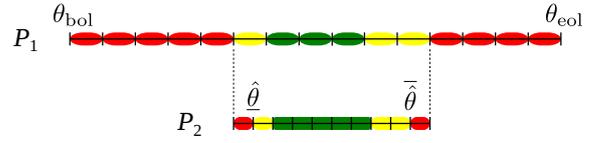


Figure 3 – Partition P_2 and test results for this partition.

We iterate the process until the precision gain $G(P_{i+1}/P_i)$ is greater than a given threshold, as it is shown in Fig. 4.

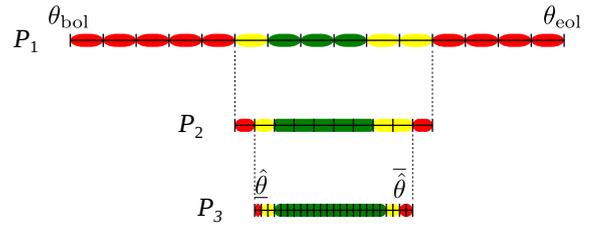


Figure 4 – Test results for partition P_3 .

Remarks

The method can be easily generalized to a system whose parameter vector has dimension $n_\theta > 1$. The computing cost is proportional to the number of boxes that are tested, i.e. $\sum_{i=1}^{n_P} 1/\epsilon(P_i)$, where n_P is the number of partitions.

Let's notice that the partition may be non-regular. For example, for a slowly ageing parameter, one may choose small boxes for the values of θ that are close to θ_{bol} and larger ones for the values close to θ_{eol} . The result is guaranteed even if the partition has not been properly chosen or if the parameter has evolved in a non expected way, although the computation cost may be higher.

The convex union provides a poor result if the set of admissible values is made of several mutually disjoint connected sets, as shown in Fig. 5. The algorithm may test some boxes that have already been rejected by the tests of the previous partition. This drawback could be addressed by defining the solution as a list of boxes whose labels (unfeasible, feasible, or undetermined) are inherited by the next partition boxes.

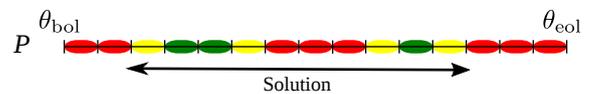


Figure 5 – The returned solution is the convex hull of mutually disjoint connected intervals.

5 SM prognosis

The prognosis phase consists in calculating the number of cycles remaining before anomaly, which is also called the *Remaining Useful Life* or RUL. To optimally adapt this calculation to the system's life requires the knowledge of the health status of the system at the current time, which was the topic of Section 4.

5.1 Component degradation

The global model ($\Sigma + \Delta$) assumes that the parameters of the behavior model Σ given by (1) evolve in time, and that their evolution is represented by the degradation model Δ given by the dynamic equation (2) that is recalled below:

$$\Delta : \dot{\theta}(t) = g(t, \theta(t), \mathbf{w}, \mathbf{x}(t)).$$

Δ provides the dynamics of the parameter vector as a function of the state of the system $\mathbf{x}(t)$ and of a degradation parameter vector \mathbf{w} that allows one to tune the degradation for each of the considered parameters.

The global model ($\Sigma + \Delta$), in the form of a dynamic model with varying parameters, cannot be directly integrated by VNODE-LP. An original method, coupling the two models Σ and Δ iteratively is proposed in the following. The method is illustrated by Fig. 6 and used to determine the degradation suffered by each parameter during one unit cycle as defined in Section 3.2.

Let us denote $\mathbf{u}_{\mathcal{C}}(t)$, $t \in [\tau, \tau + d_{\mathcal{C}}]$, the system input stress during one unit cycle \mathcal{C} . As shown in Fig. 6, the following steps are iteratively executed, every iteration corresponding to a computation step given by the sampling period δ :

1. The normal behavior model Σ is used first with input $\mathbf{u}(t) = \mathbf{u}_{\mathcal{C}}(\tau)$ to compute the state $\mathbf{x}(\tau)$ and the output $\mathbf{y}(\tau)$;
2. The parameters are updated with the degradation model Δ using the value of the state determined previously, *i.e.* $\theta(\tau)$ is computed;
3. The parameters of the behavior model Σ are updated with $\theta(\tau)$;
4. The next stress input value $\mathbf{u}_{\mathcal{C}}(\tau + \delta)$ is considered, and so on until the end of the cycle, *i.e.* until the last value of the cycle $\mathbf{u}_{\mathcal{C}}(\tau + d_{\mathcal{C}})$ is reached.

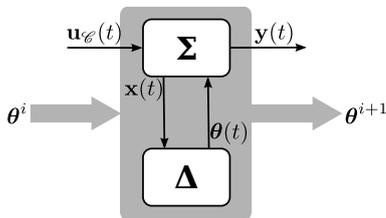


Figure 6 – Computation of the degradation parameters during one unit cycle.

The above algorithm defines the function:

$$\mathcal{D} : \mathbb{R}^{n_{\theta}} \rightarrow \mathbb{R}^{n_{\theta}} \quad (11)$$

where n_{θ} is the number of parameters of the system. Let's assume the cycle i , then \mathcal{D} maps θ^i into $\mathcal{D}(\theta^i) = \theta^{i+1}$, which is the value of θ after one unit cycle.

\mathcal{D} is nonlinear. Thus the value of the parameter vector after one cycle θ^{i+1} depends on the initial value θ^i . Indeed, we know that a system generally degrades in a nonlinear fashion. We must hence compute θ^{i+1} for all possible values of the parameter vector θ^i .

For this purpose, the domain value $\Omega(\theta_k)$ of each parameter θ_k is partitioned into N_k intervals. N_k is chosen sufficiently large to reduce non conservatism of the interval function \mathcal{D} . The domain value of the parameter vector θ is hence

partitioned into $N_{\Pi} = \prod_{k=1}^{n_{\theta}} N_k$ possible boxes that must be fed as input to \mathcal{D} . Let us for instance consider a two parameters vector and its beginning-of-life and end-of-life values as follows:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \theta_{bol} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \theta_{eol} = \begin{bmatrix} 4 \\ 9 \end{bmatrix} \text{ and } N = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad (12)$$

then, if we select the partition landmarks as $\{5\}$ for θ_1 and $\{2, 3\}$ for θ_2 ², \mathcal{D} must be run for the following 6 box values:

$$\begin{aligned} [\theta]_1 &= \begin{bmatrix} [1, 2] \\ [1, 5] \end{bmatrix}, [\theta]_2 = \begin{bmatrix} [2, 3] \\ [1, 5] \end{bmatrix}, [\theta]_3 = \begin{bmatrix} [3, 4] \\ [1, 5] \end{bmatrix}, \\ [\theta]_4 &= \begin{bmatrix} [1, 2] \\ [5, 9] \end{bmatrix}, [\theta]_5 = \begin{bmatrix} [2, 3] \\ [5, 9] \end{bmatrix}, [\theta]_6 = \begin{bmatrix} [3, 4] \\ [5, 9] \end{bmatrix}. \end{aligned} \quad (13)$$

For each of these box values taken as input for cycle i , *i.e.* $\theta^i = [\theta]_l$, $l = 1, \dots, 6$, \mathcal{D} returns the (box) value θ^{i+1} after one unit cycle. This computation is then projected on each dimension to obtain a set of n_{θ} tables, \mathcal{D}_{θ_k} , $k = 1, \dots, n_{\theta}$, that provide the degradation of each individual parameter θ_k after one unit cycle.

5.2 RUL determination

The RUL, understood as a RUL for the whole system, can now be determined by computing the number of cycles that are necessary for the parameters to reach the threshold defining the end-of-life (cf. Fig. 7).

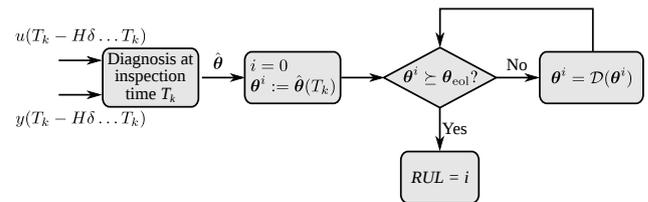


Figure 7 – RUL computation

For the cycle $i = 0$, θ^0 is initialized with $\hat{\theta}$, which is the result of the parameter estimation computed by the diagnosis engine. $\hat{\theta}$ is given as input to \mathcal{D} , which returns $\mathcal{D}(\hat{\theta}) = \theta^1$. i is incremented by 1 and θ^1 is given as input to \mathcal{D} and so on until the set-membership test $\theta^i \geq \theta_{eol}$ is achieved, which provides the stopping condition. This test may take several forms as explained in Section 5.3. If the test is true, then the index i is the number of cycles required to reach the degradation threshold, so $RUL = i$.

For a given cycle i , the box value θ^i that must be given as input to \mathcal{D} is not necessarily among the values $[\theta]_l$, $l = 1, \dots, N_{\Pi}$, of the partition. We propose to compute θ^{i+1} by assuming that the mapping between θ^i and θ^{i+1} is linear in every domain l of the partition. Considering $\mathbf{p} \in \mathbb{R}^{n_{\theta}}$, $\mathcal{D}(\mathbf{p})$ is approximated as follows:

$$\forall \theta \in [\theta]_l, \mathcal{D}(\theta) \approx \mathbf{a} \odot \theta + \mathbf{b}, \quad l=1, \dots, N_{\Pi} \quad (14)$$

where $\mathbf{a} = w(\mathcal{D}([\theta]_l)) / w([\theta]_l)$, $\mathbf{b} = \mathcal{D}([\theta]_l) - \mathbf{a}[\theta]_l$, and \odot is the product of two vectors term by term.

Equation (14) is applied to $\bar{\theta}^i$ and $\underline{\theta}^i$ to obtain an approximation of $\mathcal{D}(\theta^i)$.

²Notice that the intervals issued from the partitioning are not required to be of equal length.

5.3 Set-membership test for the RUL

The set-membership test implemented with the order relation \preceq may take several forms. For instance, if the test $\theta^i \succeq \theta_{eol}$ is interpreted as:

$$\exists k \in \{1, \dots, n_\theta\} \mid \bar{\theta}_k^i \geq \theta_{k,eol} \text{ if } \alpha_k > 1 \text{ or } \bar{\theta}_k^i \leq \theta_{k,eol} \text{ if } \alpha_k < 1, \quad (15)$$

then it means that the bound of the interval value of at least one parameter θ_k is above or below its end-of-life threshold value $\theta_{k,eol}$. The RUL is then qualified as the “worst case RUL”, which means that the RUL indicates the earliest cycle at which the system may fail.

One can also test whether the value higher bound of one of the parameters is higher than its end-of-life threshold, that is to say:

$$\exists k \in \{1, \dots, n_\theta\} \mid \bar{\theta}_k^i \geq \theta_{k,eol} \text{ if } \alpha_k > 1 \text{ or } \bar{\theta}_k^i \leq \theta_{k,eol} \text{ if } \alpha_k < 1. \quad (16)$$

The RUL then represents the cycle at which it is certain that the system will fail.

It is obviously possible to combine these different tests applied to the different individual parameters depending on their criticality.

6 Case study

6.1 Presentation

The case study is a shock absorber that consists of a moving mass connected to a fixed point via a spring and a damper as illustrated by Fig. 8. The movement of the mass takes place in the horizontal plane in order to eliminate the forces due to gravity. Aerodynamic friction forces are neglected.

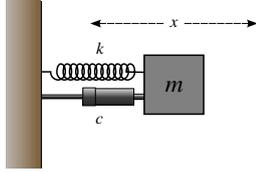


Figure 8 – Spring and damper system

The Newton's second law is written as:

$$m\vec{a} = \Sigma \vec{F} = \vec{F}_r + \vec{F}_c + \vec{u} \quad (17)$$

where m is the mass, \vec{a} is the acceleration, \vec{F}_k is the spring biasing force, \vec{F}_c is the friction force exerted by the damper and \vec{u} is the force applied on the mass. Expressing the forces and the acceleration as a function of the position of the mass $x(t)$, we get:

$$\ddot{x}(t) + \frac{c}{m}\dot{x}(t) + \frac{k}{m}x(t) = u(t) \quad (18)$$

where k is the spring stiffness constant (N/m), m is the mobile mass (kg), and c is the damping coefficient (Ns/m). (18) is a second order ODE. Let us rewrite

$$\frac{c}{m} = 2\zeta\omega_0 \text{ and } \frac{k}{m} = \omega_0^2$$

and we get

$$\omega_0 = \sqrt{\frac{k}{m}} \text{ and } \zeta = \frac{c}{2\sqrt{km}}.$$

The impulse response of such system depends on the value of ζ :

- if $\zeta = 0$, then the answer is a sinusoid;
- if $0 < \zeta < 1$, then the answer is a damped sinusoid;
- if $\zeta \geq 1$, then the answer is a decreasing exponential.

The state model is given by the equation:

$$\begin{cases} \dot{X}(t) = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-c}{m} \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} U(t) \\ Y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} X(t) \end{cases} \quad (19)$$

with $X(t) = [x(t), \dot{x}(t)]^T$, and the transfer function is:

$$\frac{X(p)}{U(p)} = \frac{1}{p^2 + \frac{c}{m}p + \frac{k}{m}}. \quad (20)$$

An example of bounded error step response obtained with VNODE-LP with a sampling parameter $\delta = 0.1$ s, $c = 1$, $m = 2$ and $k = [3, 9; 4, 1]$ is shown in Fig. 9a. There, $\zeta \simeq 0.177$ and the step response is a damped sinusoid. Because k is assumed to have an uncertain value bounded by an interval, the outputs are in the form of envelopes.

6.2 Unit cycle

In the case study, a unit cycle is defined by the application of a power unit for a determined time. The force is applied at time $t_0 + 5$ s, where t_0 is the cycle starting time. The force lasts 20s and cancels at $t_0 + 25$ s as shown by the red curve of Fig. 9b. The cycle ends at $t_0 + 50$ s.

Fig. 9b presents the system's response for a spring constant $k = [3.9, 4.1]$ N/m, a mass $m = 2$ kg, a damping coefficient $c = 10$ Ns/m, and initial speed and position equal to zero. The response is a decreasing exponential.

6.3 Degradation model

The degradation model chosen is the ageing of the damper cylinder. It is represented by a reduction of the damping coefficient proportional to the velocity of the mass [15]:

$$\dot{c} = \beta \dot{x}, \quad \beta < 0. \quad (21)$$

The more the spring is used, the weaker it becomes, characterized by the change in the damping coefficient.

6.4 Diagnosis

The FRP parameter estimation method presented in Section 4 has been used with the measures shown in Fig. 9c. These measures were obtained for

$$\theta = \begin{bmatrix} c \\ k \\ m \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix} \quad (22)$$

The goal is to estimate the damping coefficient c and the stiffness constant k . The search space is defined by the interval [4 9] for c and [3.5, 9] for k . The value of m is assumed to be known $m = 2$. Using the notation introduced above, we have:

$$\theta_{bol} = \begin{bmatrix} 4 \\ 3.5 \\ 2 \end{bmatrix}, \theta_{eol} = \begin{bmatrix} 9 \\ 9 \\ 2 \end{bmatrix} \quad (23)$$

The partition P_1 is achieved with a precision $\epsilon(P_1) = 1/10$ for the two parameters to be estimated c and k . Fig. 10 presents two examples of prediction results with two parameter boxes of P_1 : $[\theta]_i = [[4, 1, 4, 2], [4, 7, 4, 8], 2]^T$ on the

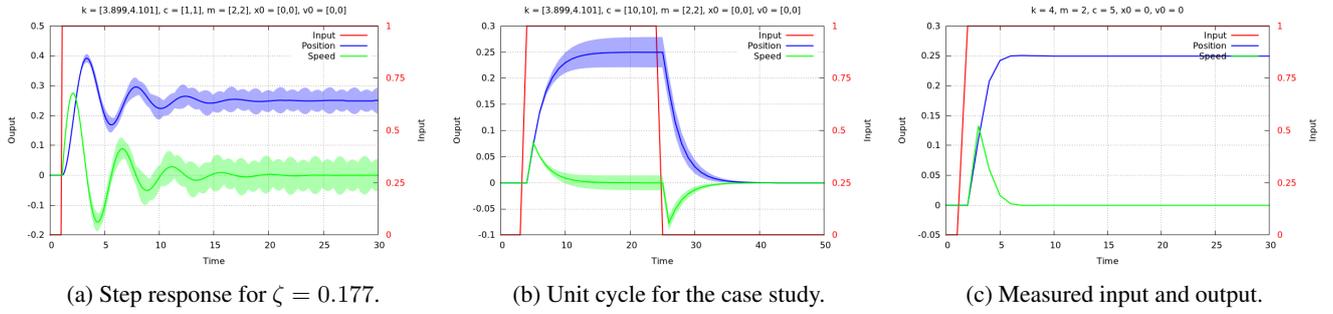


Figure 9 – Cases study simulation and data plots.

left and $[\theta]_j = [[5, 5, 1], [4, 4, 1], 2]^T$ on the right. On the left figure, one can see that there is no intersection between the estimate and the measurement for the position, hence the box used for the simulation is rejected. On the right, there is an intersection between the measurement and the estimation for all time points, but the estimate is not included in the measure envelop, hence the parameter box is considered undetermined.

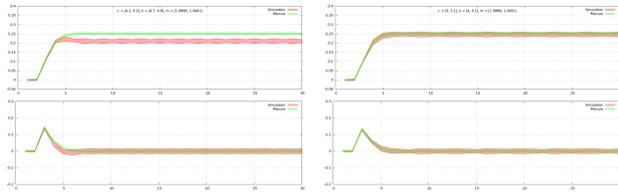


Figure 10 – Estimation results with a rejected parameter box (left) and an indetermined box (right)

The results for partition P_1 are presented in Fig. 11a and we obtain a first estimation for θ :

$$\hat{\theta} = \begin{bmatrix} [4, 1, 5.8] \\ [3, 7, 4, 2] \\ 2 \end{bmatrix}.$$

The estimation precision for partition P_1 is given by:

$$\omega(P_1) = \left| \text{mid}(\hat{\theta}) \right| ./ \left(\left| \text{mid}(\hat{\theta}) \right| + w(\hat{\theta})/2 \right) = \begin{bmatrix} 0.85 \\ 0.94 \\ 1 \end{bmatrix} \quad (24)$$

The first estimation for θ is used as the search space for partition P_2 , whose precision is increased by a factor of 10, i.e. $\epsilon(P_2) = 0, 1$. The obtained estimation results are shown in Fig. 11b.

The estimation is refined as:

$$\hat{\theta} = \begin{bmatrix} [4, 51, 5, 57] \\ [3, 85, 4, 14] \\ 2 \end{bmatrix}.$$

The precision is now $\omega(P_2) = [0.9, 0.96, 1]^T$, and the precision gain is $G(P_2/P_1) = [0.056, 0.025, 0]^T$. The values for the gain indicate that partitioning a third time might be quite inefficient. To confirm this fact, let us perform a third partition P_3 , whose precision is increased by a factor of 5, i.e. $\epsilon = 0.02$ (cf. Fig. 11c). The new estimation for θ is $\hat{\theta} = [[4.548, 5.526], [3.872, 4.132], 2]^T$, and the precision gain is $G(P_3/P_2) = [0.0073, 0.0036, 0]^T$. As expected, the gain is quite negligible with respect to the computation time increase.

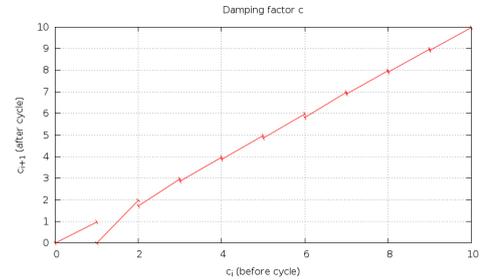
6.5 RUL computation

In this section we apply the set-membership method described in Section 5.2 to compute the RUL for the damping coefficient c .

The damper is assumed to fail when $c \leq c_{eol} = 2$. The degradation model (21) with $\beta = -0, 1^3$ allows us to determine the degradation table \mathcal{D}_c for the parameter c for a unit cycle:

$$\mathcal{D}_c = \begin{array}{c} \begin{array}{|c|c|} \hline c^i & \mathcal{D}(c^i) = c^{i+1} \\ \hline [9, 10] & [8.917, 9.977] \\ [8, 9] & [7.911, 8.978] \\ [7, 8] & [6.898, 7.979] \\ [6, 7] & [5.814, 6.982] \\ [5, 6] & [4.859, 5.979] \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline c^i & \mathcal{D}(c^i) = c^{i+1} \\ \hline [4, 5] & [3.874, 4.977] \\ [3, 4] & [2.863, 3.973] \\ [2, 3] & [1.721, 2.97] \\ [1, 2] & [0, 1.98] \\ [0, 1] & [0, 0.9755] \\ \hline \end{array} \end{array} \quad (25)$$

After proceeding to the linear interpolation given by (14), the graphical representation of c^{i+1} as a function of c^i is given by Fig. 12.


 Figure 12 – Approximated degradation of the damping coefficient c

The number of elements of the partition has been chosen relatively small to better illustrate the method. In a real situation, this number should be high in order to obtain less conservative predictions.

The value of c has been previously estimated and is

$$\hat{c} = [4.548, 5.526].$$

The graph of Fig. 12 allows us to approximate the predicted value after one unit cycle:

$$\mathcal{D}(\hat{c}) = c^1 = [4.4787, 5.4481].$$

The next iteration of the algorithm allows us to compute c^2 , etc. After 30 iterations, we obtain $c^{30} = [1.7665, 3.4235]$.

³The coefficient β has been chosen arbitrarily to illustrate the approach; it does not represent the real ageing of a damper.

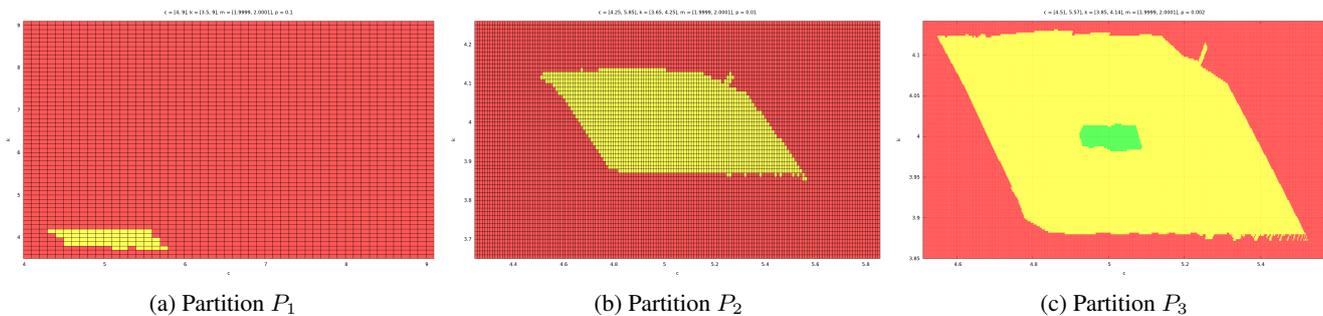


Figure 11 – Partitions and estimation results (red, yellow and green boxes are resp. rejected, undetermined, accepted parameters values).

Since $\bar{c}^{30} < c_{eol}$, we get $\overline{RUL} = 30$ cycles. After the 44th iteration, we get $c^{44} = [0.037591, 1.985928]$. We then have $\bar{c}^{44} < c_{eol}$ and hence $\overline{RUL} = 44$ cycles. The RUL of the damper is hence given by:

$$RUL = [30, 44] \text{ cycles.}$$

7 Conclusion

This paper addresses the condition-based monitoring and prognostic problems with a new focus that trades the traditional statistical approach by an error-bounded approach. It proposes a two stages method whose principle is to first determine the health status of the system and then use this result to compute the RUL of the system. This study uses advanced interval analysis tools to obtain *guaranteed* results in the form of interval bounds for the RUL.

The results for the case study demonstrate the feasibility of the approach. The next step is to adapt the FRP-based SM parameter estimation algorithm in order to output a list of boxes instead of a single box given by the convex hull of the boxes. The convex hull is indeed a very conservative approximation when the solution set is not convex.

The second stream of work is to consider contextual conditions and their associated uncertainties. Environmental conditions, like weather, different usage, etc. may indeed significantly affect the stress input and prognostics results.

References

- [1] Indranil Roychoudhury and Matthew Daigle. An integrated model-based diagnostic and prognostic framework. In *Proceedings of the 22nd International Workshop on Principle of Diagnosis (DX'11)*. Murnau, Germany, 2011.
- [2] Matthew J Daigle and Kai Goebel. A model-based prognostics approach applied to pneumatic valves. *International Journal of Prognostics and Health Management*, 2:84, 2011.
- [3] J. Luo, K. R Pattipati, L. Qiao, and S. Chigusa. Model-based prognostic techniques applied to a suspension system. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(5):1156–1168, 2008.
- [4] Q. Gaudel, E. Chanthery, and P. Ribot. Hybrid particle petri nets for systems health monitoring under uncertainty. *International Journal of Prognostics and Health Management*, 6(022), 2015.
- [5] D. Gucik-Derigny, R. Outbib, and M. Ouladsine. Estimation of damage behaviour for model-based prognostic. In *Fault Detection, Supervision and Safety of Technical Processes*, pages 1444–1449, 2009.
- [6] X. Guan, Y. Liu, R. Jha, A. Saxena, J. Celaya, and K. Geobel. Comparison of two probabilistic fatigue damage assessment approaches using prognostic performance metrics. *International Journal of Prognostics and Health Management*, 1(005), 2011.
- [7] R.E. Moore. *Automatic error analysis in digital computation*. Technical report LMSD-48421, Lockheed Missiles and Space Co, Palo Alto, CA, 1959.
- [8] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.
- [9] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with examples in parameter and state estimation, Robust control and robotics*. Springer, Londres, 2001.
- [10] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29:1053–1064, 1993.
- [11] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with examples in parameter and state estimation, Robust control and robotics*. Springer, Londres, 2001.
- [12] C. Jauberthie, N. Verdière, and L. Travé-Massuyès. Fault detection and identification relying on set-membership identifiability. *Annual Reviews in Control*, 37:129–136, 2013.
- [13] N. Nedialkov. VNODE-LP, a validated solver for initial value problems for ordinary differential equations.
- [14] R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. W. Kaucher, U. W. Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286. Wiley-Teubner, Stuttgart, 1987.
- [15] Ian M Hutchings. *Tribology: friction and wear of engineering materials*. Butterworth-Heinemann Ltd, 1992.

Configuration as Diagnosis: Generating Configurations with Conflict-Directed A* - An Application to Training Plan Generation -

Florian Grigoleit, Peter Struss
Technische Universität München, Germany
email: {struss, grigolei}@in.tum.de

Abstract

Although many approaches to knowledge-based configuration have been developed, the generation of optimal configurations is still an open issue. This paper describes work that addresses this problem in a general way by exploiting an analogy between configuration and diagnosis. Based on a problem representation consisting of a set of ranked goals and a catalog of components, which can contribute in combination to their satisfaction, configuration is formulated as a finite constraint-satisfaction-problem. Configuration is then solved by state-search, in which a problem solver selects components to be included in an appropriate configuration. A variant of Conflict-Directed A* has been implemented to generate optimal configurations. To demonstrate its feasibility, the concept was applied, among other domains, to personalized automatic training plan generation for fitness studios.

1 Introduction

Besides diagnosis, the task of configuration has been one of the earliest application areas of work on knowledge-based systems, initially in the form of rule-based “expert systems”, for instance in [1]. Today, systems for automated configuration have reached maturity for practical applications, as shown in [2], [3], and [4]. Despite this success, developing algorithms for computing **optimal** or optimized configurations with general applicability still deserves more research efforts.

Driven by a number of different configuration tasks, we developed GECKO (Generic constraint-based Konfiguration), a generic solution to the configuration problem that can be specialized to different application domains and that, among other objectives, aims at supporting the generation of optimal configurations.

In a nutshell, the solution exploits an analogy:

- The configuration task can be seen as searching for an assignment of *active* or *non-active* to the components in a given repository, representing whether or not a component is included in the configuration, such that it achieves some goals in an optimal way
- Diagnosis has been formalized as a search for an assignment of behavior modes (*normal* or *fault_x*)

to a set of system components such that it is optimally compliant with a set of observations.

Based on this analogy, we exploit a search technique that has been developed as consistency-based diagnosis, see [5]), and as a generalization for optimal constraint satisfaction, called conflict-directed A*, see [6]

In the following section, we discuss related work on configuration systems. In section 3, we present some examples of configuration problems that we tackled using GECKO and that will serve for illustration purposes. Next, we introduce our formalization of the configuration task and the key concepts of GECKO. In section 5, we discuss the analogy between diagnosis and configuration, the application of CDA*, variants of utility functions and how they relate to different types of configuration applications. The results are shown in section 6. Finally, our current work and some of the open issues are discussed.

2 Knowledge-based Configuration

Applications of configuration are immensely diverse, but they all share a number of common problems, such as compliance with domain knowledge, size of the solution space, and the resulting complexity of the problem solving task. It requires knowledge-based approaches to support the problem-solving activities, such as product configuration or variability management see [3] and [4].

Current research on configuration, especially for large applications, tends to neglect global optimization, focusing on local optimization, user interaction, or aiming at producing “good” solutions, see [3] and [7].

The focus of this paper is a generic, constraint-based configurator (GECKO) for solving optimal configuration problems. The core of GECKO is a variant of Brian Williams’ Conflict-Directed A* (CDA*, [6]). The solution works on a generic representation of configuration knowledge and tasks. We consider the task of generating configurations as similar to consistency-based diagnosis. Instead of assigning modes for fault identification as in [5], GECKO assigns the activity to components contributing to goals. A configuration is consistent if all task-relevant goals are satisfied. The quality of a configuration is given by the level of goal satisfaction and the amount of resource consumption. Our approach allows the arbitrary selection of optimization criteria, like minimal resource consumption or maximal goal contribution. In the presented case study, our aim was to maximize the number of satisfied goals under consideration of available resources.

3 Application Examples

Configuration problems are almost ubiquitous in modern life, with applications as different as creating a customized computer as done by R1 in [1] and adapting the system functions of a car, see [8]. To illustrate the versatility of GECKO, we present three applications.

3.1 Car Configuration

Today, car manufacturers offer a vast number of models, model variants, and equipment options to their customer. The resulting complexity does not only prohibit a comprehensive exploration of the solution space, but is also likely to provide customers with sub-optimal car variants. A domain model for car configuration was created and mapped to the GECKO concepts, which are presented in 4.2([9])

3.2 User Interface Configuration

The Beam Instrumentation group at CERN is responsible for the design and implementation of particle beam measurement systems. These systems are specifically built for each case, resulting in extensive work on constructing them. While the generation of the GUIs, that is the implementation, is automated, the configuration is not. This task currently requires an expert to select libraries, graphical elements, and data sources and to parameterize them. Such tasks are typical configuration tasks and thus enable the automation of the configuration of the GUIs by GECKO ([10]).

3.3 Training Planning in Sport Science

At a first glance, training planning may appear to be a typical scheduling task, instead of a configuration problem. Taking a closer look shows that it mainly involves activities we consider the core of configuration: selecting, parameterizing, and arranging components to satisfy goals, whereas assigning time slots to the selected exercises is, in general, fairly straightforward

A trainer has to analyze the biometric state of his trainee, such as fitness or age, to consider constraints on the created training plan, for example duration or available equipment, and to select and order appropriate exercises.

The sheer number of existing exercises and the size of the solution space show that training planning includes optimization. In general, a trainer tries to maximize the training effect within the available time and under consideration of the trainee's goal and abilities. The specialization of GECKO to training planning is described in section 5.

4 GECKO - Foundations

4.1 Intuition

With GECKO, we aim at developing a generic solution to configuration problems, which can be tailored towards a particular domain by specializing some basic classes and creating a knowledge base in terms of domain-specific constraints. Its design is driven by the following objectives:

- supporting both automatic and interactive configuration;

- enabling the use of the system without deep domain knowledge, esp. about how high-level goals of the user break down to more detailed and technical ones;
- handling also soft domain constraints and user preferences, and
- offering support to the user by providing explanations for generated parts of the configuration and for unavailable options and by suggesting revisions to resolve inconsistencies.

However, this paper focuses on the basis, a generic problem solver for (optimal) configuration. Determining the solution – the configuration - means selecting a set of instances of given types of elements - components -, perhaps with certain attribute values and organized in a particular structure. The configuration has to

1. satisfy a set of high-level user goals,
2. be compliant with particular attributes and restrictions supplied by the user,
3. be realizable both in principle (i.e. not violating domain-specific restrictions on valid configurations),
4. under consideration of available resources, and
5. optimal (or near optimal) according a criterion that reflects the degree of fulfilling the goals and the amount of resources consumed.

Configurations can be physical devices, such as turbines, communication systems, and computers, abstract ones like a curriculum or a company structure, or a software system. In contrast to a design task involving the creation of new types of components, configuration assumes that all required Components are instances of component types from a repository ([11]). This leads to different kinds of reasoning involved: innovative design has to verify that its result satisfies the goals by inferring that they achieved by the system behavior based on behavior models of the components, whereas for a configuration task, it is assumed that behavioral implications of aggregated components have been compiled into explicit interdependencies of Goals and Components. As a result, software systems for configuration are typically based on knowledge encoded as constraints or rules, as in [1] and [2], and do not require the exploitation of behavior models.

4.2 Core Concepts

The core concepts of GECKO are derived from the description above, as depicted in Fig. 1:

- **Goals** express the achievements expected from a specific configuration. They may have an associated priority dependent on the task and different criteria for goal satisfaction.
- **Components** are the building blocks of the Configuration. They may be organized in a type hierarchy (for example, Lithium battery is a voltage source). In addition, there may be Components that are aggregations of lower level components.
- A **Task** specifies the requirements on a configuration from the user's perspective. It is split into three kinds of restrictions:

$$\text{Task} = \text{TaskGoals} \cup \text{TaskParameters} \cup \text{TaskRestrictions.}$$

TaskGoals are a collection of Goals the user is aware of and which can be (de)activated or prioritized by the user. Each TaskGoal_n is stated as a restriction $\text{TaskGoal}_n.\text{Satisfied}=\text{T}$ in the Task description.

While **TaskGoals** represent objectives a user requires, **TaskParameters** associate values to properties of the Task, hence have the form $\text{TaskParameter}_k=\text{value}_{kj}$. For instance, in vehicle configuration, the target country may have an influence on daytime running lights being mandatory. However, these implications are not drawn by the user (who only provides the country information), but by the domain knowledge represented in the system.

In contrast, **TaskRestrictions** refer explicitly to the choice of Components and their attributes, e.g. that for the user, a convertible is not an option or that the engine should be a Diesel engine.

A specific, and often essential, TaskRestriction can be included:

- A **ResourceConstraint** limits the cost of the configuration, which may be indeed money (car configuration) or time (in training plan configuration), but also computer memory etc. Components have to have an attribute that allows calculating the resources needed for the entire configuration (often as the sum).

4.3 Constraints on Configurations

The configuration knowledge of a particular application comprises the domain-specific specialization and instantiation of Goals, Components (possibly including component attributes and their domains), and relevant TaskParameters and their domains as well as constraints that capture interdependencies among these instances. Dependent on which kinds of objects are related, we distinguish between the following (illustrated in Fig. 1):

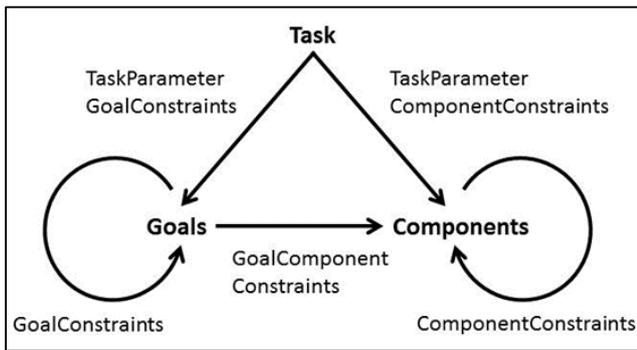


Fig. 1 Task constraints in GECKO

- **TaskParameterGoalConstraints** express that certain TaskParameter values may exclude or require certain goals
- **GoalConstraints** relate goals to each other, in particular for refinement of higher-level (esp. TaskGoals) to lower-level ones, such as goals related to various muscle groups that should be exercised, although the user is not aware of this
- **GoalComponentConstraints** capture essential configuration knowledge, namely whether and how the available components contribute to the achievements of goals

- **ComponentConstraints** establish interdependencies among components (and their attributes): a component may be dependent on or incompatible with the presence of another component in the configuration
- **TaskParameterComponentConstraints** may include or exclude certain components based on TaskParameter values

A fundamental constraint type is

Requires (x, y)

which is defined by

$$x.\text{active}=\text{T} \Rightarrow y.\text{active}=\text{T}$$

and used to express dependencies among goals (e.g. refining a goal to a set of mandatory sub-goals) and components (e.g. cruise control requires automatic transmission) and as the fundamental coupling between goals and components (to achieve high-speed driving, an engine of a certain power is needed). Furthermore, in order to express that several goals or components provide some partial contributions that jointly result in the satisfaction of a goal (or establish the preconditions of a component), we introduce the concept of a **choice**, which can also fill the role of y in a Requires-constraint. A choice is given by a relation

$$\text{GoalChoice} \subset \text{Goals} \times \text{ContributionDom}$$

or

$\text{ComponentChoice} \subset \text{Components} \times \text{ContributionDom}$, where ContributionDom specifies a set of values for quantifying how much a goal or component contributes to the satisfaction of the choice and needs a zero element and an operator Σ to add up contributions (e.g. addition of integers). The idea behind choices is implemented by three kinds of constraints. The degree of the satisfaction of a (component) choice is given by the combined contributions of the active components of the choice:

$$\text{Choice}.\text{satLevel} = \Sigma \text{Choice}.\text{goal}.\text{actContribution}$$

and

$$\text{Choice}.\text{goal}.\text{actContribution} = \begin{cases} \text{Choice}.\text{goal}.\text{contribution} & \text{IF goal.active}=\text{T} \\ \text{zero} & \text{IF goal.active}=\text{F} \end{cases}$$

The choice is satisfied, if the satLevel lies in a specified range, satThreshold:

$$\text{Choice}.\text{active} = \text{T} \Leftrightarrow$$

$$\text{Choice}.\text{satLevel} \in \text{Choice}.\text{satThreshold}$$

This allows implementing not only a minimum level as a precondition for the satisfaction of a choice, but also a maximum. Preventing “over-satisfaction” may not be a common requirement, but in the fitness domain, one may want to restrict the set of exercises that impose a load on a particular muscle group.

Another predefined general type of constraint is

Excludes (x, y)

defined by

$$x.\text{active}=\text{T} \Rightarrow y.\text{active}=\text{F}$$

to express conflicting goals, incompatible components, and TaskParameterGoal/ComponentConstraints (e.g. high body weight may rule out certain exercises).

The application-specific configuration knowledge is, thus, basically encoded as a set of the constraints explained above. This, together with the domain-specific ontology (as a specialization of the basic GECKO concepts, including choices, and associated attributes) and, perhaps, specific contribution domains and operators, establishes the con-

figuration knowledge base, called **ConfigKB** in the following.

We make some reasonable fundamental assumptions about ConfigKB:

- Each potential **TaskGoal** is **supported**: it is the starting node of a connected hyper graph of Requires constraints that includes components, i.e. it actually needs a (partial) configuration in order to be satisfied (which does not mean it can actually be satisfied).
- **Closure assumption**: the encoded interdependencies, esp. the Requires constraints, are complete. In other words, if all constraints Requires (x, y) associated with x are satisfied by a configuration, then x is satisfied.
- It is **consistent**.

4.4 Definition of the Configuration Task

The goal is to select an appropriate subset of the available components, which we call the active ones, and possibly determine or restrict their attributes.

Definition 1 (Complete Configuration)

A **configuration**

$PARCONFIG = (ACTCOMPS, COMPATTR)$

is **complete** if includes exactly the active components:

$comp \in ACTCOMPS \Leftrightarrow comp.Active = T.$

GECKO has to generate a configuration PARCONFIG that satisfies the criteria stated in section 4.1.

Definition 2 (Solution to a Configuration Task)

A **configuration task** is a pair

$(ConfigKB, Task)$

(as specified in sections 4.3 and 4.2, respectively), and a complete configuration PARCONFIG is a **solution** to it, if it is consistent with the ConfigKB and the Task,

$PARCONFIG \cup ConfigKB \cup Task \neq \perp.$

This may seem too weak, because criterion 1 in section 4.1 requires the entailment of the satisfaction of the TaskGoals in Task.

Proposition 1

If PARCONFIG is a solution to a configuration task $(ConfigKB, Task)$, then

$PARCONFIG \cup ConfigKB \models \forall goal \in TaskGoals \ goal.Satisfied = T.$

This follows from the closure assumption: Since for the chosen TaskGoals, Satisfied=T is explicitly introduced in Task, it follows that all Requires constraints related to them are satisfied, and, hence, they are not only consistent, but entailed. As for the other criteria of section 4.1:

2. Compliance with specific application requirements is guaranteed by consistency with the TaskParameters under the TaskParameterGoal/ComponentConstraints in ConfigKB and with TaskRestrictions in Task
3. Realizability is established by consistency with ComponentConstraints
4. The ResourceConstraint is also consistent.

Criteria 5, optimality, will be discussed in the following section.

5 Generating (Near) Optimal Configurations

5.1 Configuration as Diagnosis

The current version of GECKO is based on the assumption that there exists a **finite set of components, COMPS**, as a repository for all configurations. This means, no new instances of components types are created during configuration and, more specifically, a component will not be duplicated if it is included in the configuration due to several constraints. In this case, determining ACTCOMPS of a complete configuration can be seen as an activity assignment

$AA: COMPS \rightarrow \{active, inactive\},$

indicating the inclusion in or exclusion from the configuration, and the consistency test of Definition 2 becomes

$AA \cup ConfigKB \cup Task \neq \perp.$

This representation shows the analogy to the consistency-based formalization of component-oriented diagnosis: an assignment MA of modes (i.e. nominal or faulty behavior) to a set of components,

$MA \rightarrow \{OK, fault1, fault2, \dots\}$

characterizes a diagnosis, if it is consistent with the domain knowledge (a library of behavior models and a structural description), called system description, SD, and a set of observations, OBS:

$MA \cup SD \cup OBS \neq \perp.$

In both cases, the **assignments to the components**

$AA \leftrightarrow MA$

are checked for consistency with a fixed set of constraints representing the **domain knowledge**

$ConfigKB \leftrightarrow SD,$

and a set of constraints representing a **specific problem instance**

$Task \leftrightarrow OBS.$

In consistency-based diagnosis, theories and algorithms have been developed to determine diagnostic solutions, which can be exploited for the configuration task based on the analogy outlined above.

5.2 Conflict-directed A*

Based on the above formalization, many implementations of consistency-based diagnosis exploit a best-first search for consistent mode assignments, using probabilities of individual behavior modes as a utility function (and usually making the assumption that faults occur independently) as SHERLOCK does([12]). Classical A* search has been extended and improved by pruning the search space based on inconsistent partial mode assignments that have been previously detected during the search (called **conflicts**), exploiting a truth-maintenance system (TMS, such as the assumption-based TMS [13]) as a dependency recording mechanism that delivers conflicts. From the diagnostic solutions, this approach has been generalized later as conflict-directed A* search, see [6].

Procedure CDASTAR

- 1) Terminate=F
- 2) Solutions= \emptyset
- 3) Conflicts= \emptyset
- 4) VA=VA_{initial}
- 5) DO WHILE Terminate=F
- 6) Apply Constraints(VA)
- 7) Check consistency of VA
- 8) IF consistent
- 9) THEN append VA to Solutions
- 10) Terminate=Solutions.Terminate
- 11) ELSE
- 12) Conflicts=APPEND(Conflicts, newConflicts)
- 13) END IF
- 14) VA=Conflicts.BestCandidateResolvingConflicts
- 15) END DO WHILE
- 16) RETURN Solutions

The effectiveness of the pruning of the search space based on previously detected inconsistencies (highlighted in the above pseudo code) grows with the number of (non-redundant) conflicts that are extracted. Achieving this, however, can be computationally expensive and may have to be traded off against the computational cost of the consistency test and/or the optimality of the solution. We will get back to this issue below.

The straightforward mapping of the configuration problem to CDA* is obtained by representing configurations as variable assignments:

$$\text{VARS}=\{ \text{Comp}_i.\text{active} \mid \text{Comp}_i \in \text{COMPS} \}$$

$$\text{DOM}(\text{Comp}_i.\text{active})=\{ T, F \} .$$

To illustrate how the algorithm works using a simple example, assume that goal G_1 depends on a component choice that involves 3 components, C_i , each with a contribution of 1 in this choice, which has a satisfactionThreshold (2,3), i.e. it is satisfied if at least two of the components are active. Search starts with an empty configuration (active=F for all components) which leads to an inconsistency with the constraints related to the choice. Each pair of inactive components establishes a (minimal) conflict:

$$\{ C_1.\text{active}=F, C_2.\text{active}=F \},$$

$$\{ C_1.\text{active}=F, C_3.\text{active}=F \},$$

$$\{ C_2.\text{active}=F, C_3.\text{active}=F \}.$$

Configurations resolving these conflicts are the ones with active components

$$\{ C_1, C_2 \}, \{ C_1, C_3 \}, \text{ or } \{ C_2, C_3 \},$$

and the best one would be checked further. If this is done against another choice for a goal G_2 , which is based on components C_3, C_4, C_5 (again all with contribution 1) and a threshold (1, 3), then a new conflict

$$\{ C_3.\text{active}=F, C_4.\text{active}=F, C_5.\text{active}=F \}$$

is detected, and the configurations resolving all include active components are

$$\{ C_1, C_3 \}, \{ C_2, C_3 \}, \{ C_1, C_2, C_4 \}, \text{ or } \{ C_1, C_2, C_5 \}.$$

5.3 Diagnosis vs. Configuration

Despite the mentioned basic commonality, there are some important distinctions at a conceptual level, but with a potentially strong impact on the computational complexity.

Partial vs. complete assignments

In diagnosis, it is possible to check partial mode assignments to detect useful conflicts. In configuration, we have to consider **complete** variable assignments, which, in assigning T or F to activity variables of **all** components, correspond to complete configurations. The reason is that, as illustrated by the above trivial example, the constraints related to a choice deliver important **conflicts** based on components being **not active**. A partial configuration, e.g. assigning active=T to, say, C_1 only, is consistent with the respective choice; that this configuration does not satisfy G_1 is detected only, if all other components are assumed to be inactive (Of course, if the satThreshold has an upper limit, we obtain conflicts involving too large sets of active components, as well). This observation is related to another difference:

(NON-)Locality of the Domain Theory

In diagnosis, the domain theory is as modular as the device: it consists of constraints that represent the local interaction of components and constraints that capture the local behavior of components under certain modes. Checking the consistency of a partial mode assignment requires applying the directly related constraints only. In contrast, constraints representing configuration knowledge are almost by definition non-local: they are meant to relate many components across the entire configuration, e.g. as choices. If choices play a major role and are large, this can be a source of severe problems.

The training plan generation application forms an extreme example: choices may involve in the order of 100 components, because many exercises may be related to a particular muscle group, while only a handful of them together satisfy the goal. In addition, exercises are challenging several muscle groups. If the lower boundary of the satisfactionThreshold of a choice is k and the size of the choice is n , then (assuming a contribution 1 for each component), the number of resulting minimal conflicts will be

$$\binom{n}{k-1}$$

– prohibitively large in the training application. This has an impact on the algorithm, as discussed in section 5.5.

First, we have to introduce appropriate utility functions to measure the quality of a configuration.

5.4 Utility Functions

The utility of a configuration should essentially reflect

- the degree of **fulfilment** of the relevant **goals** and
- the amount of resources required.

A measure of the former may also consider **priorities** of goals. The same holds for individual components. Since inactive components neither make contributions nor consume resources, it is plausible to assume that the **utility** of a configuration depends on its **active components only**.

In the following, it is assumed that

- the contribution of a configuration is obtained solely as a **combination of contributions** of the active components included in the configuration and otherwise independent of the type of properties of the components,
- we can define a subtraction “-” of contributions,
- the cost of the contribution is given as the sum of the cost of the involved active components and will usually be numerical,

- we can define a ratio “/” of contributions and resources,
- there is a function that maps priority of goals to a weight of the contributions and a kind of multiplication,

$$*: \text{DOM}(\text{weight}) \times \text{DOM}(\text{contribution}) \rightarrow \text{DOM}(\text{contribution}),$$
 is defined.

Then the following specifies a family of utility functions (where we simplify the notation by writing $\text{Goal}_j.\text{SatThreshold}$ instead of $\text{Choice}_j.\text{SatThreshold}$ etc.):

For an active Goal Goal_j , the TotalContribution of a Configuration is

$$\text{Configuration.TotalContribution}(\text{Goal}_j) :=$$

$$\sum_{\text{Comp}_i \in \text{Configuration.ACTCOMPS}} \text{Comp}_i.\text{Contribution}(\text{Goal}_j)$$

where Σ denotes the Combine operation, the ActualContribution is given as

$$\begin{aligned} \text{Configuration.ActualContribution}(\text{Goal}_j) := \\ \max(\text{Configuration.TotalContribution}(\text{Goal}_j), \\ \text{Goal}_j.\text{Combine}(\text{Goal}_j, \\ \text{SatThreshold}, \text{posTolerance})), \end{aligned}$$

and a penalty (for over-satisfaction) as

$$\begin{aligned} \text{Configuration.Penalty}(\text{Goal}_j) := \\ \max(0, \\ \text{Configuration.TotalContribution}(\text{Goal}_j) \\ - \text{Goal}_j.\text{Combine}(\text{Goal}_j, \text{SatThreshold}, \text{posTolerance})). \end{aligned}$$

Based on this, we define the utility function as

$$\begin{aligned} \text{Configuration.Utility}(\text{ACTGOALS}) := \\ \sum_{\text{Goal}_j \in \text{Configuration.ACTGOALS}} \\ \text{weight}(\text{Goal}_j.\text{Priority}) \\ * \text{Configuration.ActualUtility}(\text{Goal}_j) \\ + f * \text{Configuration.Penalty}(\text{Goal}_j) \\ / \sum_{\text{Comp}_i \in \text{Configuration.ACTCOMPS}} \text{Comp}_i.\text{Resource}. \end{aligned}$$

The factor f determines whether or not excessive contributions are penalized (by the excessive amount); the weight can emphasize contributions to Goals with high priority, and the tolerance interval can express how exactly the intended SatThreshold has to be hit.

5.5 GECKO Algorithm

For the GECKO variant of CDA* we modified CDA* by activating only the constraints needed at a specific stage, thereby reducing the number of occurring conflicts significantly.

GECKO characterizes a stage in the problem solving process and hence the criteria for constraint activation as a pair

$$S = (\text{GOALS}, \text{configuration}),$$

that is a set of goals that are considered and a configuration to be checked for consistency. This allows for search strategies that do not consider all active goals from the beginning. Therefore, the constraints to be applied are not only determined by the variable assignment, but also by the goals. In our first application, goals are activated in a descending order, according to their priority.

To determine the hitting sets of the conflicts we use different algorithms from [14], depending on the domain. In `BestCandidateResolvingConflicts`, the next-best solution is generated.

Procedure GECKO Configuration Algorithm

- 1) `ApplyConstraints(Constraints(Initial))`
- 2) `ActComps=ACTCOMPS0`

- 3) `Priority=max(actGoals.Priority)`
- 4) `DO WHILE Priority >=1`
- 5) `ApplyConstraints(Constraints(GoalPriorityClass(Priority))`
- 6) `VA=VA(ActComps,COMPS\ActComps)`
- 7) `NewActComps=`
`GECKO.CDASTAR(VA).ActComps`
- 8) `ActComps = NewActComps.Commit`
- 9) `END DO WHILE`
- 10) `RETURN ActComps`

In line 8, the algorithm fixes the components added to satisfy the recently considered goals. This means, when trying to satisfy further goals (with lower priority) they will not be de-activated. This heuristic aims at satisfying as many goals as possible with the given resources in the order of their priority, but, obviously, may miss a globally optimal solution.

6 Case Study: Training Plan Generation

We are working on the realization of the three applications presented in section 3. To demonstrate the specialization of GECKO concepts and the capabilities of the GECKO algorithm, we selected the fitness training example. From the three examples, fitness is best suited to illustrate the advantages of CDA* in configuration.

6.1 Domain Theory

In fitness, trainees perform exercises, like push-ups or running, to train body parts under certain aspects (endurance, muscle gain). To train means to improve physical abilities, like endurance, and to influence biometric parameters, such as weight. In configuration terms: exercises contribute to a set of fitness goals. Hence, we created the domain theory for training planning using the concepts specified in section 4.2. Table 1 contains an overview on the most important specializations.

The result may appear straightforward to outsiders, but it is actually the result of several months of analyses carried out jointly with experts from sports sciences, which took as to several versions and revisions of the model.

Table 1: Specialization of GECKO Concepts

GECKO Concept	Fitness Concept	Example
Goal	TraineeGoal	Muscle Gain
	TrainingGoal	Strength
	TargetGoal	Biceps
Component	Exercise	Push-up
Task	Trainee	-
TaskRestriction		TrainingDuration
TaskParameter	TrainingProperty	Equipment
	TraineeProperty	Fitnesslevel

Task

A GECKO Task in fitness is a trainee, or more precisely the request of a training plan by a trainee. A trainee has expectations regarding the result of the training, represented by `TraineeGoals`. The Trainee also has a set of `TraineeProperties`, like `Fitnesslevel`, and sets the `TrainingProperties`. Furthermore, a trainee has to specify the desired `TrainingDuration`.

Special among the `TraineeProperties` are the `FitnessTargets` and `FitnessCategories`. A `FitnessTarget` has to be

trained by an Exercise, such as legs. FitnessCategories are the main abilities of a Trainee, such as strength.

Goals

The domain theory contains three types of goals:

- TraineeGoal: The only **TaskGoal** in fitness, describing the expected effect of the fitness training
- TrainingGoal: Abstract goals, specifying the type of physical ability to be improved e.g. strength
- TargetGoal: Body part the training has to stimulate.

To capture the structure of the human body and the differences in fitness categories, we decompose the TargetGoals into three levels:

- RegionGoal
- MuscleGroupGoal
- MuscleGoal

Reflecting the FitnessTargets, TargetGoals are structured in a goal tree. Because FitnessTargets are trained at different levels, the tree is unbalanced. For example Endurance is generally trained for the whole body, while Strength is trained at a muscular level.

Components

All components in fitness are exercises. Each exercise is related to a FitnessCategory, e.g. pushup is a StrengthExercise. Exercises can contribute to multiple TargetGoals, but only TargetGoals of their own FitnessCategory. For example, a StrengthExercise can only contribute to TargetGoals related to strength.

Exercises comprise a set of fixed attributes, such as requiredEquipment or requiredFitnesslevel, as well as a set of unspecified attributes, like TrainingWeight or Durations. The values of such volatile attributes depend on the selected TraineeGoal, because they define how an exercise effects a FitnessTarget – an increase in strength is achieved by a small number of slow repetitions with very high weight, while fat is burnt best with many fast repetitions with little weight.

Utility

The utility of a configuration in SmartFit depends on the contributions of the active components to required Choices $DOM(\text{comp}_i, \text{contribution}_i) = \{20, 40, 60, 80, 100\}$

The satThreshold of the Choices depends on the priority of the associated goal

$\text{satThreshold} = \text{combine}(\text{Goal}_j, \text{Priority}, \text{normThreshold})$,
with $DOM(\text{Priority}) = \{1, 2, 3, 4, 5\}$.

For the example in 6.2, we simply multiplied the priorities with the normThreshold = 80.

The domain of the combined contribution is from 0 to 500 in steps of 20. In case of contributions larger than 500, the overshoot is cut, and the value set to 500.

The utility for fitness training is given by the following equation:

$$\text{Config.Utility (ACTGOALS)} := \frac{\sum_{\text{Goal}_j \in \text{Config.ACTGOALS}} \text{weight}(\text{Goal}_j, \text{Priority}) * \text{Config.ActualUtility}(\text{Goal}_j)}{\sum_{\text{Comp}_i \in \text{Config.ACTCOMPS}} \text{Comp}_i.\text{Resource}}$$

6.2 Simplified Example

To make the capabilities of GECKO more tangible, we present a small experiment. For brevity and clarity, we use a reduced knowledge-base, with three MuscleGoals(table

2), 12 exercises(table 3), and 2 TaskParameters, namely Equipment and a general Fitnesslevel – thus omitting the consideration of different Fitnesslevels related to the specific FitnessTarget, as done in the application system. Furthermore, we set the duration of all exercises to require 5 minutes.

Using this reduced knowledge-base, we applied both the basic GECKO algorithms and the goal-focused variant. The results are described in the following subsection.

Table 2: Exemplary muscle goals with priorities

ID	MuscleGoal	Priority: MuscleGain	Priority: GeneralFitness
G1	Biceps	1	2
G2	Triceps	1	2
G3	Latissimus	2	3

Table 3: Exercises and parameters

ID	Exercise	Contributions	Required Equipment	Required Fitness level
C1	Biceps Curl	Biceps: 100	None	1
C2	Dips	Triceps: 100	None	1
		Latissimus: 20		
C3	Lat-Pull	Biceps: 20	Machines	1
		Latissimus: 100		
C4	Rev. Butterfly	Triceps: 40	Machines	1
C5	Pushup	Triceps: 80	None	2
C6	Pushup on knees	Triceps: 60	None	1
C7	Shoulder press	Triceps: 80	Machines	1
C8	Rowing	Biceps: 40	Machines	1
		Latissimus: 80		
C9	Pull up	Biceps: 100	None	2
		Latissimus: 80		
C10	Triceps Pulldown	Triceps 100	Machines	2
C11	Pull up (supported)	Biceps: 20	None	1
		Latissimus: 80		
C12	Rowing one-armed	Biceps: 40	Machines	2
		Latissimus: 100		

To compare the results of different tasks, we conducted to experiments with different TraineeGoals and TaskParameter values. For the basic algorithm, we used the Tasks shown in Table 4.

Table 4: Task for experiments A and B

Variable	Values A	Values B
TaskGoal	General Fitness	Muscle Gain
TaskParameter: FitnessLevel	Untrained (1)	Trained (2)
TaskParameter: Equipment	Machines	none
TaskRestriction: TrainingDuration	15 minutes	30 minutes

The results of the configuration with the basic GECKO algorithm are shown in Tables 5 and 6.

Table 5: Configuration results basic GECKO Algorithm

Experiment A	Experiment B
Lat-Pull	Pull up
	Dips
Rowing	Pull up (supported)
	Pushup on knees
Shoulder press	Biceps Curl

Table 6: State of the Goals after running the basic Algorithm

Muscle Goal	Value A	Value B
Biceps	Partially Satisfied	Satisfied
Triceps	Satisfied	Satisfied
Latissimus	Satisfied	Partially Satisfied

Application Evaluation

The results indicate that the GECKO algorithms are capable of generating optimal solutions to configuration problems. In experiment B, it can be seen that GECKO was not able to satisfy G3 completely, since there were not enough consistent exercises available. Thus, the less important goals were satisfied, but not the important one. In experiment A on the other hand, the algorithm was able to fully satisfy G3 but not G1, since the duration resource was only sufficient for three exercises.

6 Discussion and Outlook

The results shown above indicate that treating configuration as a diagnostic problem, and solving it with techniques from consistency-based diagnosis is a promising approach to user-oriented configurators for optimal configuration problems.

The analysis of different application domains, including the ones mentioned in section 3, triggers the insight that variations of the search algorithm may be required in order to reflect the specific requirements and structure of the problems. This is particularly true for applications that involve a high level of interaction, such as leaving choices to the user, providing explanations for system decisions, and allowing him to modify his/her decisions in an informed way. Retracting decisions and also generating explanations can be supported by the ATMS, which also produces conflicts.

The conceptual and algorithmic solution to configuration generation presented in this paper could certainly be implemented using other techniques that have been proposed and used for configuration. However, our choice of an ATMS-based solution (and CDA*) was strongly motivated by the overall objectives stated in section 4.1: we intend to base explanation facilities (“which user inputs and domain restriction prevent option x to be viable?”), preferences and soft constraints, and the possibility to retract input and explore several alternative solutions on capabilities of the ATMS.

A goal of our work is to extract features from the case studies that can support a classification of configuration applications as a basis for selection from a set of predefined algorithm variants and strategies for man-machine interaction.

Other options, such as compiling (parts of) the constraint network and moving search heuristics to a lower

technical level (the constraint system) will also be explored.

Furthermore, we are currently preparing an application to configuration of automation systems for collaborative, flexible manufacturing and modular multi-purpose vehicles. This application of GECKO is likely to require stronger spatial and also temporal constraints for structuring a configuration.

Acknowledgments

We would like to thank our project partners for providing their domain knowledge and their assistance, esp. Florian Kreuzpointner and Florian Eibl. Special thanks to Oskar Dressler (OCC’M Software) for proving the constraint system (CS3 or Raz’r). The project was funded by the the German Federal Ministry of Economics and Technology under the ZIM program (KF2080209DB3).

References

- [1] JP McDermott, “RI: an Expert in the Computer Systems Domain”, *Artificial Intelligence*, 1980.
- [2] U. Junker, D. Mailharro, “The logic of ILOG(J) Configurator: Combining Constraint Programming with a Description Logic”, *IJCAI*, 2003.
- [3] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, “Knowledge-based Configuration: From Research to Business Cases.”
- [4] D. Sabin, R. Weigel, “Product configuration frameworks – a survey.” *IEEE Intelligent System*, 1998.
- [5] J. de Kleer, BC Williams, “Diagnosing Multiple Faults,” *Artificial Intelligence*, 1987
- [6] BC William, R.J. Ragno, “Conflict-directed A* and its role in model-based embedded systems”, *Discrete Applied Mathematics*, 2007.
- [7] M. Stumptner, G. Friedrich, A. Haselböck, “Generative constraint-based configuration of large technical systems“, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 1998.
- [8] G. Weiß, F. Grigoleit, P. Struss, “Context Modeling for Dynamic Configuration of Automotive Functions”, *ITSC*, 2013
- [9] C. Richter, “Development of an interactive car configuration system”, *Master’s Thesis, Tech. Univ. of Munich*,
- [10] A. Verikios, “A tool for the Configuration of CERN Particle Beam Measurement Systems”, *Master’s Thesis, Tech. Univ. of Munich*,
- [11] U. Junker, “Configuration.” *Handbook of Constraint Programming, Configuration*, p. 837-868, 2006.
- [12] J. De Kleer, BC Williams, “Diagnosis with Behavioral Model”, *IJCAI*, 1993.
- [13] J De Kleer, “An assumption-based TMS” *Artificial intelligence*, 1986.
- [14] J De Kleer, “Hitting Set Algorithms for Model-based Diagnosis”, *Principles of Diagnosis*, 2011.

Decentralised Fault Diagnosis of Large-scale Systems: Application to Water Transport Networks

Vicenç Puig and Carlos Ocampo-Martinez

Universitat Politècnica de Catalunya - BarcelonaTech
 Institut de Robòtica i Informàtica Industrial, CSIC-UPC
 Llorens i Artigas, 4-6, 08028 Barcelona, Spain
 e-mail: {vpuig, cocampo}@iri.upc.edu

Abstract

In this paper, a decentralised fault diagnosis approach for large-scale systems is proposed. This approach is based on obtaining a set of local diagnosers using the analytical redundancy relation (ARRs) approach. The proposed approach starts with obtaining the set of ARRs of the system yielding into an equivalent graph. From that graph, the graph partitioning problem is solved obtaining a set of ARRs for each local diagnoser. Finally, a decentralised fault diagnosis strategy is proposed and applied over the resultant set of partitions and ARRs. In order to illustrate the application of the proposed approach, a case study based on the Barcelona drinking water network (DWN) is used.

1 Introduction

Large-scale systems (LSS) present new challenges due to the large size of the plant and its resultant model [1, 2]. Traditional supervision methods for LSS (including diagnosis and fault tolerant control) have been mostly developed assuming a centralized scheme that assumes to have the full information. In the same way, a global dynamical model of the system is considered to be available for supervision design (off-line). Moreover, all measurements must be collected in one location in a centralised way. When considering LSS, the centrality assumption usually fails to hold, either because gathering all measurements in one location is not feasible, or because a centralised high-performance computing unit is not available. These difficulties have recently led to research in fault diagnosis (and fault-tolerant control) algorithms that operate in either decentralised or distributed way. Depending on the degree of interaction of the diagnoser associated to the subsystems and their diagnosis process, they can be classified into decentralised and distributed diagnosis categories.

In the decentralised diagnosis, both a central coordination module and a local diagnoser for each subsystem that forms the whole supervision system are running in parallel. Some examples were presented in [3, 4, 5], where local diagnosers are communicated to a coordination process (supervisor), obtaining a global diagnosis. On the other hand, in the distributed approach, a set of local diagnosers share information by means of some communication protocol instead of requiring a global coordination process such as in a decentralised approach. In the related literature, there are

several proposals where there is no centralised control structure or coordination process among diagnosers [6, 7, 8]. Every diagnoser shares information with the neighbouring diagnosers. In these systems the model is distributed, the diagnosis is locally generated and the consistency among the subsystems should be satisfied.

In this paper, the main contribution relies on the development of a decentralised fault diagnosis approach for LSS based on analytical redundancy relations (ARRs) and graph theory. The algorithm starts considering a set of ARRs and then stating an equivalent graph. From that graph, the problem of graph partitioning is then solved. The resultant partitioning consists of a set of non-overlapped subgraphs whose number of vertices is as similar as possible and the number of interconnecting edges between them is minimal. To achieve this goal, the partitioning algorithm applies a set of procedures based on identifying the highly connected subgraphs with balanced number of internal and external connections in order to minimize the degree of coupling among the resulting partitions (diagnosers). This algorithm is specially useful in systems where there is no a clear functional decomposition. Finally, a decentralised fault diagnosis strategy is introduced and applied over the resultant set of partitions, in a similar way to the one introduced in [5]. In order to illustrate the application of the proposed approach, a case study based on the Barcelona drinking water network (DWN) is used.

The remainder of this paper is organised as follows. Section 2 presents and discusses the overall problem statement. Section 3 presents the ARR graph partitioning methodology. Section 4 describes the proposed decentralised fault diagnosis approach. Section 5 shows both the considered case study and the way of implementing the proposed decentralised fault diagnosis approach. Finally, Section 6 draws the main conclusions.

2 Problem Statement

2.1 Fault Diagnosis using ARRs

Consider a dynamical system represented in general form by the state-space model

$$x^+ = g(x, u, d), \quad (1a)$$

$$y = h(x, u, d), \quad (1b)$$

where $x \in \mathbb{R}^n$ and $x^+ \in \mathbb{R}^n$ are, respectively, the vectors of the current and successor system states (that is, at time instants k and $k + 1$, respectively if the model is expressed in discrete-time), $u \in \mathbb{R}^m$ is the system input vector, $d \in$

\mathbb{R}^p is the vector containing a bounded process disturbance and $y \in \mathbb{R}^q$ is the system output vector. Moreover, $g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}$ is the states mapping function and $h : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}^q$ corresponds with the output mapping function.

The design of a model-based diagnosis system is based on utilizing the system model (1) in the construction of the diagnosis tests. According to [9], by means of the structural analysis tool and perfect matching algorithm, a set of ARR, namely \mathcal{R} , can be derived from (1). ARR are constraints that only involve measured variables (y, u) and known parameters θ . The set of ARR can be represented as

$$\mathcal{R} = \{r_i \mid r_i = \Psi_i(y_k, u_k, \theta_k), i = 1, \dots, n_r\}, \quad (2)$$

where Ψ_i is the ARR mathematical expression and n_r is the number of obtained ARR. Then, fault diagnosis is based on identifying the set of consistent ARR

$$\mathcal{R}_0 = \{r_i \mid r_i = \Psi_i(y_k, u_k, \theta_k) = 0, i = 1, \dots, n_r\}, \quad (3)$$

and inconsistent ARR,

$$\mathcal{R}_1 = \{r_i \mid r_i = \Psi_i(y_k, u_k, \theta_k) \neq 0, i = 1, \dots, n_r\}, \quad (4)$$

at time instant k when some inconsistency in (2) is detected [10]. Fault isolation task starts by obtaining the *observed fault signature*, where each single fault signal indicator $\phi_i(k)$ is defined as follows:

$$\phi_i(k) = \begin{cases} 0 & \text{if } r_i(k) \in \mathcal{R}_0, \\ 1 & \text{if } r_i(k) \in \mathcal{R}_1. \end{cases} \quad (5)$$

Fault isolation is based on the knowledge about the binary relation between the considered fault hypothesis set $\{f_1(k), f_2(k), \dots, f_{n_f}(k)\}$ and the fault signal indicators ϕ_i that are stored in the *fault signature matrix* M . An element of this matrix, namely m_{ij} , is equal to 1 if the fault hypothesis f_j is expected to affect the residual r_i such that the related fault signal ϕ_i is equal to 1 when this fault is affecting the monitored system. Otherwise, the element m_{ij} is zero-valued. A column of this matrix is known as a *theoretical fault signature*. Then, the fault isolation task involves finding a matching between the observed fault signature with some of theoretical fault signatures.

2.2 Partitioning the Set of ARR

In order to design a decentralised fault diagnosis system following the ARR approach recalled above, the set of ARR in (2) should be decomposed into subsets with minimal degree of coupling. Each subset of ARR will allow to implement a local diagnoser. With this aim, a graph representation of \mathcal{R} in (2) is determined. The graph $G(V, E)$ representing the set of ARR is obtained considering that

- the ARR are the graph vertices collected in a set V , and
- the measured input/output variables are the graph edges collected in a set E .

The graph incidence matrix I_M is obtained considering that, without loss of generality, the directionality of the edges are derived from the relation between ARR (rows of I_M) and input/output variables (columns of I_M), in analog way as proposed by [11] (and references therein) for the partitioning of LSS¹. Once I_M has been obtained from the ARR

¹There are alternative matrix representations for a graph such as the *adjacency matrix* and the *Laplacian matrix* (see [12]), which are related to the matrix representation used in this paper.

graph, the problem consists in partitioning the graph \mathcal{R} into subgraphs. Since such partitioning is oriented to the application of a decentralised fault diagnosis, it is convenient that the resultant subgraphs have the following features:

- nearly the same number of vertices;
- few connections between the subgraphs.

These features guarantee that the obtained subgraphs have a similar size, fact that balances computations between local diagnosers and allows minimising communications with a supervisory diagnoser. Hence, the partitioning the ARR graph can be more formally established following the dual problem proposed in [13] as stated here in Problem 1.

Problem 1 (ARR Graph Partitioning Problem). *Given a graph $G(V, E)$ obtained from a set of ARR, where V denotes the set of vertices, E is the set of edges, and $p \in \mathbb{Z}_{\geq 1}$, find p subsets V_1, V_2, \dots, V_p of V such that*

1. $\bigcup_{i=1}^p V_i = V$,
2. $V_i \cap V_j = \emptyset$, for $i \in \{1, 2, \dots, p\}$, $j \in \{1, 2, \dots, p\}$, $i \neq j$,
3. $\#V_1 \approx \#V_2 \approx \dots \approx \#V_p$,
4. the cut size, i.e., the number of edges with endpoints in different subsets V_i , is minimised.

Remark 2.1. *Conditions 3 and 4 of Problem 1 are of high interest from the point of view of a decentralised scheme since they are related to the degree of interconnection between resultant subsystems and their size balance.* \square

Remark 2.2. *The inclusion of additional specifications directly related to the FDI performance of each subsystem diagnoser will be addressed as a future extension of the proposed partitioning approach.* \square

Remark 2.3. *The partitioning approach starts from a given set of ARR obtained using the perfect matching algorithm. The selection of the best ARR from the set of the all possible ARR (that could be obtained using the available sensors and system structure) such that when applying the partitioning algorithm produces a set of diagnosers with good FDI performance could be considered as an additional future improvement.* \square

In general, graph partitioning approaches are considered as \mathcal{NP} -complete problems [2]. However, they can be solved in polynomial time for $\#V_i = 2$ (Kernighan-Lin algorithm); see, e.g., [14]. Since the latter condition is quite restrictive for large-scale graphs, alternatives for graph partitioning based on fundamental heuristics are properly accepted and broadly discussed.

3 Proposed Partitioning Approach

Starting from the system ARR graph obtained as described in Section 2, this section proposes a partitioning algorithm through which a decomposition of the set of system ARR can be performed. This decomposition allows the splitting of a centralised diagnoser into local diagnosers. The philosophy of the proposed approach comes from the partitioning methodology reported in [13], where a dynamic system is decomposed into several subsystems following certain criteria towards fulfilling a set of design conditions. For completeness and full understanding of the proposed diagnosis

methodology, that approach is explained below and suitably adapted if needed.

The algorithm is divided into the main kernel and auxiliary routines in order to refine the final result according to the nature of the system and the given criteria depending on the case. Here, the ARR graph is decomposed into subgraphs in the same way as a system would be divided into subsystems.

3.1 Main Kernel

This part performs the central task of defining how the equivalent ARR graph of the LSS is split into subgraphs. The steps of the algorithm are followed in the form of subroutines towards reaching the main goals outlined in Problem 1. Notice that the whole algorithm is used off-line, i.e., the partitioning of the ARR graph is not carried out dynamically on-line. Ongoing research is focused to adapt the proposed algorithm such that the partitioning could be performed on-line when some structural change of the network occurs. The different subroutines are briefly described next.

- The *start-up* routine, which requires the matrix-based definition of the graph, e.g., via the *incidence matrix*, in order to state the connections between the graph vertices.
- The *preliminary partitioning* routine, which performs a clustering-like procedure where all graph vertices are assigned to a particular subset according to predefined indices related to the resultant subgraph and its *internal weight* (defined as the number of vertices of a subgraph), its *external weight* (defined as the number of shared edges between subgraphs) and other statistical measures. The resultant amount of partitions at this stage is automatically obtained.
- The *uncoarsening* routine, which is applied for reducing the number of resultant subgraphs if their internal weight is unbalanced, which would produce partitions with large differences of amount of vertices. This routine defines a design parameter φ_{\max} for determining the variance of the internal weight for all the resultant subgraphs.
- The *refining* routine, which aims at reducing the cut size of the resultant subgraphs, i.e., the number of edges they share. This routine is based on the connectivity of the vertices of a subgraph with other vertices in the same subgraph and in neighbouring subgraphs².

Applying the aforementioned routines to the entire ARR graph, the expected result consists of a set of subgraphs that determines a particular decomposition. This set P is finally defined as

$$P = \left\{ G_i, i = 1, 2, \dots, p : \bigcup_{i=1}^p G_i = G \right\}. \quad (6)$$

3.2 Auxiliary Routines

Although the decomposition algorithm yields to an automatic partitioning of a given graph, it does not imply that the resultant set P follows the pre-established requirements stated in Problem 1. Therefore, complementary routines enhance the partitioning routine depending on their tune

²Two subgraphs are called *neighbours* if they are contiguous and share edges (see, e.g., [15] among many others).

for the particular case study. Additional auxiliary routines might be designed in such a way that the diagnosis performance that would be achieved when used in decentralised or distributed fault diagnosis is taken into account. These auxiliary routines are:

- The *pre-filtering* routine, which lightens the start-up routine by merging all these vertices with single connection to those to which they are connected. It allows to have a smaller initial graph and then performing faster clustering of vertices.
- The *post-filtering* routine, which adds a tolerance parameter δ in such a way that the uncoarsening routine yields in less subgraphs when two of them may be conveniently merged but the numerical constraints does not allow to do so. This routine might increase the complexity since the internal weight of some subgraphs would also increase, unbalancing the resultant set of partitions.
- The *anti-oscillation* routine, which leads to solve a possible issue when the refining (external balance) routine is run since it defines a maximum number of iterations ρ that the refining routine is executed.

4 Decentralised Fault Diagnosis

Once a partitioned set of ARRs has been obtained by means of the algorithm presented in Section 3, the decentralised fault diagnosis approach is introduced. In order to explain how the proposed fault diagnosis approach works, it is concentrated on faults affecting the sensors measuring the input/output variables implied in the ARRs. The approach could be easily extended to other type of faults, but in order to keep the explanation simpler, it is restricted to the discussion about the set of considered faults. In this way, a fault can be associated to each measured input/output variable.

Each subset of ARRs will allow to implement a *local diagnoser* D_i in the way described in Section 2.1. The ARRs associated to a local diagnoser can be split in two groups. The first group, named in the following *local ARRs*, is composed of ARRs that do not involve shared variables with other ARRs in a different local diagnoser. On the other hand, the second group, named *shared ARRs*, is composed by ARRs that involve shared variables. Figure 1 shows two sets of ARRs associated to two local diagnosers, named D_2 and D_4 . These two diagnosers share some variables (in this case only outputs, but can be both inputs and outputs). This set of shared variables allows to define the set of shared ARRs, named D_C in the figure. The remaining ARRs, which do not share variables, are local ARRs.

Similarly, faults in the fault signature matrix M of the local diagnoser that only involve local ARRs can be *locally diagnosed*. Thus, the local diagnoser works in a decentralised manner regarding those faults. On the other hand, faults that involve ARRs with shared variables in different subgraphs can not be locally diagnosed. On the contrary, a *global diagnoser* that evaluates the involved ARRs is used. This diagnoser has a fault signature matrix M collecting the involved ARRs with shared variables between local diagnosers and faults that should be *globally diagnosed*. When local diagnosers evaluate an ARR composed of shared variables, they send the result of the consistency check to the global diagnoser, which proceeds with the global diagnosis using a fault signature matrix that contains the involved ARRs. As

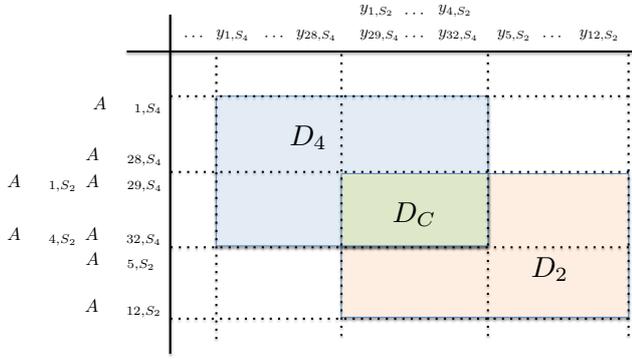


Figure 1: Subsets of ARRs of two local diagnosers sharing some variables

a result of the global diagnosis based on the involved ARRs with shared variables, a fault in these variables could be diagnosed or alternative excluded. In case of exclusion, local diagnosers sharing a given ARR whose shared variable has been considered non-faulty continue reasoning now with all ARRs, i.e., all the involved ones, proposing a fault candidate using the local fault signature.

5 Application to a Case Study

This section briefly describes a case study in order to exemplify the application of the proposed decentralised diagnosis approach in a real LSS. In particular, the transport infrastructure of the Barcelona Drinking Water Network (DWN) is used.

5.1 Case Study Description

The Barcelona DWN, managed by Aguas de Barcelona, S.A. (AGBAR), supplies drinking water to Barcelona city and its metropolitan area through four drinking water treatment plants: the Abrera and Sant Joan Despí plants, which extract water from the Llobregat river, the Cardedeu plant, which extracts water from Ter river, and the Besòs plant, which treats the underground flows from the aquifer of the Besòs river. All source together provide a total amount of flow of around $7 \text{ m}^3/\text{s}$. The water flow from each source is limited, what implies different water prices depending on water treatments and legal extraction canons. See [16] for further information about this system and [17] for further details about its modelling and management criteria.

5.2 Monitoring-oriented Model

In order to obtain a monitoring-oriented model of the DWN, the constitutive network elements (i.e., tanks, actuators, water demand sectors, nodes and sources) as well as their basic relationships should be stated [16].

By considering the mass balance at tanks and the static relations at α network nodes, the monitoring-oriented discrete-time state-space model of the DWN can be written as

$$x_{k+1} = Ax_k + \Gamma\nu_k, \quad (7a)$$

$$E_1\nu_k = E_2, \quad (7b)$$

$$y_k = Cx_k, \quad (7c)$$

with $\Gamma = [B \ B_p]$, $\nu_k = [u_k^T \ d_k^T]^T$, where $x \in \mathbb{R}^n$ is the state vector corresponding to the water volumes of the n tanks, $u \in \mathbb{R}^m$ represents the vector of manipulated flows

through the m actuators (pumps and valves), $d \in \mathbb{R}^q$ corresponds to the vector of the q water demands (sectors of consume) and $y \in \mathbb{R}^n$ are the vector of measured water volumes of the n tanks. In this case, the difference equations in (7a) describe the dynamics of the storage tanks, the algebraic equations in (7b) describe the static relations (i.e., mass balance at junction nodes) in the network and in (7c) describe the relation between the physical and measured tank volumes. Moreover, A , B , B_p , C , E_1 and E_2 are system matrices of suitable dimensions dictated by the network topology.

5.3 Implementation of the Proposed Approach

This section discusses the way the proposed decentralised fault diagnosis approach is implemented in the considered real case study. Figure 2 corresponds to the aggregate model of the Barcelona DWN, which is a simplification of the complete model, where groups of elements have been aggregated (not discarded) in single nodes to reduce the size of the whole network model. Using this aggregate model, the ARR graph of the Barcelona DWN has been derived after generating the set of ARRs from the mathematical model (7) by using the perfect matching algorithm [9] that aims to find a causal assignment which associates unknown system variables with the system constraints from which they can be calculated. Applying the partitioning algorithm to this graph, five groups of ARRs are obtained, which corresponds to five diagnosers that monitor a different part of the Barcelona DWN represented with different colors in Figure 2. Table 2 collects the descriptions of the resultant subgraphs, their number of ARRs and shared variables (manipulated flows through actuators) represented using circles in Figure 2. At this point it should be recalled that one of the goals of the partitioning algorithm is to reduce as much as possible the number of shared edges between subgraphs obtaining a graph decomposition as less interconnected as possible and with similar number of vertices for each subsystem (internal weight). This will allow an easier global diagnosis configuration, not only with respect to the number of distributed diagnosers but also with respect to the complexity of each local diagnoser D_i . Thus, the application of the approach to the Barcelona DWN implies the design of five decentralised diagnosers together with a centralised/supervisory one, which is in charge of the coupled relations within the corresponding fault signature matrix of the whole system.

Table 1: Barcelona DWN subsystems and number of both shared elements and ARRs

Number	Color	# ARRs	# Shared variables
1	green	4	1
2	red	5	5
3	yellow	8	6
4	blue	8	16
5	purple	5	5

For this example, it is important to highlight that ARRs have been obtained by considering the following assumption.

Assumption 5.1. *Fault in actuators are only taken into account. Sensors are supposed to operate properly.* \square

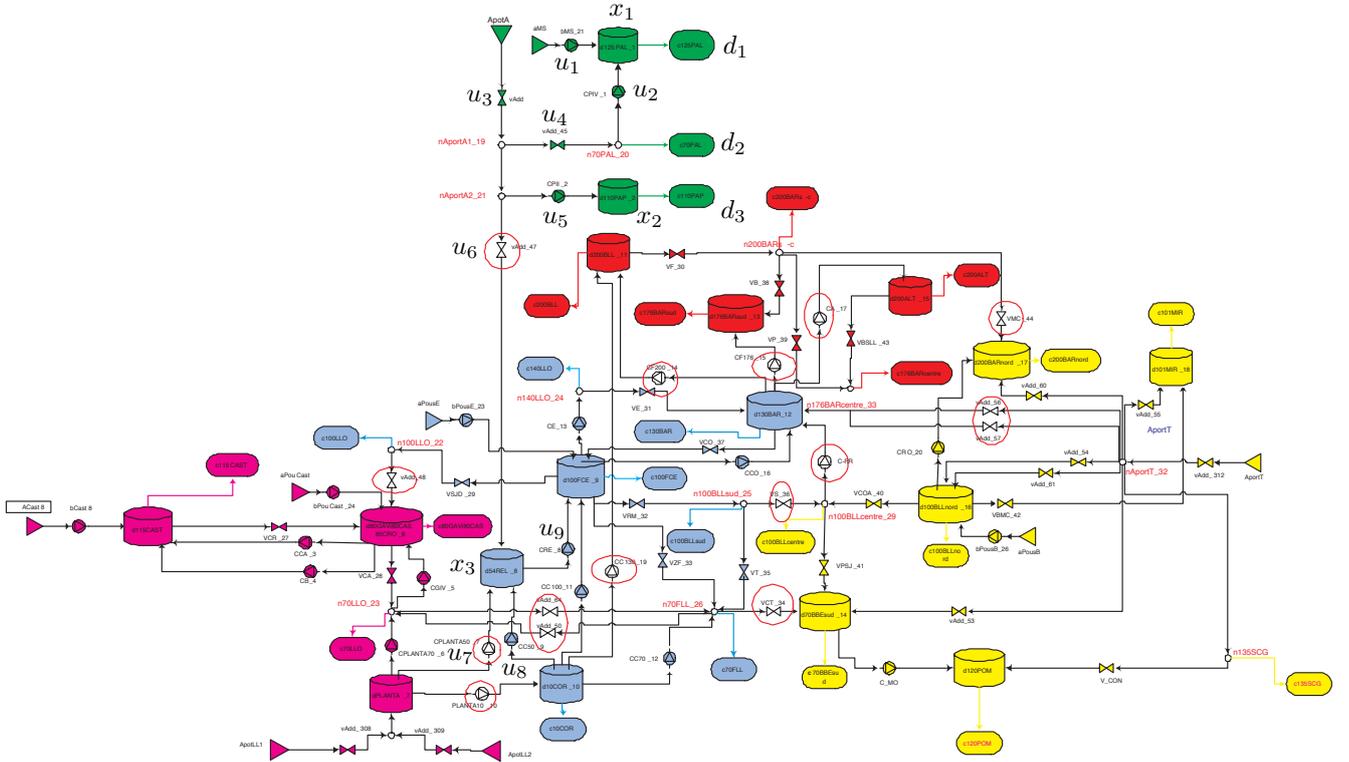


Figure 2: ARR Partitioning of the Barcelona DWN

Table 2: Barcelona DWN subsystems and number of both shared elements and ARRs

Number	Color	# ARRs	# Shared variables
1	green	4	1
2	red	5	5
3	yellow	8	6
4	blue	8	16
5	purple	5	5

In order to easily understand how the proposed decentralised fault diagnosis approach would work, it will be explained focusing on subsystems S_1 and S_4 presented in Figure 3 in red lines that corresponds to the subsystems in green (S_1) and in blue (in S_4) in Figure 2. In particular, considering the set of ARRs corresponding to S_1 as

$$r_{1,k}^{S_1} = y_{1,k} - y_{1,k-1} - \Delta t[u_{1,k-1} + u_{2,k-1} - d_{1,k-1}],$$

$$r_{2,k}^{S_1} = u_{1,k} - u_{2,k} - d_{2,k},$$

$$r_{3,k}^{S_1} = y_{2,k} - y_{2,k-1} - \Delta t[u_{5,k-1} - d_{3,k-1}],$$

$$r_{4,k}^{S_1} = u_{3,k} - u_{4,k} - u_{5,k} - u_{6,k},$$

the fault signature matrix presented in Table 3 can be obtained. From this table, it is possible to identify the shadowed part, which corresponds to the faults that the local diagnoser D_1 is able to isolate when a fault activates any of the ARRs $r_{i,k}$, $i = 1, 2, 3$, since those ARRs only involve local variables. However, if the residual $r_{4,k}$ is activated, it is necessary that a global diagnoser interacts with D_1 discriminating whether the corresponding ARR in S_4 , defined here as $r_{1,k}^{S_4}$, was also activated. If this is the case, the element

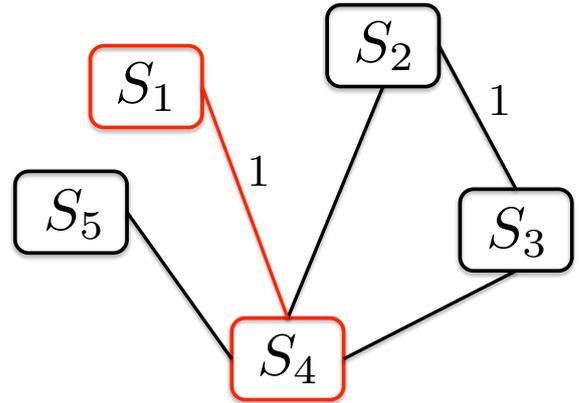


Figure 3: Scheme of decentralised diagnoser scheme for the Barcelona DWN resultant subsystems and their number of shared variables

 Table 3: Fault signature matrix of S_1

ARR	f_{y_1}	f_{u_1}	f_{u_2}	f_{y_2}	f_{u_5}	f_{u_3}	f_{u_4}	f_{u_6}
$r_{1,k}^{S_1}$	X	X	X					
$r_{2,k}^{S_1}$		X	X					
$r_{3,k}^{S_1}$				X	X			
$r_{4,k}^{S_1}$					X	X	X	X

u_6 is then in fault and hence isolated. Otherwise, D_1 can decide locally (then isolating u_3 , u_4 or u_5).

In Table 4, the fault signature matrix for the ARRs that

Table 4: Part of the fault signature matrix accounting shared variables between S_1 and S_4

ARR	...	f_{u_5}	f_{u_6}	f_{u_7}	...
$r_{4,k}^{S_1}$		X	X		
$r_{1,k}^{S_4}$			X	X	

contain shared variables between both S_1 and S_4 is presented. There, $r_{4,k}^{S_1}$ corresponds with the fourth ARR of S_1 (last row of Table 3), while

$$r_{1,k}^{S_4} = x_{3,k} - x_{3,k-1} - \Delta t[u_{7,k-1} + u_{8,k-1} + u_{6,k-1} - u_{9,k-1}]$$

corresponds with the first defined ARR for S_4 . Notice that the global diagnoser should decide by looking at the ARR activations occurred in this fault signature matrix and then interact with the different local diagnosers if needed.

6 Conclusions

In this paper, a decentralised fault diagnosis approach for large-scale systems based on graph-theory has been presented. The algorithm starts with the translation of the system model into a graph representation. Then, applying the perfect matching algorithm, a set of analytical redundancy relations is obtained. From the analytical redundancy relation graph, the problem of graph partitioning is then solved. The resultant partition consists of a set of non-overlapped subgraphs whose number of vertices is as similar as possible and the number of interconnecting edges between them is minimal. To achieve this goal, the partitioning algorithm applies a set of procedures based on identifying the highly connected subgraphs with balanced number of internal and external connections. Finally, a decentralised fault diagnosis strategy is introduced and applied over the resultant set of partitions. In order to illustrate and discuss the use and application of the proposed approach, a case study based on the Barcelona DWN has been used. As further research, the partitioning algorithm will be improved by acting directly on the system model and not on the set of ARRs in order to generate a set of ARRs for each local diagnoser with enhanced fault diagnosis properties.

Acknowledgements

This work has been partially supported by the EFFINET grant FP7-ICT-2012-318556 of the European Commission and the Spanish project ECOCIS (Ref. DPI2013-48243-C2-1-R).

References

- [1] J. Lunze. *Feedback Control of Large-Scale Systems*. Prentice Hall, Great Britain, 1992.
- [2] D.D. Šiljak. *Decentralized control of complex systems*. Academic Press, 1991.
- [3] L. Console, C. Picardi, and D. Theseider Duprè. A framework for decentralized qualitative model-based diagnosis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 286–291, Hyderabad, India, 2007.
- [4] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1-2):121–170, 2005.
- [5] S. Indra, L. Travé-Massuyès, and E. Chanthery. Decentralized diagnosis with isolation on request for spacecraft. In *Fault Detection, Supervision and Safety of Technical Processes*, pages 283–288, México, 2012.
- [6] F. Boem, R.M.G. Ferrari, T. Parisini, and M. M. Polycarpou. Distributed fault diagnosis for continuous-time nonlinear systems: The input-output case. *Annual Reviews in Control*, 37(1):163 – 169, 2013.
- [7] J. Biteus, E. Frisk, and M. Nyberg. Distributed diagnosis using a condensed representation of diagnoses with application to an automotive vehicle. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 41(6):1262–1267, November 2011.
- [8] I. Roychoudhury, G. Biswas, and X. Koutsoukos. Designing distributed diagnosers for complex continuous systems. *IEEE Transactions on Automation Science and Engineering*, 6(2):277–290, April 2009.
- [9] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, Berlin, Heidelberg, second edition, 2006.
- [10] S. Tornil-Sin, C. Ocampo-Martinez, V. Puig, and T. Escobet. Robust fault diagnosis of nonlinear systems using interval constraint satisfaction and analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(1):18–29, Jan 2014.
- [11] A. I. Zečević and D. D. Šiljak. *Control of Complex Systems: Structural Constraints and Uncertainty*. Communications and Control Engineering. Springer, 2010.
- [12] J.A. Bondy and U.S.R. Murty. *Graph Theory*, volume 244 of *Graduate Series in Mathematics*. Springer, 2008.
- [13] C. Ocampo-Martinez, S. Bovo, and V. Puig. Partitioning approach oriented to the decentralised predictive control of large-scale systems. *Journal of Process Control*, 21(5):775–786, 2011.
- [14] T.N. Bui and B.R. Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996.
- [15] L. Addario-Berry, K. Dalal, and B. Reed. Degree-constrained subgraphs. *Discrete Applied Mathematics*, 156(7):1168–1174, 2008.
- [16] C. Ocampo-Martinez, V. Puig, G. Cembrano, R. Creus, and M. Minoves. Improving water management efficiency by using optimization-based control strategies: the Barcelona case study. *Water Science & Technology: Water supply*, 9(5):565–575, 2009.
- [17] C. Ocampo-Martinez, V. Puig, G. Cembrano, and J. Quevedo. Application of predictive control strategies to the management of complex networks in the urban water cycle [applications of control]. *IEEE Control Systems Magazine*, 33(1):15–41, 2013.

Self-Healing as a Combination of Consistency Checks and Conformant Planning Problems

Alban Grastien

Optimisation Research Group, NICTA
Artificial Intelligence Group, The Australian National University
Canberra Research Laboratory, Australia

Abstract

We introduce the problem of self healing, in which a system is asked to self diagnose and self repair. The two problems of computing the diagnosis and the repair are often solved separately. We show in this paper how to tie these two tasks together: a planner searches a prospective plan on a sample of the belief state; a diagnoser verifies the applicability of the plan and returns a state of the belief state (added to the sample) in which the plan is not applicable. This decomposition of the self healing process avoids the explicit computation of the belief state. Our experiments demonstrate that it scales much better than the traditional approach.

1 Introduction

Autonomous systems are subject to faults and require regular repair actions; systems capable of performing such tasks are called self healing. Finding the optimal repair involves solving a diagnosis problem (what may the current system state be?) together with a planning problem (what optimal/near optimal course of actions, applicable in all of the possible states, leads to an acceptable state?). In large, partially observable, systems computing an explicit “belief state” can be intractable; finding a plan applicable in all elements of this belief state can be also intractable.

In this paper we propose a method that avoids these two intractable problems. This method relies on the intuition that the full belief state is not necessary to find the appropriate repair. For instance, if a self-healing problem requires to make sure that n given machines are turned off and if the status (on or off) of these machines is unknown, then the belief state is comprised of 2^n states. However the optimal plan (*press the stop button on every machine*) happens to be the optimal plan of the state where none of the machines has been shut: this single state is “representative” of all the states in the belief state.

Our approach uses a planner to compute an optimal plan for a small sample of the belief state (at most dozens of elements); the plan is applicable in all these states and leads to the goal state. In order to validate the plan for the full belief state we search for an

element of the belief state in which the plan is not applicable. To this end we define a new type of diagnoser that solves the following problem: find a possible behaviour of the system (that agrees with the model and the observations) that ends up in a state q in which the plan is not correct; this state q is added to the sample of the belief state so that the planner finds a more suitable repair plan at the next iteration. Failure on the part of the diagnoser to find such a behaviour proves that the plan is indeed correct. In practice the problem of verifying the correctness of a plan is reduced to a propositional satisfiability (SAT) problem that is unsatisfiable iff the plan is applicable in all states and that returns a counterexample if not.

The contributions of this paper are i) a formal definition of the self-healing problem, ii) the solving of self-healing as a combination of diagnosis and planning steps, and iii) the reduction of each step to SAT.

This work is performed in the context of discrete event systems [Cassandras and Lafortune, 1999]. As opposed to supervisory control, where actions (either active or passive, such as forbidding some events) are performed while the system is running, we follow the work from Cordier *et al.* [2007] and assume that the repair is being performed whilst the system is inactive.

The paper is divided as follows. Next section defines the self-healing problem formally. Section 3 presents the proposed algorithm with a set-based perspective. The SAT implementation is presented in Section 4. Experimental validation is given in Section 5. A comparison with other problems and approaches is given in Section 6.

2 Problem Definition

The problem we are addressing is illustrated on Figure 1. We are concerned with finding the most appropriate repair for a partially observed system that has been running freely.

We assume that the system can run in two different modes: the “active” (and useful) mode in which the system is free to operate (left half of the figure) and the “repair” mode in which the system state is being re-adjusted (right half). The system behaves quite differently in the two modes. In the active mode, the system is partially observable but uncontrolled. In the repair mode, the system is not observed albeit controlled; the state changes only through explicit application of actions; and special attention must be made to their

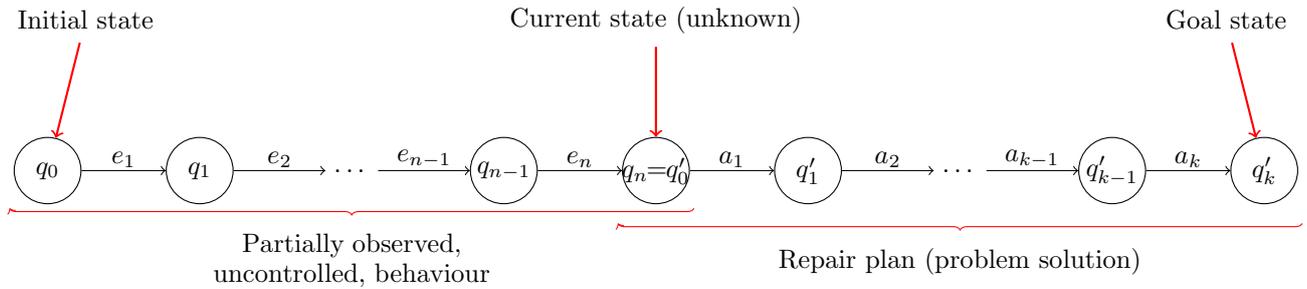


Figure 1: Schematic description of the self-healing problem: find a repair plan that returns the state in the goal set.

applicability/effects. One reason for assuming that the system does not run freely in the repair mode is that we do not want to consider scenarios where faults can occur during the repair, which would increase the overall complexity of the problem. We believe that this limitation, essentially the fact that the repair actions have deterministic effects, can be lifted.

2.1 Explicit Model

We are considering discrete event systems (DES, [Cassandras and Lafortune, 1999]). The system is modeled as a finite state machine, i.e., a finite set Q of states together with a set T of transitions labeled with finitely-many events/actions.

Definition 1 An explicit self-healing system model is a tuple $M = \langle Q, I, \Sigma, \Sigma_o, \Sigma_a, T, G, U \rangle$ where

- Q is a finite set of states, $I \subseteq Q$ is a set of initial states, $G \subseteq Q$ is a set of goal states, $U \subseteq Q$ is a set of unstable states,
- Σ is a finite set of events, $\Sigma_o \subseteq \Sigma$ is the set of observable events, $\Sigma_a \subseteq \Sigma$ is the set of actions, and
- $T \subseteq (Q \times \Sigma \times Q)$ is the set of transitions $\langle q, e, q' \rangle$ also denoted $q \xrightarrow{e} q'$.

In the active mode the system takes a path $\rho = q_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} q_n$ such that $\{e_1, \dots, e_n\} \subseteq \Sigma \setminus \Sigma_a$, $q_0 \in I$ and $q_n \notin U$. This last condition is used to prevent situations where a fault happened right before the repair is applied, i.e., before any observation of this fault was made. This assumption is similar to the one made, e.g., by Lamperti and Zanella that the system is *quiescent* (no more event is about to happen) when diagnosis is performed [Lamperti and Zanella, 2003]. This assumption can be removed by assuming $U = Q$. Finally the observation $O = \text{obs}(\rho)$ of this path is the projection of e_1, \dots, e_n on the observable events Σ_o (i.e., all non-observable events are eliminated from the sequence).

In the repair mode a sequence of actions, called a plan $\pi = a_1, \dots, a_k$, is applied ($\{a_1, \dots, a_k\} \subseteq \Sigma_a$). From state $q'_0 \in Q$, the application of π leads to the (single) state $q'_k = \pi(q'_0)$ such that $q'_0 \xrightarrow{a_1} \dots \xrightarrow{a_k} q'_k$. We assume that every action is applicable in every state (if this is not the case a non-goal sink state can be created where all inapplicable actions lead to) and have deterministic effects. If π leads q'_0 to a goal state, we say that π is correct for q'_0 .

Notice that a plan is a simple sequence: we do not assume that additional observations are available at runtime. There is no probing action available. After non deterministic action effects, the use of conditional plans is a second natural extension of this work.

Definition 2 The self-healing problem is a pair $P = \langle M, O \rangle$ where M is a model and O is an observation. A repair plan for P is a plan that is guaranteed to be correct in the current state. Formally a repair plan is a plan π such that

$$(q_0 \in I \wedge \text{obs}(\rho) = O \wedge q_n \notin U) \Rightarrow \pi(q_n) \in G. \quad (1)$$

The set of repair plans is denoted $\Pi(M, O)$ or simply Π .

Given a cost function on sequences of actions, the objective of the self-healing problem is to find a cost-minimal repair plan (for simplicity we assume that such a plan exists):

$$\pi^* = \arg \min_{\pi \in \Pi} \text{cost}(\pi).$$

This definition assumes a cost function that provides a total order on the plans. In practice we will try to minimise the number of actions (all actions have the same cost, the cost is cumulative) and break ties at random.

We see two main categories of self-healing problems, namely i) a recurring situation where the system is stopped regularly, which provides a good opportunity to perform corrective actions on the system; ii) a situation where a diagnoser/monitor detects an anomaly on the system and triggers a self-healing procedure. The present work is independent from how the problem was prompted.

2.2 Solving the Problem Explicitly

This paper works under the assumption that the system model is very large and that it is impractical to manipulate sets of states. We discuss this issue here and present some notations.

The simplest way to solve the self-healing problem is to compute the belief state and then compute the optimal plan for this set of states.

Given a model M and the observation O , the belief state \mathcal{B}^O is defined as the set of states that the system

could be in:

$$\mathcal{B}^O = \{q \in Q \mid \exists \rho = q_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} q_n, q_0 \in I \wedge \text{obs}(\rho) = O \wedge q_n \notin U \wedge q = q_n\}.$$

Notice that the definition of the belief state matches the first part of Equation (1).

A *conformant plan* for the set of states \mathcal{B}^O is a plan π that is correct for all states of \mathcal{B}^O : $\forall q \in \mathcal{B}^O. \pi(q) \in G$ (cf. Figure 2). Compared to the general definition of a conformant plan (a more detailed comparison is given in Section 6) we only deal with uncertainty on the initial state and we assume that actions have deterministic effects. Conformant planning is provably PSPACE-hard for explicit models.

We consider the conformant planning problem from the initial set of states \mathcal{B}^O and use $b = |\mathcal{B}^O|$ to denote the size of \mathcal{B}^O . The problem can be solved by considering the finite state machine M' where each state of M' is a set of states of the original model and each transition from state S labeled by action a leads to $S' = \{q' \in Q \mid \exists q \in S. \langle q, a, q' \rangle \in T\}$. The initial state of M' is \mathcal{B}^O ; a state S of M' is a goal state if it satisfies $S \subseteq G$. A plan π is a sequence of actions such that $\pi(\mathcal{B}^O)$ (in M') is a goal state. Because the original model is deterministic the transition $\langle S, a, S' \rangle$ is such that the size of S' is smaller than S . The number of states in M' is bounded by the sum of binomial coefficients $\binom{|Q|}{1} + \dots + \binom{|Q|}{b}$.

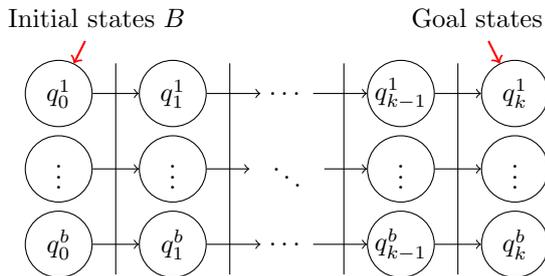


Figure 2: Solving conformant problems; the vertical lines mean that the transitions are labeled by the same action.

The model M' presented before cannot be easily expressed in planning modeling languages such as STRIPS or PDDL, or implemented in SAT. Another reduction, to M'' , can be introduced whose states are tuples (with b elements) of states from the original model: $Q'' = Q^b$. A tuple state is a goal state if all its elements are in the goal: $G'' = G^b$. The transitions in M'' correspond to the parallel execution of the same action in each state of the tuple (represented by the vertical lines on Figure 2).

In general M'' is larger than M' . The model also contains symmetries that efficient implementations might need to address explicitly: for instance in model M'' states $\langle q_1, q_2 \rangle$ and $\langle q_2, q_1 \rangle$ are different while they would be the same in M' : $\{q_1, q_2\} = \{q_2, q_1\}$.

Clearly this type of approach is only applicable if \mathcal{B}^O comprises no more than a few dozen elements.

Finally we look at a formulation of the planning problem that is complementary to the computation of the belief state. Assume that a plan π is given and we want to compute the set of states \mathcal{B}^π in which the plan π is correct: $\mathcal{B}^\pi = \{q \in Q \mid \pi(q) \in G\}$.

Lemma 1 *Plan π is a correct plan iff $\mathcal{B}^O \subseteq \mathcal{B}^\pi$.*

Writing $\overline{\mathcal{B}^\pi} \stackrel{\text{def}}{=} Q \setminus \mathcal{B}^\pi$ the set of states for which π is not correct, plan π is a correct plan iff $\mathcal{B}^O \cap \overline{\mathcal{B}^\pi} = \emptyset$.

3 Set Formulation of Self-Healing

We first present a formulation of our solution that is based on sets and that does not consider implementation issues (presented in the next section).

We propose a lazy approach to self-healing. In this approach we search a correct plan for a sample of the belief state (a “belief sample”) and then search for a state of the belief state in which the plan is not applicable; this state is added to the sample and the procedure is iterated again until a robust plan has been found.

We first give the theoretical results that justify the algorithm presented at the end of the section.

In the following we use the notations \mathcal{B}^O and B to represent sets of states such that $B \subseteq \mathcal{B}^O$. \mathcal{B}^O will represent the belief state and B a small subset (a few elements) of \mathcal{B}^O . S, S' will represent any set of states.

Let $\Pi(q)$ be the set of repair plans that are correct for state q . Let $\Pi(S)$ be the set of repair plans that are correct whichever is the current state from S . Then $\Pi(S) = \bigcap_{q \in S} \Pi(q)$. Notice that $\Pi = \Pi(\mathcal{B}^O)$.

A trivial result is:

$$S \subseteq S' \Rightarrow \Pi(S) \supseteq \Pi(S').$$

A consequence of this proposition is that the optimal repair for \mathcal{B}^O is a correct plan for B . Computing the optimal repair plan for the latter may therefore yield the optimal plan for the former. Let $\pi^*(S)$ be the optimal plan for a set of states. The next proposition determines how to characterize that an optimal plan was found:

$$S \subseteq S' \wedge (\pi^*(S) \in \Pi(S')) \Rightarrow \pi^*(S) = \pi^*(S').$$

This result can be derived from the previous proposition. $\pi^*(S')$ belongs to $\Pi(S)$ since $S \subseteq S'$; therefore $\pi^*(S)$ is better than (or equal to) $\pi^*(S')$. However, if $\pi^*(S) \in \Pi(S')$ and yet $\pi^*(S') \neq \pi^*(S)$, then $\pi^*(S')$ must be strictly better than $\pi^*(S)$, which contradicts what was just said.

Applied to $S = B$ and $S' = \mathcal{B}^O \supseteq B$, this means that $\pi^*(B) \in \Pi(\mathcal{B}^O)$ implies $\pi^*(B) = \pi^*(\mathcal{B}^O)$.

We reuse the notation \mathcal{B}^π for the set of states in which the plan π is correct, and $\overline{\mathcal{B}^\pi} = Q \setminus \mathcal{B}^\pi$ for the set of states in which it is not. With this notation, $\pi^*(B) \in \Pi(\mathcal{B}^O)$ is equivalent to $\mathcal{B}^O \cap \overline{\mathcal{B}^{\pi^*(B)}} = \emptyset$.

Assume that there exists a procedure *verify_applicability*(S, π) that extracts a state $q \in S \cap \overline{\mathcal{B}^\pi}$ if such a state exists, and returns \perp otherwise. Then, for $S \subseteq S'$, the following results are trivial:

- *verify_applicability*($S', \pi^*(S)$) = $\perp \Rightarrow \pi^*(S) = \pi^*(S')$;

- let $q = \text{verify_applicability}(S', \pi^*(S)) \neq \perp$ be a state where $\pi^*(S)$ is not applicable, then $q \notin S$ and $\pi^*(S \cup \{q\}) \neq \pi^*(S)$ (and $\text{cost}(\pi^*(S \cup \{q\})) > \text{cost}(\pi^*(S))$)¹.

The first proposition shows that *verify_applicability* can be used to check whether the plan $\pi^*(B)$ is correct for \mathcal{B}^O . The second proposition indicates how a better prospective plan can be computed if $\pi^*(B)$ is not correct: the addition of q to S guarantees that a different plan will be generated.

These results lead to the procedure presented in Algorithm 1. In this procedure, *find_plan*(B) is a method that computes a conformant plan from B as defined at the end of the previous section (and described next section). The procedure computes the optimal plan for a belief sample B . If *verify_applicability* finds a state $q \in \mathcal{B}^O$ in which this plan is not correct, then this state is added to the belief sample and a new optimal plan is generated and tested.

Algorithm 1 Diagnosis algorithm for the self-healing problem without enumerating the belief state \mathcal{B}^O

```

 $B := \emptyset$ 
loop
   $\pi := \text{find\_plan}(B)$ 
   $q := \text{verify\_applicability}(\mathcal{B}^O, \pi)$ 
  if  $q = \perp$  then
    return  $\pi$ 
  else
     $B := B \cup \{q\}$ 
  end if
end loop
    
```

Because i) each loop iteration adds an element to B and ii) \mathcal{B}^O is finite, this procedure is guaranteed to terminate. The number of iteration is, in the worst case, the size of \mathcal{B}^O ; we expect however that a handful of calls to *find_plan*(\cdot) will be sufficient to find the optimal plan.

Example

We illustrate Algorithm 1 with the example of Figure 3. Assume that the observations are $O = [o_1, o_2]$. According to the model, the belief state is $\mathcal{B}^O = \{A, D, F, H\}$ (state B is unstable, so the system cannot be in this state). The state needs to be returned to a subset of $\{A, G\}$.

Since the belief sample B_0 is initially empty, Algorithm 1 first generates the empty plan $\pi_0 = \varepsilon$. The procedure *verify_applicability* exhibits state F such that $B \xrightarrow{u} D \xrightarrow{o_1} E \xrightarrow{o_2} F$ could explain O and such that plan π_0 does not lead to a goal state when applied from F . The optimal plan for $B_1 = \{F\}$ is $\pi_1 = a_1$. This time *verify_applicability* extracts state H which also belongs to the belief state and for which the application of a_1 leads to sink state I . The belief sample B_2 now equals $\{F, H\}$ and the optimal conformant plan for B_2 is $\pi_2 = a_2, a_1$ (remember that unobservable transition $F \xrightarrow{u} H$ cannot trigger after the execution of a_2). This plan is correct for all elements in the belief state. Notice

¹Remember that no two plans have the same cost.

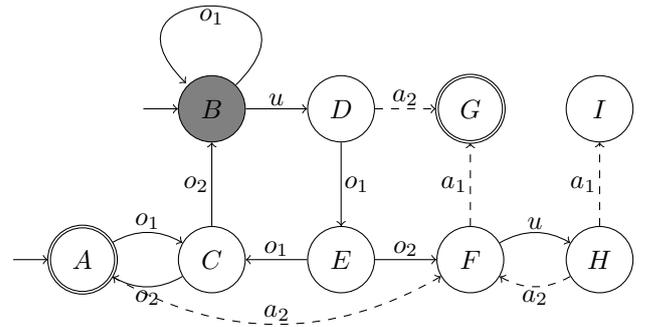


Figure 3: System example with two initial states (A and B), two goal states (A and G), one unstable state (B), two observable events (o_1 and o_2), and two actions (a_1 and a_2 ; an action affects the system state only if there is a transition).

that neither A nor D from \mathcal{B}^O were explicitly generated during the procedure.

4 SAT Formulation of Self-Healing

In this section we show how Algorithm 1 can be implemented using SAT. This implementation assumes a symbolic representation of the model, i.e., a representation where states and transitions are not enumerated but are, instead, implicitly defined by a set V of Boolean state variables (aka fluents) as can be found, e.g., in a STRIPS model.

4.1 Computing a Conformant Plan for B

The procedure we use to compute the optimal plan for a belief sample relies on a SAT solver and follows the schematic representation of Figure 2. In planning by SAT [Kautz and Selman, 1996], given a horizon k and a planning problem a propositional formula Φ is defined that is satisfiable iff there exists a sequence of actions of length k that solves the planning problem.² Furthermore Φ is defined over $k + 1$ copies of the state variables (the state SAT variables p_0 to p_k where p is a state variable) and k copies of the actions (the action SAT variables a_0 to a_{k-1} where a is an action). Φ is defined such that a solution to the planning problem can be trivially extracted from the satisfying assignment (for instance, if a_i evaluates to *true*, then the i th action of the plan is a). If, for instance, action a sets state variable p to *false*, Φ will be defined such that for all $i \in \{1, \dots, k\}$

$$\Phi \equiv (a_{i-1} \rightarrow \neg p_i) \wedge \dots$$

We refer the reader to the literature on planning by SAT for more details on this reduction.

Given a sample B of b states we create b copies of the state SAT variables: p_i^1, \dots, p_i^b ; the variables p_i^c model the effects of applying the plan on the state $q_\ell \in B$. We stick to a single set of action SAT variables and each copy of the state SAT variables is linked to this

²The value of k is initialized to 0 and incremented until Φ becomes satisfiable.

set. The formula Φ presented in the example above will therefore now translate as

$$\Phi \equiv ((a_{i-1} \rightarrow \neg p_i^1) \wedge \dots \wedge (a_{i-1} \rightarrow \neg p_i^b)) \wedge \dots$$

4.2 Verifying Correctness of a Plan

Like the plan generation, plan correctness is implemented in SAT. This time it matches the representation of Figure 1.

A plan is proved incorrect if an explanation of the observations can be found in which the application of the plan leads to a non final goal (remember that all plans are applicable).

Once again a propositional formula is defined that is satisfiable iff such an explanation exists. This formula contains two parts: SAT variables $p_{i \in \{0, \dots, n\}}$ represent the state of the system in the active mode while variables $p'_{i \in \{0, \dots, k\}}$ represent the state in the repair mode.³ The formula is the conjunction of the formulas:

- Φ_{active} a propositional formula that is satisfiable iff there exists an explanation to the observations (whose final state is represented by the variables p_n); this type of reduction is quite standard [Grastien and Anbulagan, 2013];
- Φ'_{repair} a propositional formula that is satisfiable iff there exists a state in which the proposed plan is not correct (this state is represented by the variables p'_0);
- $\bigwedge_{p \in V} (p_n \leftrightarrow p'_0)$, where p ranges over the state variables, the formula that links the final state of the active phase and the initial state of the repair phase.

Intuitively, the assignments of the variables p_n that are consistent with Φ_{active} are a symbolic representation of \mathcal{B}^O . Formally let \mathcal{V} be the set of variables that appear in Φ_{active} ; then $\exists(\mathcal{V} \setminus \{p_n \mid p \in V\})$. Φ_{active} is logically equivalent to the symbolic representation of \mathcal{B}^O . Similarly the variables p'_0 of Φ'_{repair} represent $\overline{\mathcal{B}^\pi}$.

As a consequence any other representation of \mathcal{B}^O or $\overline{\mathcal{B}^\pi}$ could be used if such representations are more convenient (e.g., if they are more compact or if they help the SAT solver).

Difference Between the Two Reductions

The first reduction aims at finding a plan of length k that is applicable in b states. Therefore it includes $b \times k$ copies of the state variables and k copies of the action variables.

The second reduction aims at finding a plan composed of two parts: a trajectory in the active space and a trajectory in the repair space. Therefore it includes $n + k$ copies of the state variables and n copies of the events (there could be k copies of the actions but the value of these variables is known in advance since the plan is an input of this reduction).

An interesting difference between the two reductions is that the trajectories of the former should lead to goal states while the trajectory of the latter should lead to a *non* goal state. As a consequence when the repair

³It is assumed that the length of the explanation can be bounded by a known value n ; k is the length of the plan being tested.

plan is finally computed the conformant planning reduction to SAT is satisfiable while the reduction of the applicability function is not.

5 Experiments

We ran some experimental evaluation of the approach presented in this paper.

Since the problem presented here is new, we had to build new benchmarks. We propose a variant of the benchmark presented by Grastien *et al.* [2007] which will be made available to the community. The system comprises 20 components interconnected in a torus shape. Each component contains eight states, including two unstable states and one goal state. The behaviour on each component can affect its neighbour and the local observations cannot allow to determine anything about the local behaviour: the full system needs to be monitored in order to understand the state system. Repair actions can also be local or affect several components.

We built 100 problem instances on this system. We restricted ourselves to totally ordered observations, but notice that one of the benefits of using diagnostic techniques is to be able to handle partially-ordered observations (observations where the order of the observed events is only partially known because the delay between their reception is small compared to the transmission/processing delay).

We compare our approach to a symbolic approach that uses BDDs (specifically the `buddy` package) to track the belief state and then uses A* to find the optimal repair plan. The heuristic used by A* is implemented as follows: a state of the system is extracted from the BDD and the optimal repair is computed for this state using SAT; the length of this optimal repair is used as a lower bound for the optimal repair from the current search node.

Our belief sample method uses `glucose_static 4.0` [Audemard and Simon, 2009]. `glucose` is heavily based on the `minisat` solver [Eén and Sörensson, 2003].

The experiments were run on 4-core 2.5GHz cpu with 4GB RAM, with GNU/Linux Mint 16 “petra”. A ten minutes (600s) timeout was provided.

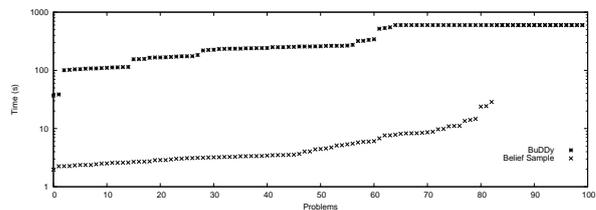


Figure 4: Runtime in seconds required to solve self-healing problem instances; sorted.

The results are summarized in Figure 4. The instances are sorted in increasing runtime, meaning that the instance at position x for one implementation may be different from the instance at the same position for the other. The approach based on the generation of the belief state only saw 64 instances solved before timeout, against 83 for our approach. In general our approach

is two orders of magnitude faster than A^* , although we would need more benchmarks and comparisons to understand better the strength of this approach.

Out of the 87 instances instances solved by the Belief Sample method, 82 could be solved by exhibiting only one element of the belief state. Another three instances could be solved with a sample of two elements, and two required a sample of three elements to generate a conformant plan.

6 Discussion

The objective of connecting the diagnostic and planning tasks is quite ambitious. From the diagnostic perspective, and since the seminal work from Sampath *et al.* [1995] the problem has generally been the detection of specific events or patterns of events [Jéron *et al.*, 2006]. The main inspiration of the present work is the self-heability question asked by Cordier *et al.* [2007]; the aforementioned work is one of the first attempt to frame diagnosis as the problem of finding the optimal repair plan, although the complexity of computing the plan is not addressed. In static contexts similar questions have been asked where the problem was framed as finding the optimal balance between increasing the cost of gathering information (observations) and improving the precision of diagnosis (and, consequently, reducing the cost of planning) [Torta *et al.*, 2008].

Supervisory control [Ramadge and Wonham, 1989] is a problem very similar to self-healing. The goal is to control some actions (forbid their occurrence) in order to meet some specification. The main difference with our work is the fact that control applies continuously while we assume that self-healing is performed when the system is not active (either because the repair process is expensive—it might require to stop the system for instance—or because it can only be performed at some time—every night for instance). Furthermore control tries to be as unobtrusive as possible: it merely forbids some transitions and generally does not choose actions to perform.

Conformant planning [Smith and Weld, 1998] is the problem of finding a sequence of actions that is guaranteed to lead to the specified goal, despite uncertainty on the initial state and nondeterministic action effects. Solutions to conformant planning have been proposed that compute the belief state and run heuristic search [Bonet and Geffner, 2000] or that represent the belief state symbolically [Cimatti and Roveri, 2000]. More similar to our work Hoffmann and Brafman [2006] proposed Conformant-FF in which the belief state is represented implicitly by the set of initial states and the sequence of actions leading to the current state; at every time step, a SAT solver is used to determine the state variable values that can be inferred with certainty. This approach is similar to ours in the way it avoids computing belief states. More generally, we would like to adapt our method to solve conformant planning problems.

The combination of planning and diagnosis has also been studied in the context of *plan repair*. There, a (possibly conformant) plan is computed that assumes that contingencies are unlikely to happen. The plan execution is then monitored and if the outcome of execution does not match the predictions, a new plan is

generated [Micalizio, 2014].

7 Conclusion and Extensions

In this paper we presented a method to solve the self-healing problem. The problem consists in finding a repair plan that can lead back to a goal state a system whose execution has been partially observed. We avoid computing the belief state. Instead we propose a method whereby plans are computed on a sample of the belief state whilst a diagnoser verifies their correctness and generates an element of the belief state (added to the sample) if the plan is not correct. Both the planning and the diagnosis problems are reduced to SAT problems. We show that non trivial problems can be easily solved by this approach.

There are many possible extensions to this work. One issue is that enforcing a conformant plan may be too restrictive. We want to avoid prohibitive repairs in situations where the system is healthy. This is a common problem in diagnosis of dynamic systems: the state of the system can never be precisely determined at the current time; it is often not unconceivable that a fault just happened on the system and has not had time to develop into a visible faulty trace. The issue here is that conformant plans must provide for such contingencies even when there is no evidence for them. An implicit assumption of our work is that unhealthy system behaviours can be detected to a large extend. The set of unstable states serves this purpose: they are useful to model the fact that any “failure” in the system will lead to abnormal observations before a repair action is performed.

We see two avenues to handle situations where the unstability feature cannot address the problem presented before. First probabilities can be incorporated into the model, which allows for chance-constrained planning [Santana and Williams, 2014]. Issues with this approach include the problem of building large models with meaningful probabilities and the problem of extending the SAT reduction to deal with probabilities (as well as scaling up to large models). A second, qualitative, possibility is to ignore contingencies that are supported by no strong evidence. For instance failures that are not part of a minimal diagnosis might be ignored.

Another restriction of the current approach is that the goal G is assumed to be known explicitly. Specification of goal states may however be more complex: Ciré and Botea [2008] have proposed to define goals as properties of states defined in linear temporal logic (LTL). Other relevant goal properties is diagnosability [Sampath *et al.*, 1995], i.e, the property that the observations on the system will allow to detect/identify the important system failures. A related issue is the incremental aspect: how to handle a repair after an active period following a first repair. A simple solution is to assume that the initial state after the repair is the goal state.

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the

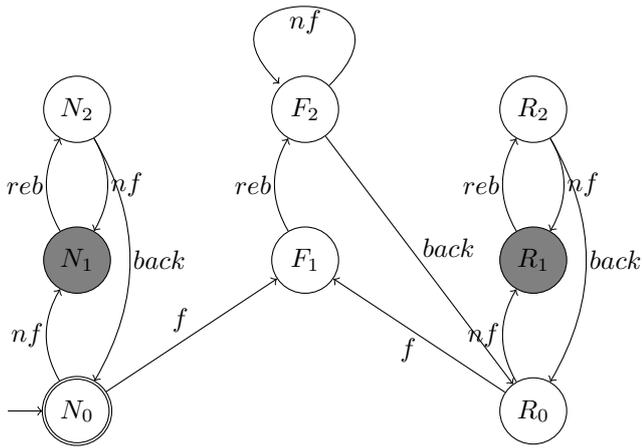


Figure 5: Active model for one component (observable events are *reb* and *back*).

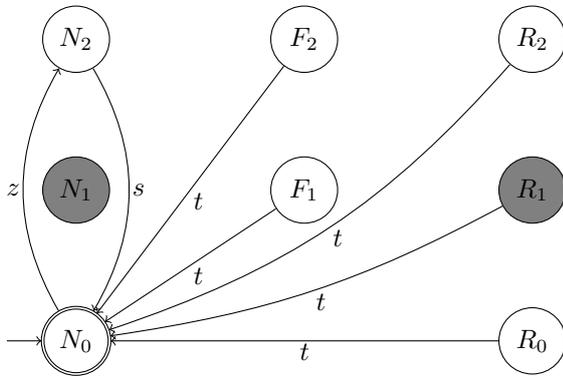


Figure 6: Repair model for one component (no transition means that the state is not affected by action).

Australian Research Council through the ICT Centre of Excellence Program.

A Problem Benchmark

We now present the system we used in the experiments.⁴

The system includes 20 components $c_{i,j}$ where i ranges between 0 and 3 and j between 0 and 4. The component $c_{i,j}$ is connected to $c_{i',j'}$ iff the total different $|i - i'| + |j - j'|$ is at most one (where i and j are taken modulo 3 and 4). For instance, $c_{0,1}$ is connected to four components $c_{0,0}$, $c_{0,2}$, $c_{3,1}$, and $c_{1,1}$.

The model of one component for the active mode is given in Figure 5 and the model for the repair mode is given in Figure 6. The connections between components implies forced transitions when some events occur; these are summarised in Table 1. For instance, when event f occurs on component $c_{0,1}$, event nf occurs on every one of its four neighbours.

A component state contains two types of information: whether a failure occurred on the component and whether it is run. The first part of the state is initially

event/action	neighbour event/action
f	nf
t	z

Table 1: Synchronised events

N (no fault); it moves to F when a fault occurs and R when it recovers. The second part of the state is generally 0 (the component is running) and moves to 1 when it needs to reboot and to 2 when it is rebooting. A fault on a component forces its neighbours to reboot. One difficulty of diagnosis for this type of system is that the observations (*reb* and *back*) do not point precisely to the faulty component.

The repair consists in returning to state N_0 . Most states require action t to return to state N_0 but this action can move the neighbours of the component to state N_2 . Therefore finding the optimal repair requires to order the actions carefully.

References

- [Audemard and Simon, 2009] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solver. In *21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.
- [Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Fifth International Conference on AI Planning and Scheduling (AIPS-00)*, pages 52–61, 2000.
- [Cassandras and Lafortune, 1999] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.
- [Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research (JAIR)*, 13:305–338, 2000.
- [Ciré and Botea, 2008] A. Ciré and A. Botea. Learning in planning with temporally extended goals and uncontrollable events. In *Eighteenth European Conference on Artificial Intelligence (ECAI-08)*, pages 578–582, 2008.
- [Cordier et al., 2007] M.-O. Cordier, Y. Pencolé, L. Travé-Massuyès, and T. Vidal. Self-healability = diagnosability + repairability. In *Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, pages 251–258, 2007.
- [Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *Sixth Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, pages 333–336, 2003.
- [Grastien and Anbulagan, 2013] A. Grastien and A. Anbulagan. Diagnosis of discrete event systems using satisfiability algorithms: a theoretical and empirical study. *IEEE Transactions on Automatic Control (TAC)*, 58(12):3070–3083, 2013.
- [Grastien et al., 2007] A. Grastien, A. Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In

⁴The benchmark is available at this address: <http://www.grastien.net/ban/data/bench-dx15.tar.gz>.

22nd Conference on Artificial Intelligence (AAAI-07), pages 305–310, 2007.

- [Hoffmann and Brafman, 2006] J. Hoffmann and R. Brafman. Conformant planning via heuristic forward search: a new approach. *Artificial Intelligence (AIJ)*, 170:507–541, 2006.
- [Jéron *et al.*, 2006] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete-event systems diagnosis. In *Seventeenth International Workshop on Principles of Diagnosis (DX-06)*, pages 117–124, 2006.
- [Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope : planning, propositional logic, and stochastic search. In *Thirteenth Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, 1996.
- [Lamperti and Zanella, 2003] G. Lamperti and M. Zanella. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.
- [Micalizio, 2014] R. Micalizio. Plan repair driven by model-based agent diagnosis. *Intelligenza Artificiale*, 8(1):71–85, 2014.
- [Ramadge and Wonham, 1989] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE: special issue on Dynamics of Discrete Event Systems*, 77(1):81–98, 1989.
- [Sampath *et al.*, 1995] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control (TAC)*, 40(9):1555–1575, 1995.
- [Santana and Williams, 2014] P. Santana and B. Williams. Chance-constrained consistency for probabilistic temporal plan networks. In *24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.
- [Smith and Weld, 1998] D. Smith and D. Weld. Conformant graphplan. In *Fifteenth Conference on Artificial Intelligence (AAAI-98)*, pages 889–896, 1998.
- [Torta *et al.*, 2008] G. Torta, D. Theseider Dupré, and L. Anselma. Hypothesis discrimination with abstractions based on observation and action costs. In *Nineteenth International Workshop on Principles of Diagnosis (DX-08)*, pages 189–196, 2008.

Implementing Troubleshooting with Batch Repair

Roni Stern¹ and Meir Kalech¹ and Hilla Shinitzky¹

¹Ben Gurion University of the Negev

e-mail: roni.stern@gmail.com, kalech@bgu.ac.il, hillash@post.bgu.ac.il

Abstract

Recent work has raised the challenge of efficient automated troubleshooting in domains where repairing a set of components in a single repair action is cheaper than repairing each of them separately. This corresponds to cases where there is a non-negligible overhead to initiating a repair action and to testing the system after a repair action. In this work we propose several algorithms for choosing which batch of components to repair, so as to minimize the overall repair costs. Experimentally, we show the benefit of these algorithms over repairing components one at a time (and not as a batch).

1 Introduction

Troubleshooting algorithms, in general, plan a sequence of actions that are intended to fix an abnormally behaving system. Fixing a system includes repairing faulty components. Such repair actions incur a cost. These costs can be partitioned into two types of repair cost. The first, referred to as the *component repair cost*, is the cost of repairing a component. The second, referred to as the *repair overhead*, is the cost of preparing the system to perform repair actions (e.g., halting the system may be required), and the cost of testing the system after performing a repair action.

This paper considers the case where the repair overhead is not negligible and is potentially more expensive than a component repair cost (of a single component). Therefore, it may be more efficient to repair a batch of components in a single repair action. We call the problem of choosing which batch of components to repair the Batch Repair Problem (BRP). BRP is an optimization problem, where the task is to minimize the *total repair costs*, which is the sum of the repair overheads and component repair costs incurred by all the repair actions performed until the system is fixed.

Note that in this paper we use the term “repair” for a single or a set of components and the term “fix” to refer to the entire system. Thus, *repairing* components eventually causes the system to be *fixed*, and a system is only fixed if it returned to its nominal behavior.

Most previous work assumed that components are repaired one at a time [1; 2; 3; 4]. This approach can be wasteful for BRP. For example, if a diagnosis engine infers that multiple faulty components need to be repaired to fix the system, then it would be wasteful to repair these components one at a time since each repair action incurring its

repair overhead. Instead, an efficient BRP algorithm would repair all the faulty components in a single repair action. More generally, we expect an intelligent BRP algorithm to weigh the cost of repairing batches of components as well as the repair overhead. Some discussion on repairing multiple components together was done in prior work on self healability [5].

Due to the repair overhead, repairing a single component, even if it is the component most likely to be faulty, can be wasteful. This is especially wasteful in cases where all the found diagnoses consists of multiple faulty components, thus suggesting that repairing a single component would not fix the problem. Alternatively, one may choose to repair the components in the most likely diagnoses. This may also be wasteful, especially if there are several diagnoses which have similar likelihood. It might be worthwhile to repair by a single repair action a set of components that “covers” more than a single diagnosis. This may reduce the number of repair actions until the system is fixed, thus saving repair overhead costs. The downside in this approach is that the component repair costs can be high, as more healthy components may be repaired.

For example, consider the small system described in Figure 1. It is a logical circuit whose output is fault. Assume that the “OR” gate is known to be healthy and there are only two possible diagnoses: either *A* is faulty or *B* is faulty, where the probability that

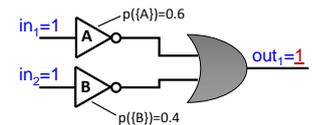


Figure 1: An example where repairing components one at a time is wasteful.

There are three possible repair actions: to repair *A*, to repair *B*, and to repair *A* and *B*. Assume the repair overhead costs 10, and repairing a component costs 1. If *A* is repaired, there is a 0.4 chance that the system would not be fixed and another repair action would be needed (repairing *B*). Thus, the expected total repair cost of repairing *A* first is 15.4. Similarly, the total repair cost for repairing *B* first is 17.6. The best option is thus to repair *A* and *B* together in a single repair action, incurring a total repair cost of 12.

Recent work [6] proposed two high-level approaches to solve BRP: as a planning under uncertainty problem, or as a combinatorial optimization problem. When modeling BRP as a planning under uncertainty problem the task is to find a

repair policy, mapping a state of the system to the repair action that minimizes the expected total repair costs. This approach, while attractive theoretically, quickly becomes not feasible in non-trivial scenarios.

In this work we focus on the second high-level approach proposed for BRP, in which BRP is modeled as a combinatorial optimization problem, searching in the combinatorial space of possible repair actions for the best repair action. There are two challenges in implementing this approach. First, how to measure the quality of a repair action and how to efficiently search for the repair action that maximizes this measure. There are many efficient heuristic search algorithms in the literature, and thus the main challenge addressed in this work is in proposing several heuristics for estimating the merit of a repair action.

The contributions of this work are practical. A range of heuristic objective functions are proposed and analyzed, and we evaluate their effectiveness experimentally on a standard benchmark. A clear observation from the results is that indeed considering batch repair actions can save repair cost significantly. Moreover, the most effective heuristics provide a tunable tradeoff between computation time and resulting repair costs.

2 Problem Definition

A classical MBD input $\langle SD, COMPS, OBS \rangle$ is assumed, where SD is a model of the system, $COMPS$ represents the components in the system, and OBS is the observed behavior of the system. Every component can be either normal or abnormal. The assumption that a component $c \in COMPS$ is abnormal is represented by the abnormal predicate $AB(c)$.

A *batch repair problem* (BRP) arises when the assumption that all components are normal is not consistent with the system description and observations. Formally,

$$SD \wedge OBS \wedge \bigwedge_{c \in COMPS} \neg AB(c) \text{ is not consistent}$$

In such a case, at least one component must be repaired.

Definition 1 (Repair Action). *A repair action can be applied to any subset of components and results in these components becoming normal. Applying a repair action to a set of components γ is denoted by $Repair(\gamma)$.*

Definition 1 assumes that repair actions always succeed, i.e., a component is normal after it is repaired.

After a repair action, the system is tested to check if it has been fixed. We assume that the system inputs in this test are the same as in the original observations (OBS). The observed system outputs are then compared to the expected system outputs of a healthy system. Thus, the result of a repair action is either that the system is fixed, or a new observation that may help choosing future repair actions.

Repairing a set of components incurs a cost, composed of a repair overhead and component repair costs. The repair overhead is denoted by $cost_{repair}$, and the component repair cost of a component $c \in COMPS$ is denoted by $cost_c$.

Definition 2 (Repair Costs). *Given a set of components $\gamma \subseteq COMPS$, applying a repair action $Repair(\gamma)$ incurs a cost:*

$$cost(Repair(\gamma)) = cost_{repair} + \sum_{c \in \gamma} cost_c$$

We assume that all repair costs are positive and non-zero, i.e., $cost_{repair} > 0$ and $cost_c > 0$ for every component $c \in COMPS$. As defined earlier, the task in BRP is to fix a system with minimum total repair cost.

As shown in Figure 1, an efficient BRP solver should consider the possibility of repairing a set of components in a single repair action. Thus, the potential number of repair actions is $2^{|COMPS|}$. Therefore, from a complexity point of view BRP is an extremely hard problem.

3 Preliminaries

Next, we provide background and definitions required for describing the BRP algorithms we propose.

SD describes the behavior of the diagnosed system, and in particular the behavior of each component. The term *behavior mode* of a component refers to a state of the component that affects its behavior. SD describes for every component one or more behavior modes. For every component, at least one of the behavior modes must represent the nominal behavior of the component.

A mode assignment ω is an assignment of *behavior modes* to components. Let $\omega^{(+)}$ be the set of components assigned a nominal (i.e., normal) behavior mode and $\omega^{(-)}$ be the set of components assigned one of the other modes.

Definition 3 (Diagnosis). *A mode assignment ω is called a diagnosis if $\omega \wedge OBS \wedge SD$ is satisfiable.*

A model-based diagnosis engine (MBDE) accepts as input SD , OBS , and $COMPS$ and outputs a set of diagnoses Ω . Although a diagnosis is consistent with SD and OBS , it may be incorrect. A diagnosis ω is *correct* if by repairing the set of components in $\omega^{(-)}$ the system is fixed. Some diagnosis algorithms return, in addition to Ω , a measure of the likelihood that each diagnosis is *correct* [7; 8]. Let $p : \Omega \rightarrow [0, 1]$ denote this likelihood measure. We assume that $p(\omega)$ is normalized so that $\sum_{\omega \in \Omega} p(\omega) = 1$ and use it to approximate the probability that ω is correct.

A common way to estimate the likelihood of diagnoses, assumes that each component has a prior on the likelihood that it would fail and component failures are independent. Therefore, if $p(c)$ represents the likelihood that a component c would fail then diagnosis likelihood can be computed as

$$p(\omega) = \frac{\prod_{c \in \omega^-} p(c)}{\sum_{\omega' \in \Omega} \prod_{c \in \omega'^-} p(c)} \quad (1)$$

where the denominator is a normalizing factor. We assume in the rest of this paper that diagnoses likelihoods are computed according to Equation 1. Other methods for computing likelihood of diagnoses also exist [9].

3.1 System Repair Likelihood

If the MBDE returns a single diagnosis ω that is guaranteed to be correct, then the optimal solution to BRP would be to perform a single repair action: $Repair(\omega^-)$. This, however, is rarely the case, and more often a possibly a very large set of diagnoses is returned by diagnosis algorithms. This introduces uncertainty as to whether a repair action would actually fix the system. We define this uncertainty as follows:

Definition 4 (System Repair Likelihood). *The System Repair Likelihood of a set of components $\gamma \subseteq COMPS$, denoted $SystemRepair(\gamma)$, is the probability that $Repair(\gamma)$ would fix the system.*

Consider the relation between $p(\omega)$ and $\text{SystemRepair}(\omega)$. If ω is correct, then repairing all components that are faulty, meaning $\omega^{(-)}$, would fix the system. Therefore, the likelihood of repairing $\omega^{(-)}$ causing the system to be fixed is at least $p(\omega)$, i.e.,

$$\text{SystemRepair}(\omega^{(-)}) \geq p(\omega)$$

Moreover, if ω is correct then repairing any superset of $\omega^{(-)}$ would also fix the system. Thus, $\text{SystemRepair}(\omega^{(-)})$ may be larger than $p(\omega)$. On the other hand, repairing any set of components that is not a superset of $\omega^{(-)}$, as there would still be faulty components in the system. Therefore, a repair action $\text{Repair}(\text{COMPSS}')$ would fix the system if and only if $\omega^{*(-)} \subseteq \text{COMPSS}'$, where ω^* is the correct diagnosis. While we do not know ω^* , we can compute $\text{SystemRepair}(\gamma)$ from Ω and $p(\cdot)$:

$$\text{SystemRepair}(\gamma) = \sum_{\omega \in \Omega \wedge \omega \subseteq \gamma} p(\omega)$$

For example, in the logical circuit depicted in Figure 1, there are two diagnoses, $\{A\}$ and $\{B\}$, such that $p(\{A\}) = 0.6$ and $p(\{B\}) = 0.4$. Thus, $\text{SystemRepair}(\{A\})=0.6$, $\text{SystemRepair}(\{B\})=0.4$, and $\text{SystemRepair}(\{A, B\})=p(\{A\})+p(\{B\})=1$.

4 BRP as a Combinatorial Search Problem

As mentioned in the introduction, the approach for solving BRP that we pursue in this paper formulates BRP as a combinatorial search problem. The search space is the space of possible repair actions, i.e., every subset of the set of components there were not repaired yet. The search problem is to find the repair action that maximizes a utility evaluation function $u(\cdot)$ that maps a repair action to a real value that estimates its merit.

The effectiveness of this search-based approach for BRP depends on the search algorithm used and how the $u(\cdot)$ utility function is defined. There are many existing heuristic search algorithm for searching large combinatorial search spaces [10; 11]. Thus, in this work we propose and evaluate a set of possible utility functions. Note that for some of the utility functions described next it is possible to find the best repair action without searching the entire search space of possible actions, while others are more computationally intensive.

4.1 k Highest Probability

A key source of information for all the utility functions described below is the set of diagnoses Ω and their likelihoods ($p(\cdot)$). We assume that this information is obtained by using a diagnosis engine over the observations of the current state of the system. The set of returned diagnoses may be very large. The first utility function we propose is based on the system's *health state*, which has been recently proposed as a method for aggregating information from a set of diagnoses [12].

Definition 5 (Health State). *A health state is a mapping $F : \text{COMPSS} \rightarrow [0, 1]$ where*

$$F(c) = \sum_{\omega \in \Omega \text{ s.t. } c \in \omega} p(\omega)$$

$F(c)$ is an estimate of the likelihood that component c is faulty given a set of diagnoses Ω and their likelihoods. Based on the system's health state, we propose the following utility function, denoted u_{HP} :

$$u_{HP}(\gamma) = \sum_{c \in \gamma} F(c)$$

where γ is any subset of COMPSS that has not been repaired yet.

The repair action that maximizes u_{HP} is trivial — repair all components. This would result in the system being repaired, but of course, may repair many components that are likely to be healthy. To mitigate this effect, we propose the *k highest probability* repair algorithm (k -HP), which limits the number of components that can be repaired in a single repair action to k , where k is a user-defined parameter. Note that computing k -HP does not need any exhaustive search: simply sort the health state in descending order of $F(\cdot)$ values and repair the first k components.

The k -HP repair algorithm has two clear disadvantages. First, the user needs to define k . Second, k -HP does not consider repair costs (neither component repair costs nor overhead costs). The next set of utility functions and corresponding repair algorithms address these disadvantages.

4.2 Wasted Costs Utilities

Before describing the next set of proposed utility functions we explain the over-arching reasoning behind it. Repairing a system requires performing repair actions. Some repair costs are inevitable. These are the repair overhead of a single repair action, and the component repair costs that repair the faulty components. We propose a family of utility functions that try to estimate the expected total repair costs beyond these inevitable costs. We refer to these costs as *wasted costs* and to utility functions of this family as *wasted cost functions*.

We model these wasted costs as being composed of two parts.

- **False positive costs** (cost_{FP}). These are the costs incurred by repairing components that are not really faulty.
- **False negative costs** (cost_{FN}). These are the overhead costs incurred by future repair actions.

It is clear why the false positive costs are wasted costs — these are repair costs incurred on repairing healthy components. The false negative costs are wasted costs because if one knew upfront which components are faulty, then the optimal repair algorithm would repair all these components in a single batch repair action, incurring no further overhead costs. Thus, future overhead costs represent wasted costs.

We borrow the terminology of false positive and false negative from the machine learning literature, but use it in a somewhat different manner. To explain this choice of terminology, assume that positive and negative mean faulty and healthy components respectively. Choosing to repair a faulty component is regarded as a true positive, and not repairing a healthy component is regarded as a true negative. Thus, the wasted costs incurred by repairing healthy components are costs incurred due to false positives, and the wasted costs incurred by not repairing a faulty component are costs incurred due to false negatives. While this is not a perfect match in terminology, we believe that it helps clarify the underlying intention of cost_{FP} and cost_{FN} .

The Wasted Cost Utility Function

For a given set of components γ , we denote by $cost_{FP}(\gamma)$ and $cost_{FN}(\gamma)$ the fast positive costs and false negative costs, respectively, incurred by performing a batch repair action of repairing all the components in γ . Given $cost_{FP}(\gamma)$ and $cost_{FN}(\gamma)$, we propose the following general formula for computing the expected wastes costs, denoted by C_{WC} .

$$cost_{FP}(\gamma) + (1 - \text{SystemRepair}(\gamma)) \cdot cost_{FN}(\gamma)$$

The left hand side of the formula is the false positive costs. The right hand side of the formula is the false negative costs, multiplied by the probability that the system will not be fixed by repairing the components in γ . Thus, the formula gives the total expected wastes costs. We define $U_{WC} = -C_{WC}$ as the *wasted cost utility function*.

The wasted cost utility function is a theoretical utility function, since one does not know upfront the values of $cost_{FP}$ and $cost_{FN}$. Next, we propose several ways to estimate u_{WC} by proposing ways to estimate $cost_{FP}$ and $cost_{FN}$.

Estimating the False Positives Cost

We propose to estimate the false positive costs by considering the system's health state (Definition 5), as follows.

$$\widehat{cost}_{FP}(\gamma) = \sum_{c \in \gamma} (1 - F(c)) \cdot cost(C_i)$$

This estimate of the false positive costs can be understood as an expectation over the false positive costs. The cost of a repaired component $c \in \gamma$ is part of the false positive costs only if c is in fact healthy. The probability of this occurring is $(1 - F(c))$. Thus, $(1 - F(c)) \cdot cost(c)$ is the expected false positive cost due to repairing component c .

False Negatives Cost

Correctly estimating $cost_{FN}$ is more problematic than $cost_{FP}$, as it requires considering the future actions of the repair algorithm. In the best case, only one additional repair action would be needed. This would incur a single additional overhead cost. We call this the optimistic $cost_{FN}$, or simply $cost_{FN}^o$, which is equal to $cost_{repair}$. The other extreme assumes that every component not repaired so far would be repaired by a single repair action, and correspondingly an incurred overhead cost. We experimented with a slightly less extreme estimate, in which we assume that only faulty component will be repaired in the future, but each will be repaired in a single repair action, incurring one $cost_{repair}$ per faulty component. Since we do not know the number of faulty components, we use the expected number of faulty components according to the health state: $\sum_{c \notin \gamma} F(c)$. The resulting estimate is referred to as the pessimistic estimate of $cost_{FN}$, denoted by $cost_{FN}^p$, is thus computed as:

$$cost_{FN}^p(\gamma) = cost_{repair} \cdot \sum_{c \notin \gamma} F(c)$$

Summarizing all the above, we propose two utility functions from the wasted cost utility function family. A pessimistic wasted cost function, that uses \widehat{cost}_{FP} and $cost_{FN}^p$ to estimate $cost_{FP}$ and \widehat{cost}_{FN} , and an optimistic wasted cost function that uses \widehat{cost}_{FP} and $cost_{FN}^o$. The corresponding repair algorithms search in the combinatorial space of all possible sets of components to find the set of components that maximizes u_{WC} .

4.3 Handling the Computational Complexity

The search space is very large — the size of the power set of all components that were not repaired so far. We explored two simple ways to handle this. The first approach is to only consider subset of components with up to k components, where k is a parameter. This approach is referred to as *Powerset-based search*.

The second approach we considered is to consider only supersets of the diagnoses in Ω . This has the intuitive reasoning that at least one of these diagnoses is supposed to be true (according to the known observation), and thus a repair algorithm should try to aim for fixing the problem in the next repair action. Thus, in this approach, we considered in the search for the best repair action every set of components that are unions of at most k diagnoses, where k is a parameter. This approach is referred to as the *Union-based search*.

For both powerset-based search and union-based search, increasing k results in a larger search space. This means higher computational complexity, but also increases the range of repair actions considered, and thus using higher k can potentially find better repair actions than using lower k values. This provides an often desired tradeoff of computation vs. solution quality. Experimentally, we observed that the union-based search approach yields much better results and thus we only show results for it in the experimental results below.

5 Experimental Results

We evaluated the proposed batch selection algorithms on two standard Boolean circuits: 74283 and 74182. We experimented on 21 observations for system 74283 and 23 observations for system 74182. These observations were selected randomly from Feldman et al.'s [13] set of observations. For each observation, all subset minimal diagnoses were found using exhaustive search.

5.1 Baseline Repair Algorithms

The main hypothesis of this line of work is that performing a batch repair action can save repair costs. To evaluate if the proposed batch repair algorithms are able to do so, we compare them with two repair algorithms that do not consider batch repair actions. These baseline repair algorithms, named “Best Diagnosis” (BD) and “Highest Probability” (HP), are inspired by previous work on test planning [14] and work as follows. BD chooses to repair a single component from the most preferred diagnosis in Ω (that with the highest $p(\cdot)$ value). From the set of components in the most probable diagnosis, BD chooses to repair the one with the lowest repair costs. The HP repair algorithm chooses to repair the component that is most likely to be faulty, as computed by the system's health state ($F[\cdot]$).

Another baseline repair algorithm we evaluated experimentally that serves as a baseline is to repair all components of the most likely diagnosis in a single batch repair action. Note that this algorithm, denoted *Batch Best Diagnosis*, ignores repair costs, and serves as an extreme alternative to the BD algorithm that repairs a single component from the most likely diagnosis.

Table 1 shows the average repair costs incurred until the system was fixed for the proposed repair algorithms. The average was over all the observations we used for system 74182. The rows labeled BD, HP, 2-HP, and 3-HP show the

Algorithm	Overhead cost			
	10	15	20	25
BD & HP	83.5	111.3	139.1	167.0
2-HP	61.5	77.8	94.1	110.4
3-HP	53.0	65.0	77.0	88.9
Opt.(1)	55.2	68.9	82.6	96.3
Opt.(2)	53.0	65.0	75.2	86.7
Opt.(3)	55.2	66.5	72.6	83.7
Pes.(1)	55.0	68.9	81.3	96.1
Pes.(2)	52.8	59.8	63.7	70.0
Pes.(3)	49.6	50.4	55.9	64.6

Table 1: Average repair costs for the 74182 system.

Algorithm	Overhead cost			
	10	15	20	25
BD	116.4	155.2	194.0	232.9
HP	109.3	145.7	182.1	218.6
2-HP	81.2	102.1	123.1	144.0
3-HP	70.5	85.7	101.0	116.2
Opt.(1)	76.0	95.7	115.2	134.8
Opt.(2)	72.9	89.8	102.4	111.7
Pes.(1)	75.2	95.7	114.0	134.8
Pes.(2)	72.4	84.8	93.6	96.0

Table 2: Average repair costs for the 74283 system.

results for the BD, HP, and k -HP repair algorithms (for $k=2$ and 3). The rows Opt.(1), Opt.(3), and Opt.(3) show the results for the union-based search repair algorithm using the wasted cost utility function with $cost_{FP}$ to estimate $cost_{FP}$ and $cost_{FN}^o$ to estimate $cost_{FN}$. The rows Pes.(1), Pes.(2), and Pes.(3) show results for the same configuration, except for using $cost_{FN}^p$ to estimate $cost_{FN}$ instead of $cost_{FN}^o$. The repair costs of a single component was arbitrary set to 5 and the cost of the overhead ($cost_{repair}$) was varied (10,15,20,25). Each column represents results for different values of $cost_{repair}$. In this domain, the results of HP and BD were virtually the same, and thus we grouped them to a single row.

The results clearly show the benefit of considering batch repair actions. The best performing repair algorithm is Pes.(3), which required more than half the repair costs needed for BD and HP, which do not consider batch repair. This supports the main hypothesis of this paper: batch repair actions can save significant amount of repair costs. As expected, the gain of batch repair actions increases as the repair overhead ($cost_{repair}$) increases. Also note that for Pes.(k) we observe the desired trend of increasing k resulting in lower repair costs. This is also observed for the k -HP repair algorithm (note that the HP algorithm is in fact 1-HP), but is not always the case for Opt.(k), where for lower overhead cost $k = 2$ yielded lower repair costs than $k = 3$. This suggests that the optimistic estimate of $cost_{FN}$ is not robust. Computationally, increasing k required much more runtime, and we could not run experiments with $k = 4$ on our current machines in reasonable time. Table 2 shows the results for the 74283 system. The trends observed are the same as those discussed above for the results of 74182 system.

6 Related Work

BRP is a troubleshooting problem, where the goal is to perform repair actions so as to fix a system. Algorithms for au-

tomated troubleshooting were proposed in previous works. Heckerman et al. [1] proposed the *decision-theoretic troubleshooting* (DTT) algorithm, that uses a decision theoretic approach for deciding which components to observe in order to identify the faulty component. Later work also applied a decision theoretic approach that integrated planning and diagnosis to a real world troubleshooting application [3; 15]. Torta et al. [4] proposed using model abstractions for troubleshooting while taking into account the cost of repair actions. All these works did not consider the possibility of repairing a set of components together, allowing only repair actions that repair a single component at a time.

Our current paper on BRP do not consider applying further diagnostic actions such as probing and testing, which are considered by previous troubleshooting algorithms. Thus, our work on BRP could be integrated in previous troubleshooting frameworks so as to consider both batch repair actions and diagnostic actions. This is left to future work.

Friedrich and Nedjl [2] discussed the relation between diagnoses and repair, in an effort to minimize the *breakdown costs*. Breakdown costs roughly correspond to a penalty incurred for every faulty output in the system, for every time step until the system is fixed. In BRP, the goal is to minimize costs until the system is fixed, and there is no partial credit for repairing only some of the system outputs.

7 Conclusion and Future Work

We addressed the problem of troubleshooting with the possibility of performing a batch repair action — a repair action in which more than a single component is repaired. Batch repair makes sense only if repairing a set of components in a single repair action is cheaper than repairing each of them separately. We proposed several algorithms for selecting which batch of components to repair. Experimental results clearly show the benefit of batch repair over single repair actions, and the benefit of the algorithms we suggested for choosing these set of components to repair. Future work will investigate when should batch repair be considered, and how to detect such cases upfront. Additionally, expanding beyond Boolean circuits is also needed, as well as addressing uncertainty on the outcome of repair actions.

References

- [1] David Heckerman, John S Breese, and Koos Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.
- [2] Gerhard Friedrich and Wolfgang Nejdl. Choosing observations and actions in model-based diagnosis/repair systems. *KR*, 92:489–498, 1992.
- [3] Anna Pernestål, Mattias Nyberg, and Håkan Warnquist. Modeling and inference for troubleshooting with interventions applied to a heavy truck auxiliary braking system. *Engineering Applications of Artificial Intelligence*, 25(4):705–719, June 2012.
- [4] Gianluca Torta, Luca Anselma, and Daniele Theseider Dupré. Exploiting abstractions in cost-sensitive abductive problem solving with observations and actions. *AI Commun.*, 27(3):245–262, 2014.
- [5] Marie-Odile Cordier, Yannick Pencolé, Louise Travé-Massuyès, and Thierry Vidal. Self-healability = diag-

- nosability + repairability. In *the International Workshop on Principles of Diagnosis (DX)*, pages 251–258, 2007.
- [6] Roni Stern and Meir Kalech. Repair planning with batch repair. In *International Workshop on Principles of Diagnosis (DX)*, 2014.
 - [7] Brian C Williams and Robert J Ragno. Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics*, 155(12):1562–1595, 2007.
 - [8] Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. Simultaneous debugging of software faults. *Journal of Systems and Software*, 84(4):573–586, 2011.
 - [9] O.J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun. Probabilistic model-based diagnosis: An electrical power system case study. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(5):874–885, 2010.
 - [10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
 - [11] Stefan Edelkamp and Stefan Schroedl. *Heuristic search: theory and applications*. Elsevier, 2011.
 - [12] Roni Stern, Meir Kalech, Shelly Rogov, and Alexander Feldman. How many diagnoses do we need? In *AAAI*, 2015.
 - [13] Alexander Feldman, Gregory Provan, and Arjan van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research (JAIR)*, 38:371, 2010.
 - [14] Tom Zamir, Roni Stern, and Meir Kalech. Using model-based diagnosis to improve software testing. In *AAAI (to appear)*, 2014.
 - [15] Håkan Warnquist, Jonas Kvarnström, and Patrick Doherty. Planning as heuristic search for incremental fault diagnosis and repair. In *Scheduling and Planning Applications Workshop (SPARK) at the International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

Formulating Event-Based Critical Observations in Diagnostic Problems

Cody James Christopher^{1,2} and Alban Grastien^{2,1}

¹Artificial Intelligence Group, The Australian National University.

²Optimisation Research Group, NICTA*

Abstract

We claim that in scenarios involving a human operator with responsibility over systems being monitored by a diagnoser, presenting said operator with a concise set of observations capturing the essence of a failure improves the operator's understanding of the diagnosis.

We take this in the context of Discrete Event Systems and demonstrate how the idea can be applied to systems utilising event-based observations, which can contain implicit information. We introduce the notion of an abstracted event stream, called a sub-observation, that makes the implicit information explicit for the operator and allows a diagnoser to arrive at the same diagnosis. We call the most abstract of these the critical observation. We provide relevant definitions, properties, and a procedure for computing the critical observation in a diagnosis problem.

1 Introduction

Diagnosis problems are concerned with the detection and identification of occurrences of specific events in a system, generally called faults or failures. These occurrences are difficult to detect as the fault events are typically not directly observable, however, they can be inferred from the system model (a description of the system behaviour) and the observations produced by the system.

Diagnosis is the first step in the fault recovery process. Once a fault has been detected and identified, the appropriate actions can be taken to mitigate its effects. The issue, however, is that this procedure acts as a black box; given a model and a sequence of observations, a diagnoser asserts a fault by claiming that there is no possible nominal execution of the system that would produce the observation sequence.

The present work is written under the assumption that a diagnosis procedure is fundamentally built for a human operator in charge of taking actions after a fault is identified. In this scenario, a black box approach does not allow for the presentation of the information relevant to the diagnosis. We assume that providing the operators with explanatory evidence is useful in convincing them of the validity of

the diagnosis, in addition to providing information as to the causes of the fault.

Further, we assume that a more concise explanation is strictly preferred to more verbose explanation, and consequently that there is merit to isolating the “smallest” amount of supporting evidence, or what we call the *critical observations*. In cognitive psychology, the seminal paper on the topic of working memory in humans supports this view, giving the average working memory capacity as 7 ± 2 distinct pieces of information [1]. Providing only the observations critical to the diagnosis also has the additional benefit of ameliorating privacy concerns in systems where privacy is considered important.

We extend the results of Christopher et al. [2] to event-based observations. We first present preliminary theory and notation, before going on to show that event-based observations contain implicit information. We then introduce what we call *sub-observations* that can capture this implicit information and make it available for use in diagnosis procedures. We then provide formal definitions of *sufficiency* and *criticality* in addition to several important properties that allow for a terminating algorithm. We present an algorithm for computing the critical observation and discuss its complexity. A discussion of alternate ways of defining sub-observations precedes a brief discussion of related work and a conclusion.

2 Preliminaries and Notations

The present work takes place in the context and standard framework of discrete event systems (DES) [3]. We denote as Σ the set of events that can take place on the system. A system *run* is a finite sequence of events, $w = e_1 e_2 \dots e_k$, and the system is modeled as the prefix-closed language $\mathcal{L}_M \subseteq \Sigma^*$ that represents all possible runs.

The set of events is partitioned into *observable* events Σ_o —events that are recorded—and *unobservable* events Σ_u —those that are not. The observation o generated by run $w = e_1 e_2 \dots e_k$, hereafter called the *trace* of w , is the projection of w on the set of observable events (i.e., all unobservable events of the run are deleted):

$$o = P_{\Sigma_o}(w) = \begin{cases} \varepsilon & \text{if } k = 0 \\ e_1 P_{\Sigma_o}(e_2 \dots e_k) & \text{if } k > 0 \text{ and } e_1 \in \Sigma_o \\ P_{\Sigma_o}(e_2 \dots e_k) & \text{otherwise.} \end{cases}$$

The observed language of a trace o , denoted \mathcal{L}_o , is the set of finite sequences of events that could produce the observed sequence: $\mathcal{L}_o = P_{\Sigma_o}^{-1}(o) = \{w \in \Sigma^* \mid P_{\Sigma_o}(w) = o\}$.

*NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

The set of unobservable events includes a subset of fault events, $\Sigma_f \subseteq \Sigma_u$. With slight abuse of notation we write $f \in w$ as short for $w \in \Sigma^* f \Sigma^*$ (or “ f appears in w ”) and $F \cap w$ as short for $\{f \in F \mid f \in w\}$ (or “the subset of events from F that appear in w ”).

A set $\delta \subseteq \Sigma_f$ of faults is *consistent* with the model \mathcal{L}_M and the trace o if there exists a run $w \in \mathcal{L}_M$ that would produce this trace ($P_{\Sigma_o}(w) = o$) and that exhibits exactly these faults ($w \cap \Sigma_f = \delta$). The diagnosis of trace o , denoted $\Delta(o)$, is the collection of all consistent sets of faults:

$$\Delta(o) = \left\{ \delta \subseteq \Sigma_f \mid \begin{array}{l} \exists w \in \mathcal{L}_M. \\ P_{\Sigma_o}(w) = o \wedge \delta = w \cap \Sigma_f \end{array} \right\} \quad (1)$$

Hereafter we use the hat notation ($\hat{\cdot}$) to indicate that the given symbol represents what actually occurred. Given a run \hat{w} , $\hat{\delta} = \hat{w} \cap \Sigma_f$ is the set of faults that occurred during the run; then the following result is trivial: $\hat{w} \in \mathcal{L}_M \Rightarrow \hat{\delta} \in \Delta(P_{\Sigma_o}(\hat{w}))$. (The premise, completeness of the model, is assumed.)

We find it more convenient to define the diagnosis in terms of emptiness of languages. Let \mathcal{L}_δ be the language that represents all sequences that contain exactly δ :

$$\mathcal{L}_\delta = \{w \in \Sigma^* \mid w \cap \Sigma_f = \delta\} = \bigcap_{f \in \delta} \Sigma^* f \Sigma^* \cap \bigcap_{f \in \Sigma_f \setminus \delta} (\Sigma \setminus \{f\})^*$$

That is, \mathcal{L}_δ represents the set of all runs containing all of the faults of δ , intersected with all possible runs where the faults not in δ never occur—the result is a set of all runs where the only faults that occur are those in δ . With \mathcal{L}_δ defined, we can equivalently express the diagnosis as an emptiness of languages problem:

$$\delta \in \Delta(o) \iff \mathcal{L}_M \cap \mathcal{L}_o \cap \mathcal{L}_\delta \neq \emptyset. \quad (2)$$

3 Sub-Observations

We first discuss event-based observations, and in particular that event-based observations contain implicit information that must be taken into consideration when performing diagnosis. We then introduce the notion of *sub-observations*, providing formal definitions and an explanatory example. Once this has been established, a procedure is given for diagnosing with sub-observations.

3.1 Event-Based Diagnosis and Implicit Information

Event-based diagnosis, contrasted with state-based diagnosis, comes with a subtlety; specifically, there is a type of implicit information encoded in the trace. Take for example the repeated observation of a window being closed without there ever being an observation of the window opening; in this case, the fact that we never observed an open event is distinctly relevant to a diagnosis procedure.

To further illustrate this, we provide a simple abstract example in the form of a DES: Take $\Sigma = \{a, b, c, d, e, f_1, f_2\}$, with $\Sigma_o = \{a, b, c, d, e\}$, $\Sigma_u = \Sigma_f = \{f_1, f_2\}$. We provide the system model in the form of a NFA in Figure 1 and consider some example traces over it:

$o_1 = abababc$. The model specifies that f_2 must have occurred in strings containing a followed by c . In this case, the intervening sequence is long ($babab$), and could be much longer. The important information, however, is that a was at some point followed by c . Reporting in some abstract sense

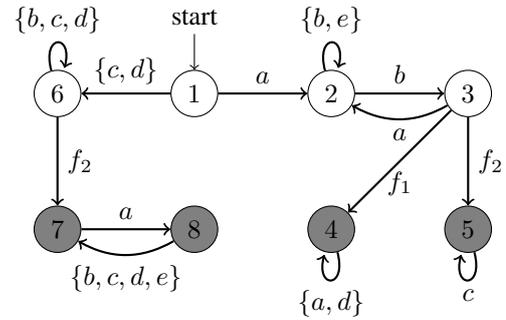


Figure 1: Example DES

that a was followed by c is enough to convince an operator of the correctness of the diagnosis.

$o_2 = ababaa$. The model specifies that f_1 must have occurred for there to be two a events that are not separated by another observable event. More specifically, the *lack* of an intervening event is the crucial piece of information that determines the fault. In this case, reporting in some abstract sense that multiple a occurred consecutively is enough to indicate the fault convincingly.

3.2 Framework

We first present a general framework for sub-observation, which is then further specified for our particular choice of implementation.

General Definition

Definition 1 We define a framework for sub-observations as a tuple: $\langle \mathbb{O}, \preceq, sub \rangle$:

1. A sub-observation, θ , is an abstraction over a trace that represented an intentional relaxation (or weakening) of the concrete knowledge contained in the trace.
2. \mathbb{O} is the space of possible sub-observations.
3. The symbol \preceq is a binary relation and partial order over \mathbb{O} and relates two sub-observations θ, θ' such that $\theta' \preceq \theta$ iff θ' is a more abstracted form of θ .
4. sub is an injective function, mapping traces to maximal (w.r.t. \preceq) sub-observations $\theta \in \mathbb{O}$:

$$sub : \Sigma_o^* \rightarrow \mathbb{O}$$

A sub-observation θ implicitly represents the set of traces for which it is a more abstract form of:

$$\psi(\theta) = \{o \in \Sigma_o^* \mid \theta \preceq sub(o)\}$$

Therefore, $\theta' \preceq \theta \Rightarrow \psi(\theta') \supseteq \psi(\theta)$.

The language of a sub-observation, denoted \mathcal{L}_θ , represents the set of all possible runs θ could represent. However, these runs are already captured by \mathcal{L}_o , and so \mathcal{L}_θ can be expressed as the union of the languages of the traces it is a more abstract form of:

$$\mathcal{L}_\theta = \bigcup_{o \in \psi(\theta)} \mathcal{L}_o \quad (3)$$

Specific Definition

For the purposes of our specific definition of sub-observations, it is necessary to distinguish between what we call *hard* and *soft* events. A *hard event* is a singleton observable event, $x \in \Sigma_o$, and represents the firm occurrence of an

event in the system. A *soft event* is a subset of observable events, $y \subseteq \Sigma_o$, that any number (including zero) of which may have occurred along with any number of unobservable events.

We now explicitly characterize our construction of sub-observations based on the general framework presented in Definition 1:

Definition 2 A *sub-observation*, θ , is a strict time-ordered alternating sequence of soft and hard events, commencing and ending with a soft event: $\theta = y_0x_1y_1 \dots x_ny_n$. We denote $\mathbb{O}(o)$ the space of sub-observations for a given trace o . $\theta \in \mathbb{O}$ has length $|\theta| = n$. For readability, sub-observations may occasionally be written as a comma separated list. The language of θ can then also be expressed:

$$\mathcal{L}_\theta = (y_0 \cup \Sigma_u)^* x_1 (y_1 \cup \Sigma_u)^* \dots x_n (y_n \cup \Sigma_u)^*$$

By way of example, take the sub-observation $\theta = (\{b, d\}, a, \emptyset, c, \{a\})$ – in this case, we say the singleton events $x_1 = a$ and $x_2 = c$ are *hard* and occurred in the specified order. The first soft event, $y_0 = \{b, d\}$, represents the possibility of any number of b or d events in any order having occurred before the first hard event – similarly, $y_1 = \emptyset$ indicates that no events occurred between the hard events x_1 and x_2 , and $y_2 = \{a\}$ that any number of a events could have occurred after the final hard event. There are multiple traces \hat{o} that this could represent, ac being the simplest, but traces such as $ddacaa$ or bac , or indeed up to infinite (or bounded length depending) other possibilities.

Definition 3 The function *sub* generates a sub-observation in \mathbb{O} from a given trace by inserting empty soft events at the head of the trace, and after every hard event:

$$\begin{aligned} \text{For } o = e_1 \dots e_n \\ \text{sub}(o) = \emptyset x_1 \emptyset \dots x_n \emptyset \in \mathbb{O} \\ \text{Where } \forall i : x_i = e_i \end{aligned}$$

Definition 4 The relation \preceq over \mathbb{O} is defined such that $\theta' \preceq \theta$ if and only if there exists a mapping function f :

$$\begin{aligned} \text{Given } |\theta'| = n, |\theta| = m \\ f : \{0, \dots, n+1\} \rightarrow \{0, \dots, m+1\} \text{ such that} \\ f(i) < f(i+1), f(0) = 0, f(n+1) = m+1 \\ x'_i = x_{f(i)} \\ y'_i \supseteq \bigcup_{f(i) \leq j \leq f(i+1)-1} y_j \cup \bigcup_{f(i) < j < f(i+1)} x_j \end{aligned}$$

The relation \preceq is provably a partial order.

In words: $\theta' \preceq \theta$ if there exists some f that maps the hard events in θ' to an equivalent sequence in θ , retaining the time-ordering of both, and each y'_i in θ' captures the union of all intervening events – y_j (inclusive) and x_j (exclusive), for j ranging between $f(i)$ and $f(i+1) - 1$. For example take $\theta = (\{ac\}, b, \{cd\}, a, \{c\}, d, \{c\}, a, \emptyset)$ and $\theta' = (\{abcd\}, a, \{bcd\}, a, \emptyset)$. The hard events in θ' are matched to x_2 and x_4 in θ , and each y'_i “swallows” the other information. Specifically, $f(1) = 2, f(2) = 4$, satisfies the constraints for $\theta' \preceq \theta$. This is illustrated in Figure 2.

To summarize, a sub-observation, in a practical sense, can be thought of as a relaxation of the information presented in the original trace. By including soft events in the sub-observation, we are allowing for the “hiding” (abstraction) of events such that an operator can be presented with only the most relevant information.

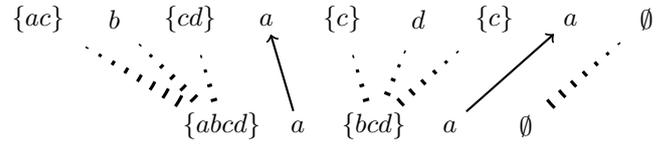


Figure 2: An example map satisfying \preceq

3.3 Diagnosis of Sub-Observations

We now formalize the usage of sub-observations in a diagnosis procedure by extending the procedure introduced for event-based diagnosis presented in §2. This involves checking the consistency of a set of possible faults.

We therefore provide the construction of the diagnosis of θ , $\Delta(\theta)$, the set of faults consistent with a given sub-observation:

Definition 5 The *diagnoses* of a sub-observation θ is the union of the diagnoses of the traces for which θ is the more abstract form of, represented by $\psi(\theta)$ as given in Definition 1:

$$\Delta(\theta) = \bigcup_{o \in \psi(\theta)} \Delta(o)$$

From Definition 5 we note that, given $\hat{\delta} \in \Delta(\hat{o})$, that if $\theta \preceq \text{sub}(\hat{o})$ then $\hat{\delta} \in \Delta(\theta)$. That is, the actual diagnosis $\hat{\delta}$ of the actual trace \hat{o} , will by definition be in $\Delta(\theta)$ if θ is an abstraction of \hat{o} .

First, we observe the following lemma:

Lemma 3.1 The possible traces permitted by the language of a more abstracted sub-observation strictly contains all the permitted traces of all its ascendants:

$$\theta' \preceq \theta \implies \mathcal{L}_{\theta'} \subseteq \mathcal{L}_\theta$$

Proof This is a direct consequence of Equation 3

Equation 2 provided a formulation of the diagnosis as a question of emptiness in the intersection of languages – that is, is there some run that is simultaneously possible according to the system model, the observations, and the faults that occurred during the run. This can similarly be extended to a similar question for sub-observations. As \mathcal{L}_θ is defined in Definition 2, then $\Delta(\theta)$ can be equivalently extended:

$$\Delta(\theta) \equiv \{\delta \mid \mathcal{L}_M \cap \mathcal{L}_\delta \cap \mathcal{L}_\theta \neq \emptyset\} \quad (4)$$

Definition 5 and Equation 4 provide a formal definition and a characterization of the diagnosis of a sub-observation, but do not specify how to implement the procedure, in particular given that $\psi(\theta)$ may be infinitely large.

The scientific literature is rich in works dealing with abstract traces. These approaches were developed to handle situations where observations can be lost [4]; sensors can fail [5; 4]; the order between observations may be only partially known [6; 4; 7; 8]; the observability can vary [9]; etc.

It is possible to interpret Equation 4 quite literally—build three finite-state machines representing all three languages \mathcal{L}_M , \mathcal{L}_δ , and \mathcal{L}_θ , synchronize them, and verify emptiness. Similarly, this emptiness verification can be reduced to a planning problem [10; 11] or a model-checking one [12].

When the model is represented by a finite-state machine, the specific definition of sub-observations makes it possible to solve the problem by tracking the belief state (the set of states that the system could be in) after each soft and hard

event in the sub-observation. Assuming the system state incorporates the diagnosis information, then the diagnosis can be inferred from the belief state at the end of the sub-observation. This procedure can be used on-line [13] or pre-processed in a fashion akin to the diagnoser [14].

4 Critical Observations

The primary objective of this work is to compute a minimal sub-observation that preserves the assertiveness of the diagnosis: a critical observation. We first give a formal definition of this notion, followed by a discussion of some relevant properties and a procedure for computing the critical observation.

4.1 Definition of a Critical Observation

We say a sub-observation is sufficiently precise if it allows us to infer a given diagnosis:

Definition 6 *Given a diagnosis D , a sub-observation θ is sufficient to prove D if $\Delta(\theta) = D$. Given a trace \hat{o} , a sub-observation $\theta \preceq \text{sub}(\hat{o})$ is sufficient for \hat{o} if $\Delta(\theta) = \Delta(\hat{o})$.*

A corollary of Definition 5 gives us $\Delta(\theta) \supseteq \Delta(\hat{o})$. As previously noted, abstracting away details to produce a sub-observation sacrifices some information about the system behavior—sufficiency, then, is the property that this information loss did not affect the diagnosis by making feasible other potential diagnoses:

$$\Delta(\theta) \setminus \Delta(\hat{o}) = \emptyset \quad (5)$$

Our goal, then, is to return a sub-observation that is sufficient for the actual trace, \hat{o} . By Definitions 1, and 5, we see that the naïve sub-observation, $\text{sub}(\hat{o})$, satisfies the criteria to be sufficient for \hat{o} , and means that at least one solution can be found:

$$\Delta(\text{sub}(\hat{o})) = \bigcup_{o \in \psi(\text{sub}(\hat{o}))} \Delta(o) = \Delta(\hat{o})$$

Given two sub-observations θ and θ' , a human operator will, from our initial assumptions, better understand and assimilate a diagnosis with θ' if θ' is more abstract than θ . We therefore search for a “most abstract”, or *critical*, sub-observation, defined as follows:

Definition 7 *Given a trace \hat{o} , a sub-observation $\theta \preceq \text{sub}(\hat{o})$ is critical for \hat{o} if it is sufficient for \hat{o} and there is no strict sub-observation of θ that is also sufficient:*

$$\forall \theta' \in \mathbb{O}. (\theta' \preceq \theta) \wedge (\Delta(\theta') = \Delta(\hat{o})) \Rightarrow (\theta' = \theta). \quad (6)$$

A critical sub-observation (more simply called a critical observation) is therefore a sufficient sub-observation that cannot be abstracted more without damaging (complicating) the precision of the diagnosis.

As \preceq is only a partial order, it is possible that there could be several critical sub-observations. For instance, using the example in Figure 1, both $\theta_1 = \Sigma_o c \Sigma_o a \Sigma_o$ (the system emitted a c and later an a) and $\theta_2 = (\Sigma_o \setminus \{a\}) d \Sigma_o a \Sigma_o$ (the system emitted anything bar an a , then a d and later an a) are critical observations for the trace $\hat{o} = cda$.

4.2 Computing the Critical Observation

We now outline a procedure for computing a critical observation for a given problem. We rely on two fundamental properties: the finiteness of the set of sub-observations of interest, and the monotonicity of sufficiency.

Lemma 4.1 (Finiteness) *Given a trace \hat{o} , the set $\mathbb{O}(\hat{o})$ of sub-observations of \hat{o} ($\{\theta \in \mathbb{O} \mid \theta \preceq \text{sub}(\hat{o})\}$) is finite.*

Proof This can be demonstrated by the fact that, by definition of \preceq , the length of a sub-observation of \hat{o} must be equal to or smaller than that of \hat{o} . This can only decrease until $|\theta| = 1$, at which point the set is exhausted.

Lemma 4.2 (Monotonicity) *Given a trace \hat{o} and two sub-observations θ_1, θ_2 such that $\theta_1 \preceq \theta_2 \preceq \text{sub}(\hat{o})$, if θ_1 is sufficient for \hat{o} , then so is θ_2 .*

Proof This is a straightforward consequence of the fact that $\psi(\theta_1) \supseteq \psi(\theta_2)$.

Monotonicity guarantees that there is no unreachable “island” of sufficient sub-observations.

Finiteness provides us three decisive properties: One—that there always exists at least one critical observation (infinite domains can prevent the existence of minimal elements; e.g., there is no minimal real number strictly greater than 0), Two—that for any sufficient sub-observation θ , there exists a critical observation that is a sub-observation of θ (possibly θ itself), Three—the *depth* of a critical observation (the maximal number k of different sub-observations θ_i such that $\theta \preceq \theta_1 \preceq \dots \preceq \theta_k \preceq \text{sub}(\hat{o})$) is finite.

As a consequence of these properties, as soon as a sufficient sub-observation θ is found the search for a critical observation can be limited to the set of sub-observations of θ (we call this a *greedy* approach). Another consequence of the above is that we can define a search algorithm that can find a sufficient, strict sub-observation of a given sub-observation (or return that no such sub-observation exists), that is guaranteed to terminate.

Finally, monotonicity together with finiteness, provides a practical characterization of criticality: a sufficient sub-observation θ is critical if and only if none of its children (defined next) are sufficient.

Definition 8 *A child of sub-observation θ is a strict sub-observation θ' of θ such that no sub-observation sits “between” θ' and θ .*

$$\theta' \in \text{children}(\theta) \iff (\theta' \prec \theta) \wedge (\nexists \theta'' \in \mathbb{O}. \theta' \prec \theta'' \prec \theta).$$

If, on the other hand, we find that one child of θ is sufficient, then, according to the greedy approach described previously, we can iteratively check criticality of this child.

The set of children for our definition of sub-observation is readily computable. We can prove that the children of a sub-observation are exactly the sub-observations obtained by applying one of two operations which we will now define: the event-softening operation and the collapse operation.

Definition 9 *Given a sub-observation $\theta = y_0 x_1 \dots x_k y_k$, the event-softening operation $\theta' = \text{es}(\theta, i, e)$ adds event e to the i th soft event of the sub-observation: $\text{es}(\theta, i, e) = y'_0 x'_1 \dots x'_k y'_k$ (defined if $e \notin y_i$) such that*

- $\forall j \in \{1, \dots, k\}. x'_j = x_j$,
- $\forall j \in \{0, \dots, k\} \setminus \{i\}. y'_j = y_j$, and
- $y'_i = y_i \cup \{e\}$.

```

Procedure FINDCRITICALOBSERVATION
: trace  $\hat{o}$ ; output: critical observation
diag :=  $\Delta(\hat{o})$ 
 $\theta := \text{sub}(\hat{o})$ 
candidates := children( $\theta$ )
while candidates  $\neq \emptyset$  do
     $\theta' := \text{pop}(\text{candidates})$ 
    if  $\Delta(\theta') = \text{diag}$  then
         $\theta := \theta'$ 
        candidates := children( $\theta$ )
    end if
end while
return  $\theta$ 
    
```

Figure 3: Finding a critical observation

Definition 10 Given a sub-observation $\theta = y_0x_1 \dots x_ky_k$, the collapse operation $\theta' = \text{coll}(\theta, i)$ “forgets” the concrete occurrence of a hard event x_i . This operation requires the soft events before and after x_i to be equal and to allow for x_i : $\text{coll}(\theta, i) = y_0x'_1y'_1 \dots x'_{k-1}y'_{k-1}$ (defined if $x_i \in y_i$ and $y_{i-1} = y_i$) such that

- $\forall j \in \{1, \dots, i-1\}. x'_j = x_j$ and $y'_{j-1} = y_{j-1}$ and
- $\forall j \in \{i+1, \dots, k\}. x'_{j-1} = x_j$ and $y'_{j-1} = y_j$.
- $y'_{i-1} = y_{i-1} = y_i$

Lemma 4.3 The children of a sub-observation θ are exactly all the sub-observations that can be obtained by applying either event-softening or collapse to θ . (See appendix for proof).

An algorithm for finding a critical observation is given in Figure 3. Starting from $\theta = \text{sub}(\hat{o})$, the algorithm verifies whether any child of θ is sufficient. If this is the case, then θ is replaced with this child and the verification continues iteratively.

Theorem 4.4 Algorithm FINDCRITICALOBSERVATION always terminates and returns a critical observation.

This theorem is a direct consequence of the properties derived from the finiteness of $\mathbb{O}(\text{sub}(\hat{o}))$ and the monotonicity of the property, as described before.

4.3 Complexity

We now discuss the difficulty of finding a critical observation, defined in term of the number of $\Delta(\cdot)$ calls. Let $n = |\hat{o}|$ be the length of \hat{o} (the number of observed events) and let $m = |\Sigma_o|$ be the number of observable events.

The maximal depth, D , of a sub-observation, namely that of $\theta_0 = \{\Sigma_o\}$, is provably $D = (n+1)m + n$: It is reached by softening m times each of the $n+1$ soft events followed by collapsing the n hard events. Furthermore each sub-observation (of length $k \leq n$), can be shown to have a bounded number of children, C , as given by Definition 8: At worst each soft event can be softened in any one of m ways ($(k+1)m$), and a potentially up to k hard events can be collapsed, giving $C = (k+1)m + k$, which we see is the same as D .

Consequently, the maximum number of $\Delta(\cdot)$ calls of Algorithm FINDCRITICALOBSERVATION is bounded by $D \times C$, and therefore in $O(n^2m^2)$. It can even be shown that, for some traces, a naïve implementation may indeed call the

diagnoser a number of times in $\Theta(n^2m^2)$ with different sub-observations every time (see appendix for proof).

Fortunately it is possible to reduce this number drastically with heuristics. Indeed it is generally possible to prove that some children of θ' are not sufficient simply because some children of the parent of θ' were proven not sufficient, thus pruning the search tree significantly.

Consider for instance the sub-observation $\theta = \emptyset a \emptyset b \emptyset a \emptyset a \emptyset$ in the example of Figure 1 (with diagnosis: fault f_1). The softening by b of the soft event $y_3 = \emptyset$ between $x_3 = x_4 = a$ leads to a sub-observation ($\emptyset a \emptyset b \emptyset a \{b\} a \emptyset$) that is not sufficient, as the nominal diagnosis N becomes possible. Consider now the sub-observation $\theta' = \Sigma_o a \emptyset a \Sigma_o$ of θ . We can deduce automatically that the softening of $y'_1 = \emptyset$ by b in θ' leads to a non sufficient sub-observation, simply because the mapping function f of Definition 4 associates y'_1 with y_3 .

It is therefore possible to “carry over” to the children of any sub-observation the information regarding which softening and collapse operations complicate the diagnosis and reduce precision. By doing so, the number of necessary calls provably drops to $\Theta(nm)$.

5 Other Definitions of Sub-Observations

In this article we presented one definition of sub-observation that, by no means, is the only viable one. We briefly discuss a few possible variants and then present the necessary elements that the reader would need to consider to use another definition.

In many circumstances the order between certain observed facts is irrelevant. In the example of Figure 1, the occurrence of both c and a , in any order, is symptomatic of fault f_2 . Reminiscent of chronicles [15], a sub-observation could be a directed graph of hard events where a directed path between two hard events expresses a temporal precedence. This bears a similarity to temporal uncertainty in observations as described by Zanella and Lamperti [4].

The hard events are currently defined as a single specific observable event; one could alter the definition to allow for it could be replaced by a set of events. Indeed in the example of Figure 1, a fault can be diagnosed when observing either c or d before e . A reason for not distinguishing c from d in this specific scenario is that these events could represent the same message emitted by different components, or different messages emitted by the same component: the exact emitter of the message or the exact content may be irrelevant to diagnose the fault. This is similar to logical uncertainty in observations [4].

One more elaborate abstraction could be to use first-order representations. For instance, a fault may be identified by demonstrating that some user who was to be explicitly refused access to some data was actually given access to that data; the identity of the actual user may be irrelevant.

Defining New Sub-Observations

To apply the theory presented in this paper to a different definition of sub-observations, one needs to define the sub-observation space as given in Definition 1, i.e., the set of sub-observations \mathbb{O} , the partial order relation \preceq , and an inductive *sub* function that associates each observation with an equivalent maximal sub-observation in \mathbb{O} . This also needs to be additionally equipped with a procedure to compute $\Delta(\theta)$. Algorithm FINDCRITICALOBSERVATION

is guaranteed to return a critical observation if the sub-observation space is finite and if the *children* function exists and is specified.

We demonstrate a scenario where these conditions may not be satisfied: Assume that the set of observable events is infinite with each observable event associated with a rational number (modeling some continuous property, e.g., temperature). A natural abstraction would replace each event by a closed interval where the value associated with the event lies (the wider the interval, the most abstract the observation). There could, however, be no maximal interval in a situation where the relevant information about the observation is that the temperature measure is strictly positive. Furthermore, there is no notion of child in this particular sub-observation space since \mathbb{Q} is a dense set. Special attention must therefore be taken when defining new types of sub-observations.

6 Related Work

Finding critical observations is a different issue from optimizing sensor placement [16] and dynamic observers [9]. These two problems aim at reducing the cost of monitoring a system (by reducing the number of sensors or switching them off). This reduction, however, needs to be conservative because the decision is made before any observations are available. Critical observations, on the other hand, can be computed after all observations are available.

Consider again the trace $o = abaa$ in the example of Figure 1 whose critical observation is $\theta = \Sigma_o a \emptyset a \Sigma_o$. Consider the question of whether the first observable event of the trace is a c . The sub-observation θ does not provide this information since it is not necessary to infer the diagnosis. A dynamic observer however, has to check this information because it is necessary to dismiss fault f_2 .

There has also been work on abstraction of event-based observations, as mentioned at the end of section 3. The subsumption (\preceq) between uncertain or partial observations has been studied by Lamperti et al. [17], although their motivation is different from ours: by identifying that the current uncertain observation θ is a refinement of a previous observation $\theta' \preceq \theta$, it is possible to reuse the diagnosis of θ' (that is, $\Delta(\theta) \subseteq \Delta(\theta')$).

7 Conclusion & Future Work

In this work we defined a notion of critical observations for the diagnosis of discrete event systems. A critical observation is a maximally abstracted observation that allows only the same diagnosis to be inferred as was from the complete observation. Critical observations are beneficial in that they contain the core proof that supports the diagnosis. An important assumption of this work is that more abstract observations are easier for a human operator to understand and act on; an important extension will be to minimize the amount of information from the model—and not only from the observations—necessary to infer the diagnosis.

We also want to be able to handle incremental and on-line diagnosis. Currently we assume that the critical observation is extracted once the diagnosis has been performed; however observations that are not critical for a given trace might become critical when more observations are produced by the system. We would like to identify as early as possible what abstraction of the currently received observations can be safely made without impairing the future diagnosis. Kurien and Nayak tried to address a similar problem [18]

of removing intermediate (state-based) observations that do not provide additional information.

Critical observations are also good at reducing the amount of information disclosed about the system behaviour. In future work we want to explore this line of research and, in particular, examine the problem of finding sub-observations that satisfy a privacy criterion, for instance, one defined by opacity [19].

References

- [1] G. A. Miller, “The magical number seven, plus or minus two: some limits on our capacity for processing information,” *Psychological review*, vol. 63, no. 2, p. 81, 1956.
- [2] C. Christopher, M.-O. Cordier, and A. Grastien, “Critical observations in a diagnostic problem,” in *IEEE Conference on Decision and Control*, 2014, pp. 382–387.
- [3] C. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.
- [4] M. Zanella and G. Lamperti, *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.
- [5] L. Carvalho, M. Moreira, J. Basilio, and S. Lafortune, “Robust diagnosis of discrete-event systems against permanent loss of observations,” *Automatica*, vol. 49, no. 1, pp. 223–231, 2013.
- [6] R. Debouk, S. Lafortune, and D. Teneketzis, “Coordinated decentralized protocols for failure diagnosis of discrete event systems,” *Journal of Discrete Event Dynamical Systems*, vol. 10, no. 1–2, pp. 33–86, 2000.
- [7] Y. Pencolé and M.-O. Cordier, “A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks,” *Artificial Intelligence (AIJ)*, vol. 164, no. 1–2, pp. 121–170, 2005.
- [8] R. Su and W. Wonham, “Global and local consistencies in distributed fault diagnosis for discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 50, no. 12, pp. 1923–1935, 2005.
- [9] F. Cassez and S. Tripakis, “Fault diagnosis with dynamic observers,” in *International Workshop on Discrete Event Systems*, 2008, pp. 212–217.
- [10] S. Sohrabi, J. Baier, and S. McIlraith, “Diagnosis as planning revisited,” in *International Conference on the Principles of Knowledge Representation and Reasoning*, 2010, pp. 26–36.
- [11] P. Haslum and A. Grastien, “Diagnosis as planning: two case studies,” in *Scheduling and Planning Applications Workshop*, 2011, pp. 37–44.
- [12] M.-O. Cordier and C. Largouët, “Using model-checking techniques for diagnosing discrete-event systems,” in *International Workshop on Principles of Diagnosis*, 2001, pp. 39–46.
- [13] A. Schumann, Y. Pencolé, and S. Thiébaux, “A spectrum of symbolic on-line diagnosis approaches,” in *Conference on Artificial Intelligence*, 2007.

- [14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamo-
hideen, and D. Teneketzis, “Diagnosability of discrete-
event systems,” *IEEE Transactions on Automatic Con-
trol*, vol. 40, no. 9, pp. 1555–1575, 1995.
- [15] M.-O. Cordier and C. Dousson, “Alarm driven mon-
itoring based on chronicles,” in *IFAC Symposium on
Fault Detection, Supervision and Safety of Technical
Processes*, 2000, pp. 286–291.
- [16] L. Brandán Briones, A. Lazovik, and P. Dague, “Opti-
mal observability for diagnosability,” in *International
Workshop on Principles of Diagnosis*, 2008, pp. 31–
38.
- [17] G. Lamperti, F. Vivenzi, and M. Zanella, “On sub-
sumption, coverage, and relaxation of temporal obser-
vations in reuse-based diagnosis of discrete-event
systems: a unifying perspective,” in *20th International
Workshop on Principles of Diagnosis (DX-09)*, 2009,
pp. 353–360.
- [18] J. Kurien and P. Nayak, “Back to the future for
consistency-based trajectory tracking,” in *Conference
on Artificial Intelligence*, 2000, pp. 370–377.
- [19] F. Cassez, J. Dubreil, and H. Marchand, “Synthesis of
opaque systems with static and dynamic masks,” *Formal
Methods in System Design*, vol. 40, no. 1, pp. 88–
115, 2012.

8 Appendix

We provide proof sketches that will not be included in the
final version of the paper.

Proof of Lemma 4.3

The proof is three-part:

- proving that the event-softening operation produces
only children;
- proving that the collapse operation produces only chil-
dren;
- proving that there is no other child.

Event-Softenings It is easy to see that $\theta^2 \stackrel{def}{=} es(\theta^1, i, e) \prec \theta^1$.

Assume now that $\theta^2 \preceq \theta^3 \preceq \theta^1$ and let f_{23} and f_{31} be the
two mapping functions—as presented in Definition 4—used
to verify the two ordering relations.

By definition of \preceq , $|\theta^2| \leq |\theta^3| \leq |\theta^1|$. However since
 $|\theta^2| = |\theta^1|$ (by definition of event-softening), the size of
all three sub-observations are equal and $f_{23} = f_{31}$ are the
identity function.

As a consequence, $x_j^3 = x_j^2 = x_j^1$ for all j . Furthermore
 $y_j^2 \supseteq y_j^3 \supseteq y_j^1$ for all j . In particular, if $j \neq i$, since $y_j^2 = y_j^1$,
then $y_j^3 = y_j^2 = y_j^1$. For i , $y_i^2 = y_i^1 \cup \{e\}$, meaning that
either $y_i^3 = y_i^2$ or $y_i^3 = y_i^1$.

Therefore either $\theta^3 = \theta^2$ or $\theta^3 = \theta^1$.

Collapse Similarly, it is easy to see that $\theta^2 \stackrel{def}{=} coll(\theta^1, i) \prec \theta^1$.

Again assume that $\theta^2 \preceq \theta^3 \preceq \theta^1$ and let f_{23} and f_{31} be
the functions defined as before.

The size of θ^3 now either equals that of θ^2 or θ^1 ; let $\ell \in$
 $\{1, 2\}$ denote the index such that $|\theta^3| = |\theta^\ell|$. Notice that
either f_{23} or f_{31} is the identity function.

By definition of \preceq , we know that $x_j^3 = x_j^\ell$. Furthermore
the set inclusions as well as the relations between y_j^2 and y_k^1
allow us to infer that $y_j^3 = y_j^\ell$ for all j .

Therefore $\theta^3 = \theta^\ell$.

No Other Children Assume now that θ' is a child of θ that
cannot be obtained by event-softening or collapse. Let f be
the mapping function used to verify the ordering relation.

By definition of the partial order \preceq , the size of θ' is
smaller or equal to θ .

If $|\theta'| < |\theta|$ (“multiple collapse”), then let i be an index
such that $f(i+1) > f(i) + 1$ (such an index exists if the two
sizes differ). If $y_{i+1} \setminus y_i \neq \emptyset$, then let $\theta'' = es(\theta, i, e)$ (where
 $e \in y_{i+1} \setminus y_i$) be the sub-observation obtained by softening
 y_i with e ; then, $\theta' \prec \theta'' \prec \theta$. Similarly if $y_i \supseteq y_{i+1}$ with
 $\theta'' = es(\theta, i+1, e)$ (where $e \in y_i \setminus y_{i+1}$). Lastly the same
applies if $y_i = y_{i+1}$ with $\theta'' = coll(\theta, i)$.

If θ and θ' have same size, then all x'_i s equal the cor-
responding x_i s, and all the y'_i s are supersets of the cor-
responding y_i s. Let i be an index such that $y'_i \neq y_i$ (if no
such index exists, then $\theta' = \theta$). Let $\theta'' = es(\theta, i, e)$ where
 $e \in y'_i \setminus y_i$. Then $\theta' \prec \theta'' \prec \theta$.

Complexity of FINDCRITICALOBSERVATION

We show that the number of $\Delta(\cdot)$ calls in FINDCRIT-
ICALOBSERVATION could be in the order of $\frac{n^2 m^2}{4}$ where
 n is the length of the trace and m the number of observable
events.

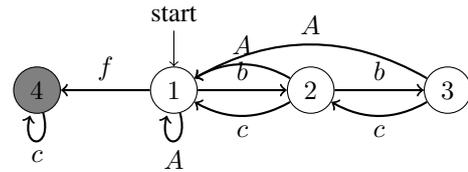


Figure 4: Example of a system: a fault is diagnosed if there
are more cs than bs after the occurrence of the last a_i (A
stands for $\{a_1, \dots, a_{m-2}\}$).

We use the example of Figure 4 which involves faulty
event f and observable events $\{a_1, \dots, a_{m-2}, b, c\}$. Con-
sider the trace of (odd) length n : $\hat{o} = \underbrace{a_1 \dots a_1}_{n/2} \underbrace{bc \dots bc}_{n/2} c$.

Clearly the trace reveals a faulty system since the number
of cs exceeds the number of bs in this instance. The critical
observation here is:

$$\Sigma_o a_1 \{c\} b \{c\} c \{c\} \dots \{c\} b \{c\} c \{c\} c \Sigma_o,$$

i.e., all the second half of the trace needs to be kept.

We assume that FINDCRITICALOBSERVATION always
tries to perform event-softening from the end of the sub-
observation first, and only tries to collapse when no soft-
ening is possible. Neglecting the first steps where the c
softenings are successful, the algorithm will need to make
 $U = \frac{n}{2} \times (m-1)$ calls to $\Delta(\cdot)$, unsuccessfully trying to
softening the second half of the sub-observation. The num-
ber of successful softenings however is $S = \frac{n}{2} \times m$ (all the
first half of the sub-observation), meaning that the number
of $\Delta(\cdot)$ calls will be at least $U \times S = \frac{n^2 m(m-1)}{4}$ calls.

A Framework For Assessing Diagnostics Model Fidelity

Gregory Provan¹ and Alex Feldman²

¹Computer Science Department, University College Cork, Cork, Ireland

e-mail: g.provan@cs.ucc.ie

²PARC Inc., Palo Alto, CA 94304, USA

e-mail: afeldman@parc.com

Abstract

“All models are wrong but some are useful” [1]. We address the problem of identifying which diagnosis models are more useful than others. Models are critical to diagnostics inference, yet little work exists to be able to compare models. We define the role of models in diagnostics inference, propose metrics for models, and apply these metrics to a tank benchmark system. Given the many approaches possible for model metrics, we argue that only information-theoretic methods address how well a model mimics real-world data. We focus on some well-known information-theoretic modelling metrics, demonstrating the trade-offs that can be made on different models for a tank benchmark system.

1 Introduction

A core goal of Model-Based Diagnostics (MBD) is to accurately diagnose a range of systems in real-world applications. There has been significant progress in developing algorithms for systems of increasing complexity. A key area where further work is needed is scaling-up to real-world models, as multiple-fault diagnostics algorithms are currently limited by the size and complexity of the models to which they can be applied. In addition, there is still a great need for defining metrics to measure diagnostics accuracy, and to measure the computational complexity of inference and of the models’ contribution to inference complexity.

This article addresses the modeling side of MBD: we focus on methods for measuring the size and complexity of MBD models. We explore the role that diagnostics model fidelity can play in being able to generate accurate diagnostics. We characterise model fidelity and examine the trade-offs of fidelity and inference complexity within the overall MBD inference task.

Model fidelity is a crucial issue in diagnostics [2]: models that are too simple can be inaccurate, yet highly detailed and complex models are expensive to create, have many parameters that require significant amounts of data to estimate, and are computationally intensive to perform inference on. There is an urgent need to incorporate inference complexity within modelling, since even relatively simple models, such as some of the combinatorial ISCAS-85 benchmark models, pose computational challenges to even the most advanced solvers for multiple-fault tasks. In addition, higher-fidelity

models can actually perform worse than lower-fidelity models on real-world data, as can be explained using over-fitting arguments within a machine learning framework.

To our knowledge, there is no theory within Model-Based Diagnostics that relates notions of model complexity, model accuracy, and inference complexity. To address these issues, we explore several of the factors that contribute to model complexity, as well as a theoretically sound approach for selecting models based on their complexity and diagnostics performance, i.e., their accuracy in diagnosing faults.

Our contributions are as follows:

- We characterise the task of selecting a diagnosis model of appropriate fidelity as an information-theoretic model selection task.
- We propose several metrics for assessing the quality of a diagnosis model, and derive approximation versions of a subset of these metrics.
- We use a dynamical systems benchmark model to demonstrate our compare how the metrics assess models relative to the accuracy of diagnostics output based on using the models.

2 Related Work

This section reviews work related to our proposed approach.

Model-Based Diagnostics: There is some seminal work on modelling principles within the Model-Based Diagnosis (MBD) community, e.g., [2; 3]; this early work adopts an approach based on logic or qualitative physics for model specification. However, this work provides no means for comparing models in terms of diagnostics accuracy. More recent work ([4]) provides a logic-based specification of model fidelity. There is also work specifying metrics for diagnostics accuracy, e.g., [5].

However, none of this work defines precise metrics for computing both diagnostics accuracy and model complexity, and their trade-offs. This article adopts a theoretically well-founded approach for integrating multiple MBD metrics.

Multiple Fidelity Modeling There is limited work describing the use of models of multiple levels of fidelity. Examples of such work includes [6; 7; 8]. In this article we focus on methods for evaluating multi-fidelity models and their impact on diagnostics accuracy, as opposed to developing methodologies for modelling at multiple levels of fidelity.

Multiple-Mode Modeling One approach to MBD is to use a separate model for every failure mode, rather than to

define a model containing all failure modes. Examples of this approach include [9; 10; 11; 12]. Note that this work does not specify metrics for computing *both* diagnostics accuracy and model complexity, or their trade-offs.

Model-Selection The metrics that we adopt and extend have been used extensively to compare different models, e.g., [13]. The metrics are used to compare *simulation performance* of models only. In contrast, we extend this framework to examine *diagnostics performance*. In the process, we explore the use of multiple loss functions for penalising models, in addition to the standard penalty functions based on number of model parameters.

Model-Order Reduction Model-Order reduction [14] aims to reduce the complexity of a model with an aim to limit the performance losses of the reduced model. The reduction methods are theoretically well-founded, although they are highly domain-specific. In contrast to this approach, we assume a model-composition approach from a component library containing hand-constructed models of multiple levels of fidelity.

3 Diagnostics Modeling and Inference

This section formalises the notion of diagnostics model within the process of diagnostics inference. We first introduce the task, and then define it more precisely.

3.1 Diagnosis Task

Assume that we have a system \mathcal{S} that can operate in a nominal state, ξ_N , or a faulty state, ξ_F , where Ξ is the set of possible states of \mathcal{S} . We further assume that we have a discrete vector of measurements, $\tilde{\mathbf{Y}} = \{\tilde{y}_1, \dots, \tilde{y}_n\}$ observed at times $t = \{1, \dots, n\}$ that summarizes the response of the system \mathcal{S} to control variables $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_n\}$. Let $\mathbf{Y}_\phi = \{y_1, \dots, y_n\}$ denote the corresponding predictions from a dynamic (nonlinear) model, ϕ , with parameter values θ : this can be represented by $\mathbf{Y}_\phi = \phi(x_0, \theta, \xi, \tilde{\mathbf{U}})$, where x_0 signifies the initial states of the system at t_0 .

We assume that we have a prior probability distribution $P(\Xi)$ over the states Ξ of the system. This distribution denotes the likelihood of the failure states of the system.

We define a residual vector $\mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi)$ to capture the difference between the actual and model-simulated system behaviour. An example of a residual vector is the mean-squared-error (MSE). We assume a fixed diagnosis task \mathcal{T} throughout this article, e.g., computing the most likely diagnosis, or a deterministic multiple-fault diagnosis.

The classical definition of diagnosis is as a state estimation task, whose objective is to identify the system state that minimises the residual vector:

$$\xi^* = \operatorname{argmin}_{\xi \in \Xi} \mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi) \quad (1)$$

Since this is a minimisation task, we typically need to run multiple simulations over the space of parameters and modes to compute ξ^* . We can abstract this process as performing model-inversion, i.e., computing some $\xi^* = \phi^{-1}(x_0, \theta, \xi, \tilde{\mathbf{U}})$ that minimises $\mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi)$.

During this diagnostics inference task, a model ϕ can play two roles: (a) simulating a behaviour to estimate $\mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi)$; (b) enabling the computation of $\xi^* = \phi^{-1}(x_0, \theta, \xi, \tilde{\mathbf{U}})$. It is clear that diagnostics inference requires a model that has good fidelity and is computationally efficient for performing these two roles.

We generalise that notion to incorporate inference efficiency as well as accuracy. We can define an inference complexity measure as $\mathcal{C}(\tilde{\mathbf{Y}}, \phi)$. We can then define our diagnosis task as jointly minimising a function g that incorporates the accuracy (based on the residual function) and the inference complexity:

$$\xi^* = \operatorname{argmin}_{\xi \in \Xi} g \left(\mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi), \mathcal{C}(\tilde{\mathbf{Y}}, \phi) \right). \quad (2)$$

Here g specifies a loss or penalty function that induces a non-negative real-valued penalty based on the lack of accuracy and computational cost.

In forward simulation, a model ϕ , with parameters θ , can generate multiple observations $\tilde{\mathbf{Y}} = \{\tilde{y}_1, \dots, \tilde{y}_n\}$. The diagnostics task involves performing the inverse operation on these observations. Our objective thus involves optimising the state estimation task over a future set of observations, $\tilde{\mathbf{Y}} = \{\tilde{\mathbf{Y}}_1, \dots, \tilde{\mathbf{Y}}_n\}$. Our model ϕ and inference algorithm \mathcal{A} have different performance based on $\tilde{\mathbf{Y}}_i, i = 1, \dots, n$: for example, [15] shows that both inference-accuracy and -time vary based on the fault cardinality. As a consequence, to compute ξ^* we want to optimise the *mean* performance over future observations. This notion of *mean* performance optimisation has been characterised using the Bayesian model selection approach, which we examine in the following section.

3.2 Diagnosis Model

We specify a diagnosis model as follows:

Definition 1 (Diagnosis Model). *We characterise a Diagnosis Model ϕ using the tuple $\langle \mathbf{V}, \theta, \Xi, \mathcal{E} \rangle$, where*

- \mathbf{V} is a set of variables, consisting of variables denoting the system state (\mathbf{X}), control (\mathbf{U}), and observations (\mathbf{Y}).
- θ is a set of parameters.
- Ξ is a set of system modes.
- \mathcal{E} is a set of equations, with a subset $E_\xi \subseteq \mathcal{E}$ for each mode $\xi \in \Xi$.

We will assume that we can use a physics-based approach to hand-generate a set \mathcal{E} of equations to specify a model. Obtaining good diagnostics accuracy, given a fixed \mathcal{E} , entails estimating the parameters θ to optimise that accuracy.

3.3 Running Example: Three-Tank Benchmark

In this paper, we use the three-tank system shown in Fig. 1 to illustrate our approach. The three tanks are denoted as T_1 , T_2 , and T_3 . Each tank has the same area $A_1 = A_2 = A_3$. For $i = 1, 2, 3$, tank T_i has height h_i , a pressure sensor p_i , and a valve $V_i, i = 1, 2, 3$ that controls the flow of liquid out of T_i . We assume that gravity $g = 10$ and the liquid has density $\rho = 1$.

Tank T_1 gets filled from a pipe, with measured flow q_0 . Using Torricelli's law, the model can be described by the following non-linear equations:

$$\frac{dh_1}{dt} = \frac{1}{A_1} \left[-\kappa_1 \sqrt{h_1 - h_2} + q_0 \right], \quad (3)$$

$$\frac{dh_2}{dt} = \frac{1}{A_2} \left[\kappa_1 \sqrt{h_1 - h_2} - \kappa_2 \sqrt{h_2 - h_3} \right], \quad (4)$$

$$\frac{dh_3}{dt} = \frac{1}{A_3} \left[\kappa_2 \sqrt{h_2 - h_3} - \kappa_3 \sqrt{h_3} \right]. \quad (5)$$

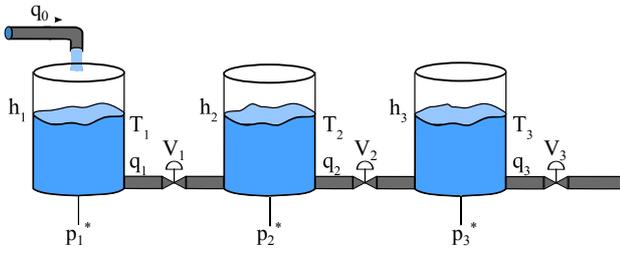


Figure 1: Diagram of the three-tank system.

In eq. 3, the coefficient κ_1 denotes a parameter that captures the product of the cross-sectional area of the tank A_1 , the area of the drainage hole, a gravity-based constant ($\sqrt{2g}$), and the friction/contraction factor of the hole. κ_2 and κ_3 can be defined analogously.

Finally, the pressure at the bottom of each tank is obtained from the height: $p_i = g h_i$, where i is the tank index ($i \in \{1, 2, 3\}$).

We emphasize the use of the κ_i , $i = 1, 2, 3$ because we will use these parameter-values as a means for “diagnosing” our system in term of changes in κ_i , $i = 1, 2, 3$. Consider a physical valve R_1 between T_1 and T_2 that constraints the flow between the two tanks. We can say that the valve changes proportionally the cross-sectional drainage area of q_1 and hence κ_1 . The diagnostic task will be to compute the true value of κ_1 , given p_1 , and from κ_1 we can compute the actual position of the valve R_1 .

We now characterise our nominal model in terms of Definition 1:

- variables \mathbf{V} consist of variables denoting the system state ($\mathbf{X} = \{h_1, h_2, h_3\}$), control ($\mathbf{U} = \{q_0, V_1, V_2, V_3\}$), and observations ($\mathbf{Y} = \{p_1, p_2, p_3\}$).
- $\theta = \{A_1, A_2, A_3, \{\kappa_1, \kappa_2, \kappa_3\}\}$ is the set of parameters.
- Ξ consists of a single nominal mode.
- \mathcal{E} is a set of equations, given by equations 3 through 5.

Note that this model has a total of 6 parameters.

Fault Model In this article we focus on valve faults, where a valve can have a blockage or a leak. We model this class of faults by including in equations 3 to 5 an additive parameter β , which is applied to the parameter κ , i.e., as $\kappa_i(1+\beta_i)$, $i = 1, 2, 3$, where $-1 \leq \beta_i \leq \frac{1}{\kappa_i} - 1$, $i = 1, 2, 3$. $\beta > 0$ corresponds to a leak, such that $\beta \in (0, 1/\kappa - 1]$; $\beta < 0$ corresponds to a blockage, such that $\beta \in [-1, 0)$. The fault equations can be written as:

$$\begin{aligned} \frac{dh_1}{dt} &= \frac{1}{A_1} \left[-\kappa_1(1 + \beta_1)\sqrt{h_1 - h_2} + q_0 \right], \\ \frac{dh_2}{dt} &= \frac{1}{A_2} \left[\kappa_1(1 + \beta_1)\sqrt{h_1 - h_2} \right. \\ &\quad \left. - \kappa_2(1 + \beta_2)\sqrt{h_2 - h_3} \right], \\ \frac{dh_3}{dt} &= \frac{1}{A_3} \left[\kappa_2(1 + \beta_2)\sqrt{h_2 - h_3} - \kappa_3(1 + \beta_3)\sqrt{h_3} \right]. \end{aligned} \quad (6)$$

The fault equations allow faults for any combination of the valves $\{V_1, V_2, V_3\}$, resulting in system modes $\Xi = \{\xi_N, \xi_1, \xi_2, \xi_3, \xi_{12}, \xi_{13}, \xi_{23}, \xi_{123}\}$, where ξ_N is the nominal

mode, and ξ is the mode where \cdot denotes the combination of valves (taken from a combination of $\{1, 2, 3\}$) which are faulty. This fault model has 9 parameters.

4 Modelling Metrics

This section describes the metrics that can be applied to estimate properties of a diagnosis model. We describe two types of metrics, dealing with accuracy (fidelity) and complexity.

4.1 Model Accuracy

Model accuracy concerns the ability of a model to mimic a real system. From a diagnostics perspective, this translates to the use of a model to simulate behaviours that distinguish nominal and faulty behaviours sufficiently well that appropriate fault isolation algorithms can identify the correct type of fault when it occurs. As such, a diagnostics model needs to be able to simulate behaviours for multiple modes with “appropriate” fidelity.

Note that we distinguish model accuracy from diagnosis inference accuracy. As noted above, model accuracy concerns the ability of a model to mimic a real system through simulation, and to assist in diagnostics isolation. Diagnosis inference accuracy concerns being able to isolate the true fault given an observation and the simulation output of a model.

A significant challenge for a diagnosis model is the need to simulate behaviours for multiple modes. Two approaches that have been taken are to use a single model with multiple modes explicitly defined (a multi-mode approach), or to use multiple models [9; 16; 17], each of which is optimised for a single or small set of modes (a multi-model approach).

The AI-based MBD approach typically uses a single model ϕ with multiple modes explicitly defined [18], or a single model with just nominal behaviour [19]. From a diagnostics perspective, accuracy must be defined with respect to the task \mathcal{T} . We adopt here the task of computing the most-likely diagnosis.

Given evidence suggesting that model fidelity for a multi-mode approach varies depending on the mode, it is important to explicitly consider the *mean performance* of ϕ over the entire observation space \mathcal{Y} (the space of possible observations of the system).

In this article we adopt the expected residual approach, i.e., given a space $\mathcal{Y} = \{\tilde{\mathbf{Y}}_1, \dots, \tilde{\mathbf{Y}}_n\}$ of observations, the expected residual is the average over the n observations, e.g., as given by: $\bar{\mathcal{R}} = \frac{1}{n} \sum_{i=1}^n \mathcal{R}(\tilde{\mathbf{Y}}_i, \mathbf{Y}_\phi)$.

4.2 Model Complexity

At present, there is no commonly-accepted definition of model complexity, whether the model is used purely for simulation or if it is used for diagnostics or control. Defining the complexity of a model is inherently tricky, due to the number of factors involved.

Less complex models are often preferred either due to their low computational simulation costs [20], or to minimise model over-fitting given observed data [21; 22]. Given the task of simulating a variable of interest conditioned by certain future values of input (control) variables, overfitting can lead to high uncertainty in creating accurate simulations. Overfitting is especially severe when we have limited observation variables for generating a model representing the underlying process dynamics. In contrast, models with low

parameter dimensionality (i.e. fewer parameters) are considered less complex and hence are associated with low prediction uncertainty [23].

Several approaches have been used, based on issues like (a) number of variables [24], (b) model structure [25], (c) number of free parameters [23], (d) number of parameters that the data can constrain [26], (e) a notion of model weight [27], or (f) *type* and *order* of equations for a non-linear dynamical model [14], where type corresponds to non-linear, linear, etc.; e.g., order for a non-linear model is such that a k -th order system has k -th derivatives in \mathcal{E} .

Factors that contribute to the true cost of a model include: (a) model-generation; (b) parameter estimation; and (c) simulation complexity, i.e., the computational expense (in terms of CPU-time and memory) needed to simulate the model given a set of initial conditions. Rather than try to formulate this notion in terms of the number of model variables or parameters, or a notion of model structural complexity, we specify model complexity in terms of a measure based on parameter estimation, and inference complexity, assuming a construction cost of zero.

A thorough analysis of model complexity will need to take into consideration the model equation class, since model complexity is class-specific. For example, for non-linear dynamical models, complexity is governed by the *type* and *order* of equations [14]. In contrast, for linear dynamical models, which have only matrices and variables in equations (no derivatives), it is the order of the matrices that determines complexity. In this article, we assume that models are of appropriate complexity, and hence do not address Model order reduction techniques [14], which aim to generate lower-dimensional systems that trade off fidelity for reduced model complexity.

4.3 Diagnostics Model Selection Task

The model in this model selection problem corresponds to a system with a single mode. Given a space Φ of possible models, we can define this model selection task as follows:

$$\phi^* = \operatorname{argmin}_{\phi \in \Phi} g_1 \left(\mathcal{R}(\tilde{\mathbf{Y}}, \mathbf{Y}_\phi) \right) + g_2 \left(\mathcal{C}(\tilde{\mathbf{Y}}, \phi) \right), \quad (7)$$

adopting the simplifying assumption that our loss function g is additively decomposable.

4.4 Information-Theoretic Model Complexity

The Information-Theoretic (or Bayesian) model complexity approach, which is based on the model likelihood, measures whether the increased “complexity” of a model with more parameters is justified by the data. The Information-Theoretic approach chooses a model (and a model structure) from a set of competing models (from the set of corresponding model structures, respectively) such that the value of a Bayesian criterion is maximized (or prediction uncertainty in choosing a model structure is minimized).

The Information-Theoretic approach addresses prediction uncertainty by specifying an appropriate likelihood function. In other words, it specifies the probability with which the observed values of a variable of interest are generated by a model. The marginal likelihood of a model structure, which represents a class of models capturing the same processes (and hence have the same parameter dimensionality), is obtained by integrating over the prior distribution of model parameters; this measures the prediction uncertainty of the model structure [28].

Statistical model selection is commonly based on Occam’s parsimony principle (ca.1320), namely that hypotheses should be kept as simple as possible. In statistical terms, this is a trade-off between bias (distance between the average estimate and truth) and variance (spread of the estimates around the truth).

The idea is that by adding parameters to a model we obtain improvement in fit, but at the expense of making parameter estimates “worse” because we have less data (i.e., information) per parameter. In addition, the computations typically require more time. So the key question is how to identify how complex a model works best for a given problem.

If the goal is to compute the likelihood of a given model $\phi(x_0, \theta, \xi, \mathbf{U})$, then θ and \mathbf{U} are nuisance parameters. These parameters affect the likelihood calculation but are not what we want to infer. Consequently, these parameters should be eliminated from the inference. We can remove nuisance parameters by assigning them prior probabilities and integrating them out to obtain the marginal probability of the data given only the model, that is, the model likelihood (also called integrative, marginal, or predictive likelihood). In equational form, this looks like: $P(\mathbf{Y}|\phi) = \int_{\theta} \int_{\mathbf{U}} P(\phi|\mathbf{Y}, \theta, \mathbf{U}) P(\theta, \mathbf{U}|\phi) d\theta d\mathbf{U}$. However, this multi-dimensional integral can be very difficult to compute, and it is typically approximated using computationally intensive techniques like Markov chain Monte Carlo (MCMC).

Rather than try to solve such a computationally challenging task, we adopt an approximation to the multidimensional integral. In the statistics literature several decomposable approximations have been proposed.

Spiegelhalter et al. [26] have proposed a well-known such decomposable framework, termed the Deviance Information Criterion (DIC), which measures the number of model parameters that the data can constrain: $DIC = \bar{D} + p_D$, where \bar{D} is a measure of fit (expected deviance), and p_D is a complexity measure, the *effective* number of parameters. The Akaike Information Criterion (AIC) [29; 30] is another well-known measure: $AIC = -2\mathcal{L}(\hat{\theta}) + 2k$, where $\hat{\theta}$ is the Maximum Likelihood Estimate (MLE) of θ and k is the number of parameters.

To compensate for small sample size n , a variant of AIC, termed AIC_c , is typically used:

$$AIC_c = -2\mathcal{L}(\hat{\theta}) + 2k + \frac{2k(k+1)}{(n-k-1)} \quad (8)$$

Another computationally more tractable approach is the Bayesian Information Criterion (BIC) [31]: $BIC = -2\mathcal{L}(\hat{\theta}) + k \log n$, where k is the number of *estimable* parameters, and n is the sample size (number of observations). BIC was developed as an approximation to the log marginal likelihood of a model, and therefore, the difference between two BIC estimates may be a good approximation to the natural log of the Bayes factor. Given equal priors for all competing models, choosing the model with the smallest BIC is equivalent to selecting the model with the maximum posterior probability. BIC assumes that the (parameters’) prior is the unit information prior (i.e., a multivariate normal prior with mean at the maximum likelihood estimate and variance equal to the expected information matrix for one observation).

Wagenmakers [32] shows that one can convert the BIC

metric to

$$BIC = n \log \frac{SSE}{SS_{total}} + k \log n,$$

where SSE is the sum of squares for the error term. In our experiments, we assume that the non-linear model is the “correct” model (or the null hypothesis H_0), and either the linear or qualitative models are the competing model (or alternative hypothesis H_1). Hence what we do is use BIC to compare the non-linear to each of the competing models.

Suppose that we obtain the BIC values for the alternative and the correct models, using the relevant SS terms. When computing $\Delta_{BIC} = BIC(H_1) - BIC(H_0)$, note that both the null (H_0) and the alternative hypothesis (H_1) models share the same SS_{total} term (both models attempt to explain the same collection of scores), although they differ with respect to SSE. The SS_{total} term common to both BIC values cancels out in computing Δ_{BIC} , producing

$$\Delta_{BIC} = n \log \frac{SSE_1}{SSE_0} + (k_1 - k_0) \log n, \quad (9)$$

where SSE_1 and SSE_0 are the sum of squares for the error terms in the alternative and the null hypothesis models, respectively.

5 Experimental Design

This section compares three tank benchmark models according to various model-selection measures. We adopt as our “correct” model the non-linear model. We will examine the fidelity and complexity tradeoffs of two simpler models over a selection of failure scenarios.

The diagnostic task will be to compute the fault state of the system, given an injected fault, which is one of (ξ_N, ξ_B, ξ_P) , denoting nominal blocked and passing valves, respectively. This translates to different tasks given the different models.

non-linear model estimate the true value of κ_1 given p_1 , which corresponds to a most-likely failure mode assignment of one of (ξ_N, ξ_B, ξ_P) .

linear model estimate the true value of κ_1 given p_1 , which corresponds to a most-likely failure mode assignment of one of (ξ_N, ξ_B, ξ_P) .

qualitative model estimate the failure mode assignment of one of (ξ_N, ξ_B, ξ_P) .

5.1 Alternative Models

This section describes the two alternative models that we compare to the non-linear model, a linear and a qualitative model.

Linear Model

We compare the non-linear model with a linearised version. We can perform this linearised process in a variety of ways [33]. In this simple tank example, we can perform the linearisation directly through replacement of non-linear and linear operators, as shown below.

Nominal Model We can linearise the the non-linear 3-tank model by replacing the non-linear sub-function $\sqrt{h_i - h_j}$ with the linear sub-function $\gamma_{ij}(h_i - h_j)$, where γ_{ij} is a parameter (to be estimated) governing the flow between tanks i and j . The linear model has 4 parameters, $\gamma_{12}, \gamma_{13}, \gamma_{23}, \gamma_3$.

Fault Model The fault model introduces a parameter β_i associated with κ_i , i.e., we replace κ_i with $\kappa_i(1 + \beta_i)$, $i = 1, 2, 3$, where $-1 \leq \beta_i \leq \frac{1}{\kappa_i} - 1$, $i = 1, 2, 3$. This model has 7 parameters, adding parameters $\beta_1, \beta_2, \beta_3$.

Qualitative Model

Nominal Model For the model we replace the non-linear sub-function $\sqrt{h_i - h_j}$ with the qualitative sub-function $M^+(h_i - h_j)$, where M^+ is the set of reasonable functions f such that $f' > 0$ on the interior of its domain [34].

The tank-heights are constrained to be non-negative, as are the parameters κ_i . As a consequence, we can discretize the h_i to take on values $\{+, 0\}$, which means that $M^+(h_i - h_j)$ can take on values $\{+, 0, -\}$. The domain for $\frac{dh_i}{dt}$ must be $\{+, 0, -\}$, since the qualitative version of q_0, Q is non-negative (domain of $\{+, 0\}$) and each $M^+(h_i - h_j)$ can take on values $\{+, 0, -\}$. We see that this model has no parameters to estimate.

Fault Model

The qualitative fault model has different M^+ functions for the modes where the valve is passing and blocked. We derive these functions as follows. From a qualitative perspective, the domain of β_i is $\{0, +\}$ for a passing valve, and $\{-, 0\}$ for a blocked valve. To create a new M^+ function for the cases of passing and blocked valve, we qualitatively apply these corresponding domains to the standard M^+ function with domain $\{-, 0, +\}$ to obtain fault-based M^+ functions: $M_P^+(h_i - h_j)$ denotes the M^+ function when the valve is passing, and $M_B^+(h_i - h_j)$ denotes the M^+ function when the valve is blocked.

5.2 Simulation Results

We have compared the simulation performance of the models under nominal and faulty conditions, considering faults to individual valves V_1, V_2 and V_3 , as well as double-fault combinations of the valves. In the following we present some plots for simulations of faults and fault-isolation for different model types.

Figure 2 shows the results from a single-fault scenario, where valve V_1 is stuck at 50% at $t = 250$, based on the non-linear model. The plot from this simulation show that at the time of the fault injection, the water level in tank T_1 starts increasing while the water level at tanks T_2 and T_3 start decreasing due to the lower inflow.

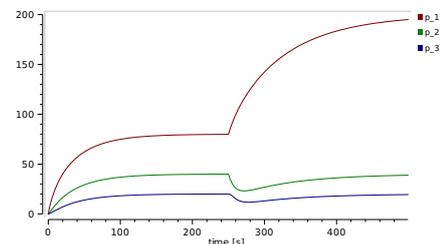


Figure 2: Simulation with non-linear model for the scenario of a fault in valve 1 at $t = 250$ s

Table 1 shows the simulation error-difference between the non-linear and linear models, for the nominal case and the faulty case (where valve 1 is faulted). Given that we measure the pressure levels for p_1, p_2 and p_3 every second, we use the difference in these outputs to identify the sum-of-squared-error (SSE) values for the simulations.

	p_1	p_2	p_3	Total
Nominal	2600.3	316.2	118.1	3034.6
V_1 -fault	2583.1	347.5	137.2	3067.8

Table 1: Data for SSE values for simulations using Non-linear and Linear representations, given two scenarios: nominal and faulty (valve V_1 at 50% after 250 s)

Figure 3 shows the results for diagnosing the V_1 -fault using the non-linear model. We can see that the diagnostic accuracy is high, as $P(V_1)$ converges to almost 1 with little time lag.

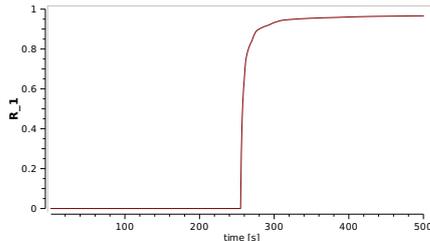


Figure 3: Simulation of fault isolation of fault in valve 1 with non-linear model. The figure depicts the probability of valve 1 being faulty.

In contrast, Figure 4 shows the diagnostic accuracy and isolation time with a linear model. First, note that there is a false-positive identified early in the simulation, and the model incorrectly identifies both valves 2 and 3 as being faulty. This linear model thus delivers both poor diagnostic accuracy (classification errors) and poor isolation time (there is a lag between when the fault occurs and when the model identifies the fault). After the fault injection at $t = 250$ [s], the predictive accuracy improves and the correct fault becomes the most likely fault.

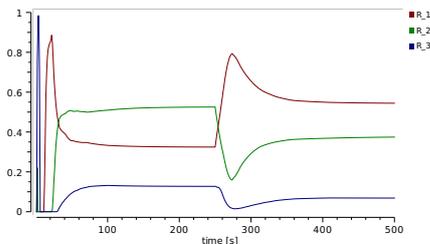


Figure 4: Simulation of fault isolation of fault in valve 1 with linear model. The figure depicts the probability of valves 1, 2 and 3 being faulty.

Figure 5 depicts the diagnostic performance with a mixed linear/non-linear model (T_1 is non-linear, while T_2 and T_3 are linear). The diagnostic accuracy is almost the same as that of the non-linear model (cf. Figure 3), except for a false-positive detection at the beginning of the scenario.

6 Experimental Results

This section describes our experimental results, summarising the data first and then discussing the implications of the results.

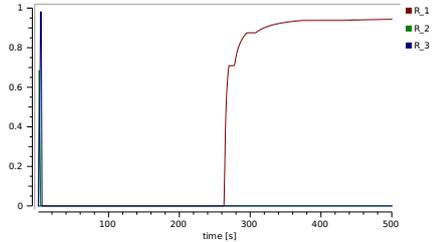


Figure 5: Simulation of fault isolation of fault in valve 1 with mixed non-linear/linear model (T_1 non-linear and both T_2 and T_3 linear). The figure depicts the probability of valves 1, 2 and 3 being faulty.

6.1 Model Comparisons

We have empirically compared the diagnostics performance of several multi-tank models. In our first set of experiments, we ran a simulation over 500 seconds, and induced a fault (valve V_1 at 50%) after 250 s. The model combinations involved a non-linear (NL) model, a model (denoted M) with tank T_1 being linear (and other tanks non-linear), a fully linear model (denoted L), and a Qualitative model (denoted Q).

To compare the relative performance of the models, we compute a measure of diagnostics error (or loss), using the difference between the true fault (which is known for each simulation) and the computed fault. We denote the true fault existing at time t using the pair (ω, t) ; the computed fault at time t is denoted using the pair $(\hat{\omega}, \hat{t})$. The inference system that we use, LNG [35], computes an uncertainty measure associated with each computed fault, denoted $P(\hat{\omega})$. Hence, we define a measure of diagnostics error over a time window $[0, T]$ using

$$\gamma_1^D = \sum_{t=0}^T \sum_{\xi \in \Xi} |P(\hat{\omega}_t) - \omega_t|, \quad (10)$$

where Ξ is the set of failure modes for the model, and ω_t denotes ω at time t .

Our second metric covers the fault latency, i.e., how quickly the model identifies the true fault (ω, t) : $\gamma_2 = t - \hat{t}$.

Table 2 summarises our results. The first columns compare the number of parameters for the different models, followed by comparisons of the error (γ_1) and the CPU-time (γ_2). The data show that the error (γ_1) does not grow very much as we increase model size, but it increases as we decrease model fidelity from non-linear through to qualitative models. In contrast, the CPU-time (a) increases as we increase model size, and (b) is proportional to model fidelity, i.e., it decreases as we decrease model fidelity from non-linear through to qualitative models.

In a second set of experiments, we focused on multiple model types for a 3-tank system, with simulations running over 50s, and we induced a fault (valve V_1 at 50%) after 25 s. The model combinations involved a non-linear (NL) model, a model with tank 3 linear (and other tanks non-linear), a model with tanks 2 and 3 linear and tank 1 non-linear, a fully linear model, and a qualitative model. Table 3 summarises our results.

The data show that, as model fidelity decreases, the error γ_1 increases significantly and the inference times γ_2 decrease modestly. If we examine the outputs from AIC_c , we see that the best model is the mixed model (T_3 -linear). BIC

Tanks		2	3	4
# Parameters	NL	7	9	11
	M	6	8	10
	L	5	7	9
	Q	2	3	4
γ_1	NL	242	242	242
	M	997	1076	1192
	L	1236	1288	1342
	Q	3859	3994	4261
γ_2	NL	10.59	23.7	39.5
	M	8.52	17.96	34.6
	L	6.11	10.57	32.0
	Q	4.64	7.31	26.4

Table 2: Data for 2-, 3-, and 4-tank models using Non-linear (NL), Mixed (M), Linear (L) and Qualitative (Q) representations

indicates the qualitative model as the best; it is worth noting that BIC typically will choose the simplest model.

	γ_1	γ_2	AIC_c	BIC
Non-Linear	0.97	23.7	29.45	43.7
T_3 -linear	3.12	17.96	26.77	42.9
T_2, T_3 -linear	21.96	13.21	31.12	39.56
Linear	77.43	10.57	35.76	37.55
Qualitative	304.41	9.74	43.01	29.13

Table 3: Data for 3-tank model, using Non-linear, Mixed, Linear and Qualitative representations, given a fault (valve V_1 at 50%) after 25 s

6.2 Discussion

Our results show that MBD is a complex task with several conflicting factors.

- The diagnosis error γ_1 is inversely proportional to model fidelity, given a fixed diagnosis task.
- The error γ_1 increases with fault cardinality.
- The CPU-time γ_2 increases with model size (i.e., number of tanks).

This article has introduced a framework that can be used to trade off the different factors governing MBD “accuracy”. We have shown how one can extend a set of information-theoretic metrics to combine these competing factors in diagnostics model selection. Further work is necessary to identify how best to extend the existing information-theoretic metrics to suit the needs of different diagnostics applications, as it is likely that the “best” model may be domain- and task-specific.

It is important to note that we conducted experiments with un-calibrated models, and we have ignored the cost of calibration in this article. The literature suggests that linear models can be calibrated to achieve good performance, although performance inferior to that of calibrated non-linear models. This class of qualitative models does not possess calibration factors, so calibration will not improve their performance.

7 Conclusions

This article has presented a framework for evaluating the competing properties of models, namely fidelity and computational complexity. We have argued that model performance needs to be evaluated over a range of future observations, and hence we need a framework that considers the *expected performance*. As such, information-theoretic methods are well suited.

We have proposed some information-theoretic metrics for MBD model evaluation, and conducted some preliminary experiments to show how these metrics may be applied. This work thus constitutes *a start* to a full analysis of model performance. Our intention is to initiate a more formal analysis of modeling and model evaluation, since there is no framework in existence for this task. Further, the experiments are only preliminary, and are meant to demonstrate how a framework can be applied to model comparison and evaluation.

Significant work remains to be done, on a range of fronts. In particular, a thorough empirical investigation is needed on diagnostics modeling. Second, the real-world utility of our proposed framework needs to be determined. Third, a theoretical study of the issues of mode-based parameter estimation and its use for MBD is necessary.

References

- [1] George EP Box. Statistics and science. *J Am Stat Assoc*, 71:791–799, 1976.
- [2] Peter Struss. What’s in SD? Towards a theory of modeling for diagnosis. *Readings in model-based diagnosis*, pages 419–449, 1992.
- [3] Peter Struss. Qualitative modeling of physical systems in AI research. In *Artificial Intelligence and Symbolic Mathematical Computing*, pages 20–49. Springer, 1993.
- [4] Nuno Belard, Yannick Pencolé, and Michel Combacau. Defining and exploring properties in diagnostic systems. *System*, 1:R2, 2010.
- [5] Alexander Feldman, Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, and David Garcia. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*, 1:24, 2010.
- [6] Steven D Eppinger, Nitin R Joglekar, Alison Olechowski, and Terence Teo. Improving the systems engineering process with multilevel analysis of interactions. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(04):323–337, 2014.
- [7] Sanjay S Joshi and Gregory W Neat. Lessons learned from multiple fidelity modeling of ground interferometer testbeds. In *Astronomical Telescopes & Instrumentation*, pages 128–138. International Society for Optics and Photonics, 1998.
- [8] Roxanne A Moore, David A Romero, and Christiaan JJ Paredis. A rational design approach to gaussian process modeling for variable fidelity models. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 727–740. American Society of Mechanical Engineers, 2011.

- [9] Peter D Hanlon and Peter S Maybeck. Multiple-model adaptive estimation using a residual correlation Kalman filter bank. *Aerospace and Electronic Systems, IEEE Transactions on*, 36(2):393–406, 2000.
- [10] Redouane Hallouzi, Michel Verhaegen, Robert Babuška, and Stoyan Kanev. Model weight and state estimation for multiple model systems applied to fault detection and identification. In *IFAC Symposium on System Identification (SYSID), Newcastle, Australia*, 2006.
- [11] Amardeep Singh, Afshin Izadian, and Sohel Anwar. Fault diagnosis of Li-Ion batteries using multiple-model adaptive estimation. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, pages 3524–3529. IEEE, 2013.
- [12] Amardeep Singh Sidhu, Afshin Izadian, and Sohel Anwar. Nonlinear Model Based Fault Detection of Lithium Ion Battery Using Multiple Model Adaptive Estimation. In *World Congress*, volume 19, pages 8546–8551, 2014.
- [13] Aki Vehtari, Janne Ojanen, et al. A survey of bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 6:142–228, 2012.
- [14] Athanasios C Antoulas, Danny C Sorensen, and Serkan Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary mathematics*, 280:193–220, 2001.
- [15] Alexander Feldman, Gregory M Provan, and Arjan JC van Gemund. Computing observation vectors for max-fault min-cardinality diagnoses. In *AAAI*, pages 919–924, 2008.
- [16] Amardeep Singh, Afshin Izadian, and Sohel Anwar. Nonlinear model based fault detection of lithium ion battery using multiple model adaptive estimation. In *19th IFAC World Congress, Cape Town, South Africa*, 2014.
- [17] Youmin Zhan and Jin Jiang. An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 4, pages 3593–3598. IEEE, 1999.
- [18] Peter Struss and Oskar Dressler. "physical negation" integrating fault models into the general diagnostic engine. In *IJCAI*, volume 89, pages 1318–1323, 1989.
- [19] Johan De Kleer, Alan K Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2):197–222, 1992.
- [20] Elizabeth H Keating, John Doherty, Jasper A Vrugt, and Qinjun Kang. Optimization and uncertainty assessment of strongly nonlinear groundwater models with high parameter dimensionality. *Water Resources Research*, 46(10), 2010.
- [21] Saket Pande, Mac McKee, and Luis A Bastidas. Complexity-based robust hydrologic prediction. *Water resources research*, 45(10), 2009.
- [22] G Schoups, NC Van de Giesen, and HHG Savenije. Model complexity control for hydrologic prediction. *Water Resources Research*, 44(12), 2008.
- [23] S Pande, L Arkesteijn, HHG Savenije, and LA Bastidas. Hydrological model parameter dimensionality is a weak measure of prediction uncertainty. *Natural Hazards and Earth System Sciences Discussions*, 11, 2014, 2014.
- [24] Martin Kunz, Roberto Trotta, and David R Parkinson. Measuring the effective complexity of cosmological models. *Physical Review D*, 74(2):023503, 2006.
- [25] Gregory M Provan and Jun Wang. Automated benchmark model generators for model-based diagnostic inference. In *IJCAI*, pages 513–518, 2007.
- [26] David J Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):583–639, 2002.
- [27] Jing Du. The "weight" of models and complexity. *Complexity*, 2014.
- [28] Jasper A Vrugt and Bruce A Robinson. Treatment of uncertainty using ensemble methods: Comparison of sequential data assimilation and bayesian model averaging. *Water Resources Research*, 43(1), 2007.
- [29] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.
- [30] Hirotugu Akaike. Likelihood of a model and information criteria. *Journal of econometrics*, 16(1):3–14, 1981.
- [31] G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6:461–466, 1978.
- [32] Eric-Jan Wagenmakers. A practical solution to the pervasive problems of p values. *Psychonomic bulletin & review*, 14(5):779–804, 2007.
- [33] Pol D Spanos. *Linearization techniques for non-linear dynamical systems*. PhD thesis, California Institute of Technology, 1977.
- [34] Benjamin Kuipers and Karl Åström. The composition and validation of heterogeneous control laws. *Automatica*, 30(2):233–249, 1994.
- [35] Alexander Feldman, Helena Vicente de Castro, Arjan van Gemund, and Gregory Provan. Model-based diagnostic decision-support system for satellites. In *Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, USA*, pages 1–14, March 2013.

Posters

A General Process Model: Application to Unanticipated Fault Diagnosis

Jiongqi WANG¹, Zhangming HE², Haiyin ZHOU³ and Shuxing LI¹

¹ College of Science, National University of Defense Technology, Changsha, Hunan, P. R. China

email: wjq_gfkd@163.com

² Institute for Automatic Control and Complex Systems, University of Duisburg-Essen, Duisburg, Germany

email: hezhangming2008@sina.com

² Beijing Institute of Control Engineering, Beijing, P. R. China

email: gfkd_zhy@sina.com

¹ College of Science, National University of Defense Technology, Changsha, Hunan, P. R. China

email: lishuxingok@163.com

Abstract

The improvement of the detection and diagnosis capability for the unanticipated fault is a tendency in the research and application of fault diagnosis. In this paper, some notions and the basic principles for the unanticipated fault detection and diagnosis are given. A general process model applied to the diagnosis for the unanticipated fault is designed, by adopting a three-layer progressive structure, which is comprised of an inherent detection layer, an unanticipated isolation layer and an unanticipated recognition layer. Several key problems in the general process model are analyzed. The model and methods proposed in this paper are driven by pure data and they can detect and diagnose the unanticipated fault. The approach is evaluated by using an example of a satellite's attitude control system, and excellent results have been obtained.

1 Introduction

At present, in the research field of fault diagnosis, a great majority of methods proposed are based on the premise of a perfect fault pattern database. The treatment on the fault detection and diagnosis are carried out for *anticipated fault* (AF) [1-3]. However, due to the high complexity and uncertainty of the technical structure, the process environment and the working state of the system etc, the occurrence of some faults which cannot be anticipated in advance (*Unanticipated Fault*, UF) is inevitable in actual work [4]. The UF is not included in the anticipated fault database, and the occurrence of the UF affects normal operation of the system and even possibly leads to thorough failure of the system. The improvement of *unanticipated fault detection and diagnosis* (UFDD) capability is a difficult issue, as well as a developing direction in the research and application for the fault diagnosis [5-8].

In retrospect to the existing researches, rather little attention has been paid to research UF detection and diagnosis. Therefore, no mature solve scheme has been shaped for either the problem itself or the technical realization [9-12]. Most research on the UF focus on the recognition and the match between different patterns based on the known fault pattern database [13-14]. For example, Tom Brotherton and Tom Johnson (2001) [15] proposed a neural network anomaly detector, which was essentially a single neural network classifier and could not identify the UF. Z. H. Duan

(2006) [16] proposed that the UF diagnosis was carried out by utilizing particle filter for incomplete patterns. As a transmission mechanism of the UF could not be obtained in advance, the UF diagnosis could not be realized based on model inference. George Vachtsevanos etc. (2008) [17] proposed an UF robust detection method, however, the isolation on the UF could not be realized. Furthermore, Z. M He (2012) [18] proposed a one-class principal component analysis (OC-PCA) method, which could only be used for processing the system with stable data in a normal pattern, and did not relate to the UF diagnosis at all. The majority of currently published articles involve only UF detection. However, the fault isolation between the UF and the AF as well as the recognition (i.e. identification) of the UF has not yet been performed.

For actual system, some impacts such as nonlinearity, uncertainty and external interference are inevitable in its actual operation, which will result difficulties in setting up a precise model for the system. Consequently, the application of the methods for fault detection and diagnosis based on model inference will be very limited [19-20]. With the development of sensor technology, the input and output data or the system's status under real-time monitor is easier to obtain. The data are redundant, real-time and reliable. As a result, the fault diagnosis ideology of extracting data instead of establishing a system's model will play a positive role.

This paper proposes a data-driven fault diagnosis method for UF. Combined with the fault diagnosis process, a *general process model* (GPM) is advanced, which is comprised of an *inherent detection layer* (IDL), an *unanticipated isolation layer* (UIL) and an *unanticipated recognition layer* (URL). Firstly, according to different characteristics of the monitoring data, the corresponding residual statistics are built and a detection criterion of the IDL is provided for fault detection. Secondly, the statistic of angle similarity is constructed on the basis of the fault feature direction, the isolation between the UF and the AF is realized in the UIL. Finally, in the URL, by the adoption of the contribution factor, the UF is recognized. The method, as a fault diagnosis method driven by pure data, is capable of carrying out detection, isolation and recognition for the UF.

The paper is organized as follows. In Section 2, some notions and the basic principles for UF and UFDD are discussed. A three-layer GPM for UFDD is introduced in Section 3. Sections 4 analyzes some key problems in the GPM and advances the corresponding solutions. In Section

5, performance evaluation of the proposed GPM and methods for the satellite's attitude control system is presented. Conclusions are drawn in Section 6.

2 Notions and Basic Principles for UFDD

2.1 Notion of UF

The fault can be divided into the *anticipated fault* (AF) and the *unanticipated fault* (UF).

Explanation 1: *Anticipated fault* (AF) is the fault which has been recognized by people, existing in the fault pattern database with the relevant monitoring data and the processing strategy.

Explanation 2: *Unanticipated fault* (UF) is the fault which lacks prior knowledge without any fault samples or with few fault data. UF does not exist in the fault pattern database, and the corresponding elimination strategy for it has not been detected.

A perfect fault pattern database should be a set including all AF patterns and UF patterns. However, due to some objective reasons, the acquisition of the perfect fault pattern database is extremely difficult. The AF rarely occurs, and most of faults occurs in the actual working process are UF [21]. At present, to detect the UF and moreover to diagnose the UF is one of the most difficult issues in fault diagnosis region, and it is also a great challenge for fault diagnosis technology.

2.2 Notion for UF Detection

Explanation 3: UF detection is a process for judging whether UF occurs.

The tasks of UF detection and AF detection are different. The two methods apply previous normal monitoring data to train a discriminator, and then the current monitoring data is used as the testing data to be input into the discriminator to judge whether the current status is a fault. However, the UF detection is carried out after the completion of fault detection, and the fault is further judged whether to be UF. Obviously, for AF detection, all faults are always assumed to be anticipated. Consequently, if the UF occurs, it will be misjudged as a certain anticipated fault.

2.3 Notion for UF Diagnosis

Explanation 4: UF diagnosis is a process of determining whether the UF occur (i.e. UF detection). In addition, the UF diagnosis further includes the isolation and the recognition of the UF after the UF detection is completed.

Compared with the AF diagnosis, due to lack of prior knowledge of the UF, the mapping relationship from fault data to fault part (essentially, the fault pattern is a function between fault data and fault part) cannot be found. Therefore, the key for UF diagnosis is to quickly establish a cognition process. The cognition comprises the recognition of superficial data characteristics or the mapping recognition from data to a physical layer. Based on a fault diagnosis method driven by pure data, this paper focuses on the recognition of superficial data characteristics.

3 General Process Model (GPM) for UFDD

By combining the notion and basic principles of the UF and the UFDD, this paper proposes a multi-layer general pro-

cess model (GPM) for UF diagnosis on the basis of pure data-driven method. The structure of GPM is shown in Figure 1. The first layer is the IDL, which establishes a detection discriminator for fault detection; the second layer is the UIL, which applies the detection residual to establish a fault feature direction so as to build an isolation discriminator to realize the isolation of the AF and the UF; the third layer is the URL, which applies a contribution factor to analyze the variant which is most relevant to the current UF and to realize the fault recognition based on superficial data characteristics.

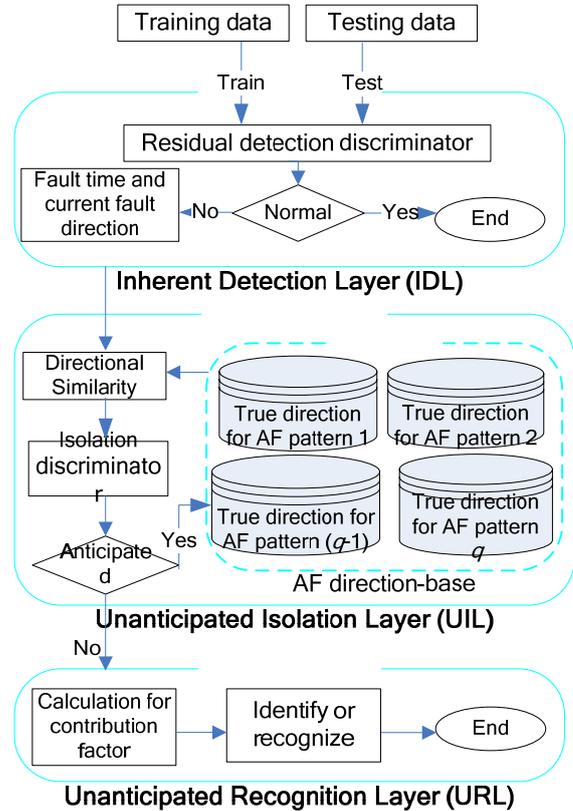


Figure 1 The GPM for UFDD

3.1 Inherent Detection Layer (IDL)

The first issue that a diagnosis system faces is to carry out normal/abnormal recognition for a feature vector of the monitoring data. The task of the IDL is to determine whether the monitoring data is normal or abnormal. The detection discriminator can be used for reflecting the characteristics of the normal system. In a given threshold, the testing data is inputted to the detection discriminator for judging whether the fault exists. If a value of the discriminator is smaller than the given threshold, the system is thought to be normal; otherwise, a fault is thought to occur. Meanwhile the occurrence time (*Fault time*) and the feature direction of the fault (*Current fault direction*) should be determined, and the testing data is presented to the UIL.

Essentially, the IDL is a single discriminator, which can be applied to catch the characteristics of the system in a normal pattern as well as to complete the detection and discrimination of the testing data. Two key problems are involved, the first is the residual generation and the second is the residual evaluation. The specific techniques can be seen in Section 4.1.

3.2 Unanticipated Isolation Layer (UIL)

The task of the UIL is to finish the isolation between the UF and AF. After detected, the current fault shall be judged whether to be the AF or the UF. If it is, the current fault will be classified as some sort of AF. All AF patterns are saved in the pattern database of AF. The isolation discriminator matches the feature of the current fault pattern with all those of the AF patterns successively, so as to realize the isolation between the UF and AF. If the feature of the current fault cannot be matched with any AF pattern, it indicates that the UF occurs. The testing data is presented to the URL. The key problem of the UIL lies in the establishment of an isolator and the design of an isolation criterion. The specific techniques can be seen in **Section 4.2**.

3.3 Unanticipated Recognition Layer (URL)

The task of the URL is to perform online learning and analysis for the UF data, so as to generate the fault pattern. The function of the URL is to learn and summarize the pattern found in unknown pattern. As it is different from the AF, it is difficult to find the mapping relationship from the fault data to the fault part for the UF. Therefore, the key point of recognition lies in establishing the corresponding relationship between the data and the unknown fault. Due to insufficient recognition on the UF and lack of historical information and prior knowledge, it is usually more difficult to establish the mapping relationship on the physical layer. The key point of this paper is to analyze the UF recognition based on the superficial data layer. According to contribution factor, the variant which is mostly relevant to the current UF can be found, so that the UF recognition is finished. The specific techniques can be seen in **Section 4.3**.

4 Some Key Problems in GPM

In the above section, a basic framework of the UF diagnosis is provided. The task of the UF diagnosis is to detect, isolate and recognize the UF. The detection is a starting point of fault diagnosis, and the target of the fault detection is to judge whether the UF occurs; the isolation is the core of fault diagnosis; and the recognition is a terminal point of fault diagnosis. Additionally, the recognition is also the starting point of fault-tolerant control (fault processing). The specific techniques on detecting, isolating and recognizing the UF can be seen below.

4.1 Detection Statistic Construction

Just as **Section 3** shows, the basic task of the IDL is to judge whether the testing data is normal. If it is a fault, simultaneously the occurrence time and the feature direction of the fault shall be determined. The key point of the IDL lies in the detection residual generation as well as the residual evaluation. The detection statistic is established according to the residual, and the fault detection is performed according to the given criterion. For different monitoring data, different residual generation approaches exist, including simple T^2 detection [18, 22], baseline data smoothing detection [23], and time-series modeling and predicting detection [24-25].

The characteristics of the monitoring system and monitoring data can be applied to select the corresponding detection method. The simple T^2 statistic detection is applied to a stable data [22]. The baseline data smoothing detection

is suitable for the system capable of obtaining the baseline data, its calculation amount is small, the detection speed is fast, and the detection effect is the best [23]. The time-series modeling prediction is suitable for the system with continuous output and without input; it is also suitable for iteration update of the pattern, while the defect is that the prediction time is short [25].

In practical application, the characteristics of the monitoring system and the monitoring data can be applied to select the corresponding detection method.

Besides, for the three methods analyzed above, only the characteristics of data output are considered. However, for some systems (such as the satellite's attitude control system), the object of the fault detection always comprises control input as well as measuring output, and the control input has a certain responding relationship with the measuring output. In the situation where there is no baseline training data, an input-output system identification method is needed to search a model structure for the system, and thus the fault detection both on control input and measuring output will be performed in the IDL.

If we assume that $(U_{n-1}, Y_{n-1}) \in (R^{(n-1) \times p}, R^{(n-1) \times m})$ are respectively as system input and system output before the n th time period, take them as the training data and make $(u_n, y_n) \in (R^{1 \times p}, R^{1 \times m})$ as the current testing data. The train purpose is to find the model structure of the system, usually with the rule as follows

$$\min_f \|Y_{n-1} - f(U_{n-1})\| \quad (1)$$

Let $\hat{Y}_{n-1} = f(U_{n-1})$ is the tendency term, $\tilde{Y}_{n-1} = Y_{n-1} - \hat{Y}_{n-1} = Y_{n-1} - f(U_{n-1})$ is the residual term; $\hat{y}_n = f(u_n, U_{n-1}, Y_{n-1})^T$ is one-step prediction, and $r_n = y_n - \hat{y}_n$ is the prediction residual, then the key point for the minimum problem in (1) is to construct the function f between the system input and system output.

If a mathematical model can be obtained for the system equation by the physical mechanism, the estimation of f can be converted into the parameter estimation (**Gray-Box Model**); and if there is no physical background, f can be estimated only according to the experiment and the system identification (**Black-Box Model**). Common linear black box models comprise an autoregression model (AR Model) with external input, an autoregressive moving average model (ARMA Model) with external input, an output error model (OE Model), a Box-Jenkins model (BJ Model) and a prediction error minimized model (PEM Model); and common nonlinear black box models comprise a nonlinear autoregression moving average model (NLARMA Model) and a nonlinear Hammerstein-Wiener model (NLHW Model) [26-29] with external input.

After obtaining the prediction residual, the detection statistics are as below:

$$T^2(y_n) = r_n^T \text{cov}(\tilde{Y})^{-1} r_n \quad (2)$$

where $\text{cov}(\tilde{Y})$ is the covariance of the residual term \tilde{Y} , and a judging threshold is set to be

$$T_\alpha^2 = \frac{m(n)(n-2)}{(n-1)(n-1-m)} F_{(1-\alpha)}(m, n-1-m) \quad (3)$$

where $F_{(1-\alpha)}(m, n-1-m)$ indicates a quantile of F distribution function when a significance level is α , the degree of freedom is $(m, n-1-m)$.

If $T^2(y_n) > T_\alpha^2$, y_{n-1} is considered as the fault point. However, a false alarm is inevitable because of noise, thus we need a more reliable criterion for detection as follows.

Criterion 1: If $T^2(y_n) > T_\alpha^2$ holds continuously for W times, then the fault has really happened, where W is called *time threshold*. The W -th alarm time is considered as *the fault time* (t_f) (i.e. the occurrence time of the fault) and the residual r of the fault time is called the *current fault direction* or *current direction* (i.e. the feature direction of the fault).

The detection statistic threshold is decided by Equation (3). The time threshold should not be too large (usually 2 to 4) to avoid any false alarms. A larger *time threshold* makes a more reliable decision, but it will cause some detection delay which will cause harm to the system. *Current fault direction* is the key information of each fault, and it is the base for the isolation fault. According to **Criterion 1**, the current fault is detectable if and only if

$$\|r_n\| > \sqrt{T_\alpha^2 (r_n^T \text{cov}(\tilde{Y})^{-1} r_n)^{-1}} \quad (4)$$

In the IDL, the fault detection is realized by the adoption of the input-output system identification method. Moreover, the occurrence time and feature direction of the fault can also be obtained.

Obviously, the input-output system identification method is provided with all the advantages of the time-series modeling prediction method. It is particularly suitable for the system with discontinuous input and discontinuous output at the same time, its defect is that the calculation amount is large, and the iteration process is relatively difficult.

4.2 Directional Similarity and Isolation Criterion

The basic task of the UIL is to utilize the feature direction of the fault obtained in the IDL to establish the isolation discriminator, and then to realize the isolation between the AF and the UF. The key point lies in the isolator establishment. Here the concept of direction similarity is induced, and a fault isolation criterion is given. In **Criterion 1**, the definition of *current fault direction* or *current direction* (i.e. the feature direction of a fault) is given. We adopt the true fault feature direction as defined below to be the fault's pattern characteristics on superficial data layer.

Explanation 5: *True (fault) direction* of a fault pattern is defined as the unified mean of all possible current fault directions from the same pattern.

The relationship between the current directions and the true direction is just like that between discrete random variable and its expectation. It is easy to understand that

$$\xi = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n r_i / \left\| \frac{1}{n} \sum_{i=1}^n r_i \right\|_2 \quad (5)$$

$$r = \|r\| \xi + \varepsilon \quad (6)$$

where $\{r_i\}_{i=1}^n$ are all possible current directions from the same pattern, and ε is the noise and $\|r\|$ is the magnitude of the current direction.

It is shown in Figure 2 that there are two opposite true directions for each fault pattern, e.g. the true direction, ξ_1 , is in the center of a symmetric cone, around which are the

current directions from the same pattern. ξ_2 is another true direction, corresponding to another fault pattern. The origin of the coordinates can be regarded as the true direction for the normal pattern.

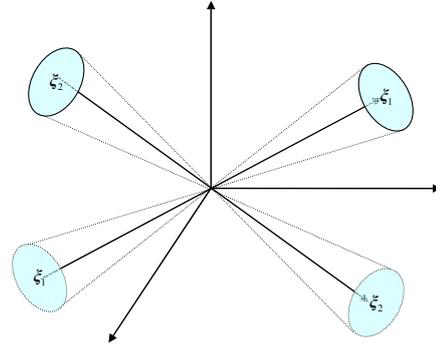


Figure 2 True detections and current directions

Denote $\theta(r, \xi)$ is the angle between the current direction and the true direction, $Ddisc(r, \xi) = 1 - |\cos(\theta(r, \xi))|$ is called the *directional discrepancy* between them. We can find that if they are from the same pattern, $Ddisc(r, \xi)$ will be small, otherwise, it will be large.

Suppose that $\varepsilon \sim N(\mathbf{0}, \mathbf{\Omega})$, the current direction is $r = \varepsilon + \|r\| \xi$, and $\{\xi_i\}_{i=1}^q$ is all anticipated true directions, and $\xi_{i_0} = \arg \min_{\xi} \{1 - |\cos(r, \xi)|\}_{i=1}^q$, then the isolation statistic is given as follows

$$Iso(r) = \frac{\|r\| (1 - |\cos(r, \xi_{i_0})|)}{\sqrt{\xi_{i_0}^T \mathbf{\Omega} \xi_{i_0}}} \quad (7)$$

Theorem 1: If $Iso(r)$ is defined in Equation (7), then

$$Iso(r) \sim N(0, 1) \quad (8)$$

Proof: Suppose that the current direction is $r = \varepsilon + \|r\| \xi$, where ξ is the true direction and ε is the observation noise, and $\varepsilon \sim N(\mathbf{0}, \mathbf{\Omega})$. According to **Explanation 5** we have $\|\xi\| = 1$. If $\cos(r, \xi) \geq 0$, we can approximately obtain that

$$\cos(r, \xi) = \frac{\xi^T r}{\|\xi\| \|r\|} = \frac{\xi^T \varepsilon}{\|r\|} + 1 \sim N(1, \|r\|^{-2} \xi^T \mathbf{\Omega} \xi) \quad (9)$$

i.e. $\cos(r, \xi)$ satisfies truncated normal distribution.

Thus

$$\|r\| (1 - \cos(\xi_{i_0}, r)) \sim N(0, \xi_{i_0}^T \mathbf{\Omega} \xi_{i_0}) \quad (10)$$

Similarly, if $\cos(r, \xi) < 0$, we can prove that

$$\|r\| (1 + \cos(\xi, r)) \sim N(0, \xi^T \mathbf{\Omega} \xi) \quad (11)$$

According to Equation (10) and Equation (11), we obtain

$$\|r\| (1 - |\cos(\xi, r)|) \sim N(0, \xi^T \mathbf{\Omega} \xi) \quad (12)$$

Then

$$Iso(r) = \frac{\|r\| (1 - |\cos(r, \xi_{i_0})|)}{\sqrt{\xi_{i_0}^T \mathbf{\Omega} \xi_{i_0}}} \sim N(0, 1) \quad (13)$$

and thus the theorem is proved. Therefore, the threshold for $Iso(r)$ is $\Phi_{(1-\alpha)}$, where α is the significance level, and Φ is the inverse of the normal cumulative distribution function. We provide the isolation criterion as follows.

Criterion 2: If $Iso(\mathbf{r}) > \Phi_{1-\alpha}$ holds true, the current fault is unanticipated; otherwise, it is anticipated.

Criterion 2 indicates that UF with too small a magnitude cannot be isolated. If the current fault is unanticipated, a new fault pattern is found and the unified current direction is regarded as its true direction. If the current fault is anticipated, then the current direction should be added to the corresponding AF direction database in UIL of the GPM, and the true direction shall be updated.

4.3 Calculation for Contribution Factor

The basic task of the URL is to carry out online learning and analysis for UF data. The key point of recognition or identification is to establish the corresponding relationship from the monitoring data to the unknown fault or the characteristics of the unknown fault. The UF diagnosis discussed in this paper is an approach driven by pure data, thus the characteristic recognition on the data layer is more focused. According to the contribution factor, the variant which is most relevant to the current UF can be found, and then the UF recognition is completed.

Known from **Criterion 1** that after the residual detection statistic is established, if $T^2(y_n) > T_\alpha^2$, it is thought that a fault occurs at time period $n-1$. For the system with the control input and measure output, firstly a residual covariance matrix \mathbf{R} (i.e. $\text{cov}(\tilde{\mathbf{Y}})$ in Equation (2)) is subjected to the singular value decomposition, which is

$$\mathbf{R} = \mathbf{P}^T \text{diag}(\boldsymbol{\lambda}) \mathbf{P} \quad (14)$$

where $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$, $\mathbf{P} = (\mathbf{p}_1, \dots, \mathbf{p}_m)$, \mathbf{p}_i indicates the i th column of \mathbf{P} , and p_{ji} indicates the j th component of \mathbf{p}_i . Let $t_i = \mathbf{r}^T \mathbf{p}_i$, and r_j indicates the j th component of the current fault feature direction \mathbf{r} , where $1 \leq j \leq m$.

Explanation 6: The contribution factor of the j th variant to the current fault feature direction \mathbf{r} is

$$\text{Cont}(j) = \sum_{i=1}^m (t_i r_j p_{ji} / \lambda_i) \quad (15)$$

From the aspect of characteristic recognition in the data layer, the variant with the largest contribution factor is the fault variant. If it is a sensor fault, the sensor corresponding to the variant with the largest contribution factor is the sensor hardware with the fault.

5 Simulation and Performance Evaluation

The effectiveness of the proposed GPM and the corresponding UF fault detection, isolation and recognition method are demonstrated in this section through a satellite's attitude control system model.

5.1 Input and Output of Satellite Control System

The satellite's attitude control system is a main part of a satellite, which consists of four main parts: *a satellite body*, *a controller*, *an execution mechanism* and *a measuring mechanism* [30].

As the complexity of the satellite's attitude control system, faults particularly for the measuring mechanism and the execution mechanism occur rather frequently.

Here on consideration of the monitoring data for the satellite's attitude control system. The monitoring data are provided by *China Aerospace Science and Technology Corporation (CASA)*.

The monitoring data comprises of not only the output data of the measuring mechanism, but also the control input of the execution mechanism. The dimension of the data output by the measuring mechanism is $m = 7$, The dimension of the data input by the execution mechanism is $p = 4$, which can be seen in Table 1. There are altogether 10 batches of monitoring data, which can be seen in Table 2. The first batch is the normal data, and the normal pattern data is discontinuous and unstable (Figure 3). The subsequent 9 batches are used for testing, and different fault patterns (a sudden-change fault, a gradual-change fault and so on) are given. In Figure 3, the comparison of the monitoring data in the fault with drift-increasing of gyro at roll axis and the normal pattern is given. The time of each batch of data is 45000s-48000s; each piece data is collected per second, and the data length $n = 3000$.

Additionally, the public parameters used in the simulation are assigned as follows: The significance level $\alpha = 0.01$ and the time threshold defined in **Criterion 1** is $W=3$.

Table 1 Data explain of attitude control system

Variable subscript		Code	Sensor
1	Input	Wheel1	Output of the first momentum wheel
2		Wheel2	Output of the second momentum wheel
3		Wheel3	Output of the third momentum wheel
4		Wheel4	Output of the fourth momentum wheel
1	Output	EarthPhi	Output of earth sensor at roll axis
2		EarthTheta	Output of earth sensor at pitch axis
3		SunPhi	Output of sun sensor at roll axis
4		SunTheta	Output of sun sensor at pitch axis
5		GeoPhi	Output of gyro at roll axis
6		GeoTheta	Output of gyro at pitch axis
7		GeoPsi	Output of gyro at yaw axis

Table 2 Batch number of monitoring data

Batch number	Data description	Fault time
1	Normal data	Null
2	Sudden-change fault data of earth sensor at roll axis	46000s
3	Gradual-change fault data of earth sensor at roll axis	46000s
4	Sudden-change fault data of earth sensor at pitch axis	46000s
5	Gradual-change fault data of earth sensor at pitch axis	46000s
6	Loss fault data of sun sensor at roll axis	46000s
7	Loss fault data of sun sensor at pitch axis	46000s
8	Drift-increasing fault data of gyro at roll axis	46000s
9	Drift-increasing fault data of gyro at pitch axis	46000s
10	Drift-increasing fault data of gyro at yaw axis	46000s

5.2 Performance Evaluation

The monitoring data are relatively more complex, comprising of the output data of the measuring mechanism and the control input of the execution mechanism (seen in Table 1). The normal pattern data is discontinuous and unstable (seen in Figure 3), and the fault pattern is diversified (with sudden-change fault, gradual-change fault and so on). Therefore, the normal pattern data is difficult to be discriminated from the fault pattern data (seen from Figure 3).

With the input-output system identification method, the Hammerstein-Wiener model (NLHW) is adopted. Equation (1) is optimized, and the responding function f between the input and output is estimated. Similarly, for the same data (Drift-increasing fault data of gyro at roll axis (the batch number is 8) in Table 2), the detection result of the IDL is

given in Figure 4, which can be seen that the fault detection is timely, the detection effect is remarkable, and 4s detec-

tion is delayed caused by the time threshold, $W = 3$.

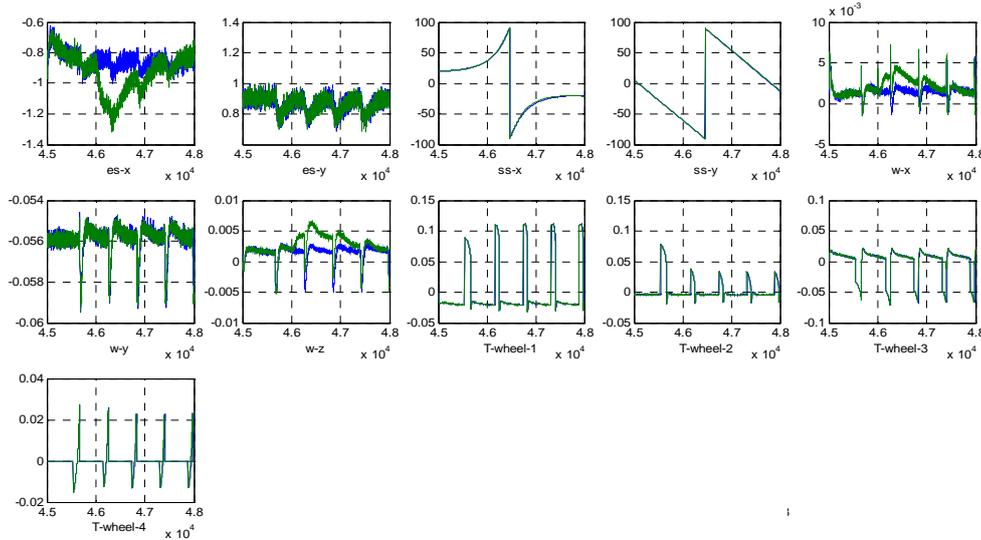


Figure 3 Drift-increasing fault of gyro at roll axis (Blue line shows the output in the normal pattern while green line shows the output in the fault patter

By adopting the input-output system identification method, the detection results in the IDL for the data in Table 2 are shown in Table 3. The fault detection is timely, and the detection effect is more obvious (both of the FAP (false alarm probability) and the MAP (missing alarm probability) are much lower).

In the IDL, the fault detection can be realized, and the fault time and the current fault direction are also determined. In the UIL, **Criterion 2** is adopted to realize the isolation between the UF and the AF. In the initial stage, the AF pattern is assumed to be empty, therefore, when the second batch of data in Table 2 is filled into the UIL, the detected fault must be the UF, and then the isolation result is transferred into the URL. When the third batch of data in Table 2 is filled into the IDL, the fault time is that $t = 1001s$, the statistic of the directional similarity is $\|r\|(1 - |\cos(r, \xi_1)|) / \sqrt{\xi_1^T R \xi_1} = 7.3179$, and the isolation threshold of the UF is also $\Phi_{0.99} = 2.3263$. Obviously $\|r\|(1 - |\cos(r, \xi_1)|) / \sqrt{\xi_1^T R \xi_1} > \Phi_{0.99}$, the current fault pattern is different from the first fault pattern, and an UF occurs. Then the UF is transferred into the URL. The fault isolation result for all the tested data in Table 2 can be seen in Table 4. From Table 4, we know that the isolator with the fault fea-

ture direction and the direction similarity is valid, and the isolation between the UF and the AF can be truly realized.

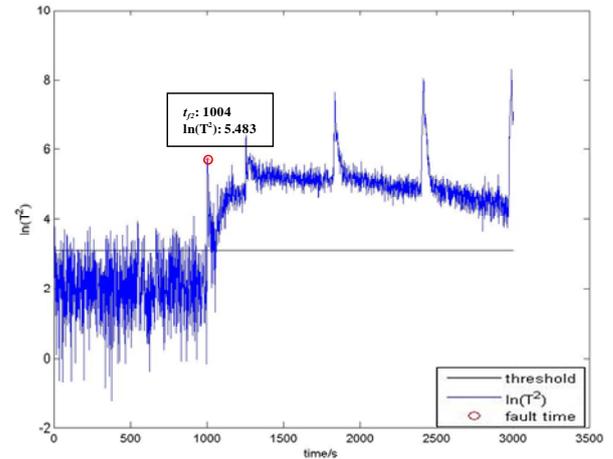


Figure 4 The detection result (with input-output system identification method) for drift-increasing fault data of gyro at roll axis

Table 3 Unanticipated fault diagnosis—IDL

Inherent Detection Layer (IDL)											
Batch number	Normal or Fault	FAP (%)	MAP (%)	Fault time (s)	Current fault direction						
					0	0	0	0	0	0	0
1	N	5			0	0	0	0	0	0	0
2	F	3	2	1000+2	0.9876	-0.0042	0.041	-0.053	0.0453	-0.1342	0.0678
3	F	4	1	1000+1	-0.9997	0.0005	-0.034	0.049	0.0049	-0.0036	0.0222
4	F	5	1	1000+2	-0.1510	-0.9747	-0.0097	0.0105	0.0442	-0.1550	0.0345
5	F	4	1	1000+2	-0.0018	1.0000	0.0007	0.0006	-0.0009	-0.0022	-0.0077
6	F	5	1	1000+2	0.0086	-0.0093	-0.9752	0.0046	-0.0007	0.0003	0.0008
7	F	3	2	1000+3	-0.0067	0.0052	0.0016	-0.9925	-0.1553	0.0028	-0.0016
8	F	5	1	1000+4	-0.0769	0.0051	0.0037	0.0018	0.9682	-0.0139	-0.0549
9	F	3	1	1000+2	-0.0742	0.0215	-0.0029	0.0016	0.0454	-0.9968	0.0447
10	F	3	1	1000+2	0.0627	-0.0201	-0.0079	0.0086	-0.0476	-0.0441	-0.9849

Table 4 Unanticipated fault diagnosis—UIL

Unanticipated Isolation Layer (UIL)									
Batch number	Anticipated or Unanticipated	Fault Pattern code	Updated true fault direction						
1	Null	0	0	0	0	0	0	0	0
2	U	1	1	-0.0043	0.0415	-0.0537	0.0459	-0.1359	0.0687
3	U	2	-1	0	-0.0340	0.049	0	0	0.0223
4	U	3	-0.1549	-1	-0.01	0.0108	0.0453	-0.1590	0.0354
5	U	4	-0.0018	1	0	0	0	-0.0022	-0.0077
6	U	5	0.0088	-0.0095	-1	0.0047	0	0	0
7	U	6	-0.0068	0.0052	0.0016	-1	-0.1565	0.0028	-0.0016
8	U	7	-0.0794	0.0053	0.0038	0.0019	1	-0.0144	-0.0567
9	U	8	-0.0744	0.0216	0.0029	0.0016	0.0455	-1	0.0447
10	U	9	0.0637	-0.0204	-0.0080	0.0087	-0.0483	-0.0448	-1

In the IDL, the fault detection can be realized, and the fault time and the current fault direction are also determined. In the UIL, **Criterion 2** is adopted to realize the isolation between the UF and the AF. In the initial stage, the AF pattern is assumed to be empty, therefore, when the second batch of data in Table 2 is filled into the UIL, the detected fault must be the UF, and then the isolation result is transferred into the URL. When the third batch of data in Table 2 is filled into the IDL, the fault time is that $t = 1001s$, the statistic of the directional similarity is $\|r\|(1 - |\cos(r, \xi_1)|) / \sqrt{\xi_1^T R \xi_1} = 7.3179$, and the isolation threshold of the UF is also $\Phi_{0.99} = 2.3263$. Obviously $\|r\|(1 - |\cos(r, \xi_1)|) / \sqrt{\xi_1^T R \xi_1} > \Phi_{0.99}$, the current fault pattern is different from the first fault pattern, and an UF occurs. Then the UF is transferred into the URL. The fault isolation result for all the tested data in Table 2 can be seen in Table 4. From Table 4, we know that the isolator with the fault feature direction and the direction similarity is valid, and the isolation between the UF and the AF can be truly realized.

After isolating the UF, the recognition of the UF should be carried out on the data layer. For the data in Table 2, the recognition result is that: the fault feature direction is $(0.9876, -0.0042, 0.041, -0.053, 0.0453, -0.1342, 0.0678)^T$. The variance with the largest contribution factor is the first dimension. According to **Explanation 6**, the contribution factor reaches 97 percent, and it indicates that the fault occurs for the earth sensor at the roll axis. Similarly, the result of the UF recognition in the URL for other batches of data is shown in Table 5. From Table 5, the recognition of the UF corresponding to the fault variance is correct, and the UF recognition of the data layer is reached.

Table 5 Unanticipated fault diagnosis—URL

Unanticipated Recognition Layer			
Batch number	Anticipated or Unanticipated	Fault pattern code	Variable subscript in Table 3
1	Null	0	0
2	U	1	1
3	U	2	1
4	U	3	2
5	U	4	2
6	U	5	3
7	U	6	4
8	U	7	5
9	U	8	6
10	U	9	7

6 Conclusion

The paper firstly takes the UF as a main diagnosis object. The detection and diagnosis method based on data driven for the UFs has been researched. The GPM for the UF diagnosis has been designed. The GPM is comprised of the IDL, the UIL and the URL. This GPM has provided a framework support for the UF diagnosis. According to the system both with the control input and the measure output, the system identification detection method corresponding to the IDL has been provided. The current fault feature direction and the feature direction of the AF pattern have been used to establish the statistic of directional similarity. The isolation between the AF and the UF has been realized in the UIL. According to the singular value decomposition, the fault contribution factor of each variance has been obtained, and the fault recognition in data layer has been completed. The application to fault diagnosis of the satellite's control system has demonstrated its validity.

Our research shall be furthered in two directions. Firstly, based on the framework of the GPM, the fault detection, isolation and recognition method on the foundation of model inference shall be researched. Secondly, the GPM and methods shall be applied to the diagnosis of other complex system for both military and civil use.

Acknowledgments

This work was supported in part by *National Natural Science Foundation of China* (NSFC) under Grant No. 61304119. Besides, we would like to especially thank *China Aerospace Science and Technology Corporation* (CASA) for providing the satellite control system data.

References

- [1] P. Nomikos, J. F. Macgregor. Monitoring of batch processes using multiday principal component analysis. *AIChE J*, 1994, 40(8): 1361-1375.
- [2] R. Isermann. Supervision, Fault detection and fault diagnosis methods-an introduction. *Journal of Control Engineering Practice*, 1997, 5(5): 639-652.
- [3] D. M. J. Tax. One-class classification. *Ph.D., Delft University of Technology*, Holland, 2001.
- [4] S. Gayaka1, B. Yao. Accommodation of unknown actuator faults using output feedback-based adaptive robust control. *International Journal of Adaptive Control and Signal Processing*, 2008, 25(11): 965-982.

- [5] P. Smyth. Markov monitoring with unknown States. *IEEE Journal on Selected Areas in Communication*, 1994, 12(9):1600-1610.
- [6] Hofbauer, B.C. Williams. Hybrid diagnosis with unknown behavioral modes. *Proceedings of the 13th International Workshop on Principles of Diagnosis (DX02)*, May, 2002.
- [7] V.J. Hodge, J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review. Kluwer Academic Publishers*, Vol. 22, 2004, 85-124.
- [8] Patcha, J. M Park. An overview of anomaly detection techniques: existing solutions and latest technology trends. *Computer Networks*, 2007, 51: 3448-3470.
- [9] K. Kojima, K. Ito. Autonomous learning of novel patterns by utilizing chaotic dynamics. *IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC '99*, 1999, 1:284-289.
- [10] Petra Perner. Concepts for Novelty Detection and Handling Based on a Case-Based Reasoning Process Scheme. *Springer-Verlag Berlin Heidelberg*, 2007.
- [11] Satnam Singh, Haiying Tu, William Donat. Anomaly detection via feature-aided tracking and Hidden Markov Models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 2009, 39(1): 144-159.
- [12] Ching-Fang Lin. Predictive fault diagnosis system for intelligent and robust health monitoring. *AIAA Infotech@Aerospace*, 20-22, April, 2010, Atlanta, Georgia.
- [13] E. Sobhani-Tehrani, H. A. Talebi, K. Khorasani. Neural parameter estimators for hybrid fault diagnosis and estimation in nonlinear systems. *IEEE International Conference on Systems, Man and Cybernetics, Montreal*, 7-10, Oct, 2007, 3171-3176.
- [14] Amitabh Barua. Hierarchical fault diagnosis and health monitoring in satellites formation flight. *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, 2011, 41(2): 223-239.
- [15] B. Tom and J. Tom: Anomaly detection for advanced military aircraft using neural networks. *Aerospace Conference, IEEE Proceedings*, 2001, 6: 3113-3134.
- [16] Z. H. Duan: Theoretic and methodological research on fault diagnosis of mobile robots based on adaptive particle filters. *Ph.D., Central South University*, 2007, 63-89. [in Chinese].
- [17] B. Zhang, Chris Sconyers, Carl Byington, Romano Patrick, Marcos Orchard. Anomaly detection: A robust approach to detection of unanticipated faults. *International Conference on Prognostics and Health Management*, 6-9, Oct, Denver, CO, 2008: 1-8.
- [18] Z. M He, H. Y. Zhou, J. Q. Wang. Model for Unanticipated Fault Detection by OCPA. *Advanced Materials Research*, Vols. 591-593, 2012: 2108-2113.
- [19] J. Chen, R.J. Patton. Robust model-based fault diagnosis for dynamic systems. Boston: *Kluwer Academic Publishers*, 1999.
- [20] B. Zhang, S. Chris, B. Carl. A probabilistic fault detection approach: application to bearing fault detection. *IEEE Transactions on Industrial Electronics*, 2010, 58(5): 2011-2018.
- [21] Pierre Sens. An unreliable failure detector for unknown and mobile networks. *OPODIS 2008*, LNCS 5401, 2008, 555-559.
- [22] Anna M. Bartkowiak. Anomaly, novelty, one-class classification: a short introduction. *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference*, Wrocław, Poland, 8-10, Oct, 2010, 1-6.
- [23] F. N. Zhou. Extended DCA method for unknown multiple faults diagnosis. *Huazhong Univ. of Sci. & Tech. (Natural Science Edition)*, 2009, 37(4): 84-94 [in Chinese].
- [24] N. Gebraeel, J. Pan. Prognostic degradation models for computing and updating residual life distributions in a time-varying environment. *IEEE Trans. Rel.*, 2008, 57(4): 539-550.
- [25] Wang Z M, Yi D Y, Duan X J. Measurement data modeling and parameter estimation. *CRC Press*, 2011.
- [26] Adrian Wills, Brett Ninness. On gradient-based search for multivariable system estimates. *IEEE Trans. Automat. Control*, 2008, 53(1): 298-306.
- [27] E. Wernholt, S. Moberg. Nonlinear gray-box identification using local models applied to industrial robots. *Automatica*, 2011, 4(47): 650-660.
- [28] Lennart Ljung. System identification: Theory for the User. *Linköping University, Sweden Published*, 1998.
- [29] Goethals, K. Pelckmans, J. A. K. Suykens, B. De Moor. Sup-space identification of Hammerstein systems using least squares support vector machines. *IEEE Transactions on Automatic Control*, 2005, 50(10): 1509-1519.
- [30] Tu S C. Satellite attitude dynamics and control. Beijing: *Chinese Astronautic Publishing House*, 2003; 125-168 [in Chinese].

A SCADA Expansion for Leak Detection in a Pipeline*

Rolando Carrera* Cristina Verde* and Raúl Cayetano**

*Universidad Nacional Autónoma de México, Instituto de Ingeniería
e-mail: rcarrera@unam.mx, verde@unam.mx

** Universidad Nacional Autónoma de México, Posgrado de Ingeniería
email: rcayetano@ii.unam.mx

Abstract

A solution for expanding an already existing pipeline SCADA for real time leak detection is presented. The work consisted in attaching a FDI scheme to an industrial SCADA that regulates liquid distribution from its source to end user. For isolation of the leak a lateral extraction is proposed instead of the traditional pressure profile of the pipeline. Friction value is a function of pipe physical parameters, but on line friction estimation achieved better results. Aspects that were important in the integration of the FDI scheme into the SCADA were the non synchrony of pipeline variables (flow, pressure) and their accessibility, that led to data extrapolation and the use of data base techniques. Vulnerability of the location algorithm due to sensors bandwidth and sensitivity is showed, so the importance of selecting them. The FDI scheme was programmed in LabVIEW and executed in a personal computer.

1 Introduction

Leak detection and isolation in pipelines is an old problem that has attracted the attention of the scientific community since decades. A paradigmatic example is the oil leakage in the Siberian region [1], where the effects on the surrounding nature have been disastrous. In Mexico, a semi desert country, there is the need to transport water to the population on long distances via aqueducts; this requires complex supervision systems that detect leakages in early ways. Also, there exist a complex net of pipelines that transport oil and its by-products; in this net, besides the leakage problem, there exist also the illegal extraction of the product transported in the pipeline; this forces that the distribution system should have a leak detection and location monitoring system.

Since the 1970's years have been issued several works that have been fundamental for the detection and location of leakages as the one of Siebert [2], where on the basis of the steady state pressure profile along the pipeline simple expressions are derived, based on correlations, that detect and locate a leakage. Later Isermann [3] published a survey showing the state of the art on fault detection by using the plant model and parameter identification. Recently, Verde published a book [4] making emphasis on signal processing,

pattern recognition and analytical models for failure diagnosis.

But all the later is pure academical, our aim here is to share some of our practical experiences acquired during a re-engineering project that consisted on adding a real time leak detection and location layer to an already existing SCADA. The original objectives of that SCADA were the administration and delivery of some products, through pipelines, from the source to the end user. As it was our first approach to integrating a FDI to an existing SCADA and that we didn't have experience on this subject, we proposed a solution that involves simple algorithms for detecting and locating a leak. In future work we'll use more elaborate algorithms as dedicated observers or detecting two simultaneous leaks.

In order to show how we solved the targets of the project we divided the solution in five major parts (each one included in sections 2 to 6 down here). Some of them are extracted from available theory, as the dynamical model for a flow in a pipe and the expression for leak location, and others are consequence of the experience achieved in our lab facilities, as the calculus of pipe friction and the choice of sensors, and finally the data acquisition imposed by the nature of the available SCADA.

Delivering a fluid to clients means steady operation, then our solution required a suitable model for that condition, section two describes how to achieve a simple steady state model for a pipeline. Once the model is at hand an appropriate expression for leak location is needed, for that purpose in section three a simple method for locating a leak is presented. From our experience, pipe friction plays a fundamental role in the exact location of the leak and that real time estimated friction is better than a beforehand constant one; an on-line expression for calculating the pipeline friction is showed in section four. In this project we didn't have the option to choose sensors, but we consider appropriate to share here our experience in this matter, a comparative study on how different type of sensors affect the leak location is presented in section five. The data acquisition system of the SCADA is based on a MODBUS system and a database with the information of the pipe variables, we didn't have the right to get into the MODBUS but in the database, section six shows how the indirect measurement of pipe variables issue was solved by using ethernet and data bases, also, the extrapolation of data of non existing data during sample times is presented. Finally, the concluding remarks of this work are presented in section seven.

*Supported by II-UNAM and IT100414-DGAPA-UNAM.

2 Pipeline steady state model

In most applications a dynamical model of the system is required but not here because of the steady operation of the pipeline, then a steady state model is more suitable. Besides, the pipeline lies buried in the field and has an irregular topography, but it is possible to derive a model that handles it like a horizontal one. This model is simpler as will be showed.

In the following we modify the model of a pipeline with topographical profile as showed in Figure 1 into one with a right profile piezometric head, where the pressure variable depends on a reference value h , as is the height over sea level along the pipeline. Consider the one dimension simplified flow model in a pipeline with n sections [5],

$$\frac{1}{A^i} \frac{\partial Q^i(z^i, t)}{\partial t} + g \frac{\partial H^i(z^i, t)}{\partial z^i} + \frac{f Q^i(z^i, t) |Q^i(z^i, t)|}{2D^i (A^i)^2} + g \sin \alpha^i = 0 \quad (1)$$

$$\frac{\partial H^i(z^i, t)}{\partial t} + \frac{b^2}{g A^i} \frac{\partial Q^i(z^i, t)}{\partial z^i} = 0 \quad (2)$$

which assumes that fluid is slightly compressible, pipe walls are slightly deformable and negligible convective changes in velocity. Q is volumetric flow, H is pressure head, A is pipe cross-sectional area, g is gravity, f^1 is the D'Arcy-Weissbach friction [6], b is the velocity of pressure wave, D is pipe diameter, z is distance variable and t the time. Super index $i = 1, 2, \dots, n$ indicates pipeline section characterized by its slope with angle α^i , n is the total number of sections.

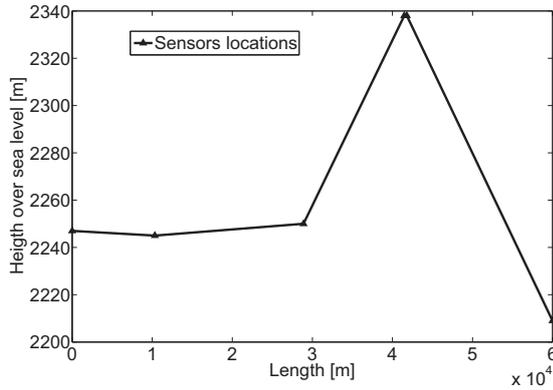


Figure 1: 60 km Pipeline topographical layout

We start with the following hypothesis: the system works in steady state and that the pipeline lay on an horizontal surface. Therefore we need a steady state model that takes into account these conditions.

In order to describe the behaviour of the pressure head $H^i(z^i, t)$ along a section without branches it is assumed steady state flow, so from (2) one gets

$$\frac{\partial Q^i(z^i, t)}{\partial z^i} = 0 \Rightarrow Q^i \text{ constant} \quad (3)$$

Combining (1) and (2)

$$\frac{dH^i(z^i)}{dz^i} + M^i(Q^i) = 0, \quad (4)$$

¹This friction characterizes the shear stress exerted by the conduit walls on the flowing fluid.

with

$$M^i(Q^i) = \mu^i Q^i |Q^i| + \sin(\alpha^i) = m^i(Q^i) + \sin(\alpha^i) \quad (5)$$

that is independent of the spacial coordinate z^i , and $\mu^i := f^i / 2D^i (A^i)^2 g$. Then the solution of (4) reduces to

$$H^i(z^i) = -M^i(Q^i) z^i + H^i(0) \quad \text{for } 0 \leq z^i \leq L^i \quad (6)$$

with $H^i(0)$ the pressure head at the beginning of section i . Defining boundary conditions for section i in terms of pressure at the ends:

$$H^i(z^i = 0) := H_{in}^i \quad H^i(z^i = L^i) := H_{out}^i. \quad (7)$$

with (7) in (6), we obtain

$$H_{in}^i - H_{out}^i = M^i(Q^i) L^i = m^i(Q^i) L^i + \Delta H_i, \quad (8)$$

where $\Delta H_i = L^i \sin(\alpha^i)$ is the height difference between section ends.

It is reported in [7] and [8] that the pressure head

$$H^i(z^i) = \frac{P^i(z^i)}{\rho g} \quad (9)$$

can be written in terms of the *piezometric head* $\tilde{H}^i(z^i)$, wich depends on a heigth h that can be related to sea level, i.e.

$$\tilde{H}^i(z^i) = H^i(z^i) + h(z^i), \quad (10)$$

$h(z^i)$ in m over reference datum or sea level, ρ is fluid density. Then the profile pressure (8) is equivalent to

$$\tilde{H}_{in}^i - \tilde{H}_{out}^i = m^i(Q^i) L^i \quad (11)$$

for section i and sea level $h(z^i)$ along the section. Finally, considering that boundary conditions are related by

$$\tilde{H}_{out}^i = \tilde{H}_{in}^{i+1}, \quad (12)$$

from this equation and (11) one gets

$$\tilde{H}_{in}^1 - \tilde{H}_{out}^n = \sum_{i=1}^n L^i m^i(Q^i) \quad (13)$$

which is function of the piezometric head for a pipeline with n sections without branches.

The profile of Figure 1 corresponds to the topography of the pipeline under study. The pressure head $H(z)$ and the resulting piezometric head $\tilde{H}(z)$ are shown in Figures 2 and 3, respectively. Take into account the uniformity of $\tilde{H}(z)$ similar to the one of a horizontal pipeline. The reference datum was the height of the first sensors location.

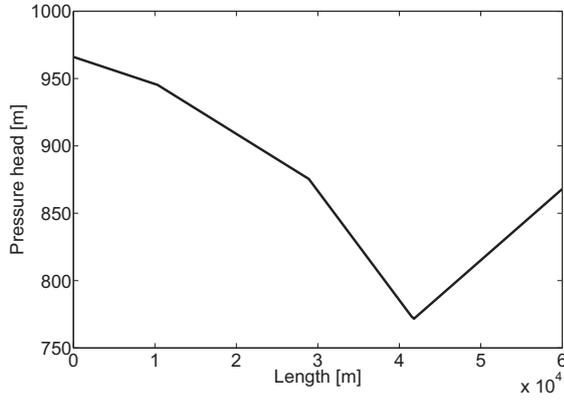
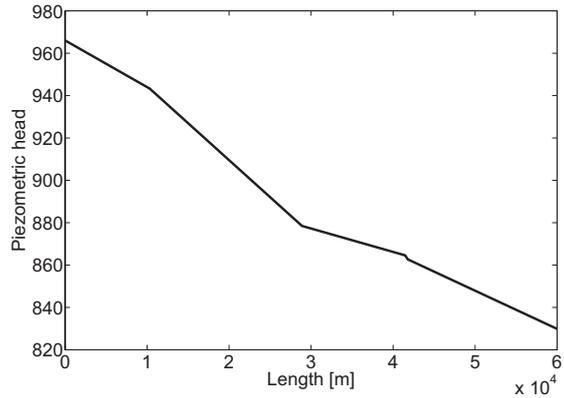
As a consequence, if $\tilde{H}_{in}^1 = \tilde{H}_{in}$ and $\tilde{H}_{out}^n = \tilde{H}_{out}$, besides if $m^i(Q^i) = m(Q) = M(Q)$ for all i , then Equation (13) becomes

$$\tilde{H}_{in} - \tilde{H}(z) = LM(Q) \quad (14)$$

where $L = \sum_{i=1}^n L^i$ the total length of the pipeline. Equation (14) is the steady state piezometric model for the pipeline viewed as a horizontal one.

3 Leak location

We consider a leakage as an outlet pipe at the leak location as is shown in Figure 4. A branch or lateral pipe in section i breaks the continuity of variables $Q(z, t)$ and $H(z, t)$, therefore new boundary conditions must be satisfied [9]. In

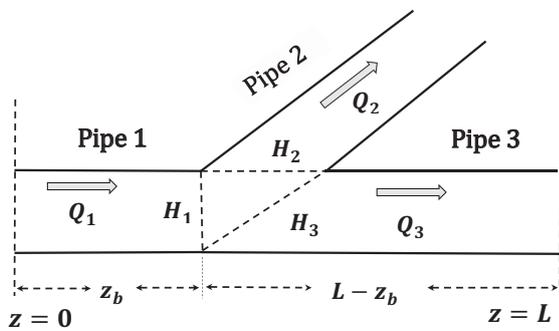

 Figure 2: Pipeline pressure head profile $H(z)$

 Figure 3: Profile of the piezometric head $\tilde{H}(z)$

particular, the union of three pipes is associated to a geometry shown in Figure 4 and the corresponding conditions that describe the action of separating flow are reduced to

$$H_2 = H_1 + \kappa_{12}(H_2, H_1) \quad (15)$$

$$H_3 = H_1 + \kappa_{13}(H_3, H_1) \quad (16)$$

where H_2 and H_3 are pressures at the beginning of pipes 2 and 3 and the functions $\kappa_{1\eta}(\cdot, \cdot)$ with $\eta = 2, 3$ represent losses caused by friction and change of flow direction. For adjusting the order of magnitude of these functions flow simulations were held with Pipelinestudio [10] with the topology of the study case shown in Figure 1. Simulation reported that terms κ_{12} and κ_{13} were negligible, then


 Figure 4: Union of three branches in point z_b of pipeline with transversal section areas A_1, A_2 and A_3

$H_1 = H_2 = H_3$. Thereafter in the study was included only the balance

$$Q_1 - Q_2 - Q_3 = 0, \quad (17)$$

as consequence,

$$Q_1 = Q_{in}, \quad Q_3 = Q_{out} \quad (18)$$

with Q_{in} y Q_{out} flows at the ends of the pipeline. So the differential equation (4) transforms in two equations

$$\frac{dH^1(z)}{dz} - M(Q_1) = 0; \quad \text{for } 0 \leq z \leq z_b \quad (19)$$

$$\frac{dH^3(z)}{dz} - M(Q_3) = 0; \quad \text{for } z_b < z \leq L,$$

describing the pressure head along the section with a branch in point z_b . As the equations (19) have the same form as (4), their solutions also have the same as (6). Therefore, with boundary conditions:

1. $H^1(z = 0) = H_{in}$,
2. $H^3(z = L) = H_{out}$,
3. $Q_{in} = Q_{out} + Q_{z_b}$ and
4. $H_{z_b} - \epsilon = H_{z_b} + \epsilon$ with $\epsilon \rightarrow 0$

Assuming that all pipes have same diameters, solutions of (19) evaluated at the ends are reduced to

$$\frac{H_{in} - H_{z_b}}{z_b} - M(Q_{in}) = 0 \quad (20)$$

$$\frac{H_{z_b} - H_{out}}{L - z_b} - M(Q_{out}) = 0.$$

Obtaining the variable z_b associated to the position of the branch

$$\begin{aligned} z_b &= \frac{M(Q_{out})L^i + H_{out} - H_{in}}{M(Q_{out}) - M(Q_{in})} \\ &= \frac{L \sin \alpha + m(Q_{out})L + H_{out} - H_{in}}{m(Q_{out}) - m(Q_{in})}, \end{aligned} \quad (21)$$

in terms of the piezometric head

$$z_b = \frac{m(Q_{out})L + \tilde{H}_{out} - \tilde{H}_{in}}{m(Q_{out}) - m(Q_{in})}. \quad (22)$$

Equation (22) is the key for leak isolation. In order to see the performance of this leak location method some experiments were held in our pipe prototype [11], which is an iron pipe of 200 m long, 4 inches diameter and six valves attached to it for leak simulations. Table 1 shows the percent deviations of locating the leak position. In each experiment a valve was fully open. Coriolis sensors were used.

4 Pipeline friction

The D'Arcy-Weissbach friction is a function of the pipe parameters, [6] and [12], and operation conditions, as the Reynolds number. For practical purposes the friction f is obtained from tables provided by the pipe manufacturers. But we observed that that value differs from the real one of a working pipeline where, no matter that is working in steady state, the value is influenced by noise -caused by pipe inner surface imperfections and attachments (nipples, elbows, etc.-), therefore using a previous fixed value of f is of no use in Equation (1).

Table 1: Location error in percentage of total pipe length

Experiment	Δz_b [%]	Valve position [m]
1	1.66	11.54
2	2.93	49.83
3	0.135	80.36
4	0.54	118.37
5	0.375	148.93
6	3.42	186.95
Mean	1.0	

To overcome the problem of not having the friction right value, we proposed a solution that was an on line friction estimation. In the following we show how to calculate this friction. For that, we part from the steady state momentum equation, Equation (4). Turning back the original parameters we get

$$g \frac{dH}{dz} + \frac{f}{2DA^2} Q |Q| + g \sin \alpha = 0 \quad (23)$$

solving the integral, considering that H_0 and H_L are pressures at the beginning and at the end of the pipeline and L the length, results

$$g(H_L - H_0) = -\left(\frac{f}{2DA^2} Q_\infty^2 + g \sin \alpha\right) L \quad (24)$$

where Q_∞ is volumetric flow in steady state, the absolute term disappears when flow goes in one direction only. Friction has the following expression

$$f = \frac{2DA^2 g (H_0 - H_L - L \sin \alpha)}{L Q_\infty^2} \quad (25)$$

Equation (25) is used to calculate on line the friction value, as is shown in Figure 5, experiment realized in our pipeline prototype. The calculated friction has a considerable amount of noise, but this noise can be attenuated via weighted mean value with forgetting factor (MVFF, continuous line in figure). Actually, we are working on the use of recursive identification procedures for a better friction estimated.

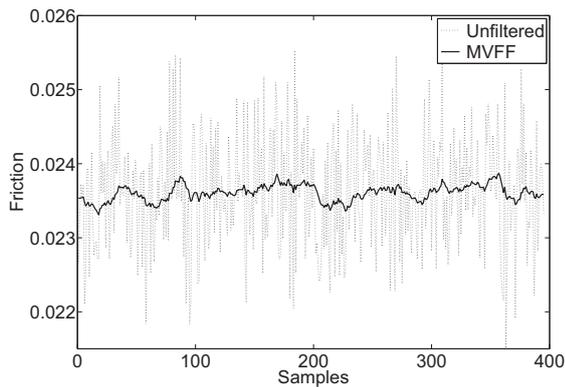


Figure 5: Friction estimated, raw and filtered

5 Influence of sensors on location

Flow measurement in a pipeline is fundamental for leak location, in view that most of the pipeline leak detection meth-

ods are based on processing a residual that is a flow difference. Due to our lack of experience, and by suggestion of a supplier, we start our flow measurements with a paddle wheel flow sensor [13]. Later on, as ultrasonic sensors are widely used in the field, we decide to change to them [14], thinking that our measurements would be better. Finally, we reached the conclusion that success on leak detection and location depends strongly on the sensors quality (make and sensing principle), so we acquired sensors based on the Coriolis effect [15].

An experiment that we made in our pipe prototype was to cause a leakage (outflow in a extraction point) and estimate the location with the measurements of the three sensors. Figure 6 shows the deviation of the calculated location depending on the type of sensor. Oscillations are observed around the operating point, which leads to the necessity of signal filtering in the diagnosis process. Table 2 shows the error leak location, Paddle Wheel and Coriolis sensors have similar error, but standard deviation is bigger with the Paddle Wheel. In order to compare performance in the fourth column the accuracy of the instruments are presented; remark that Coriolis error standard deviation is about seventy times bigger than sensor accuracy. The observation here is that the quality of the results depends more on the behavior of the flow than on the accuracy of the instrument used.

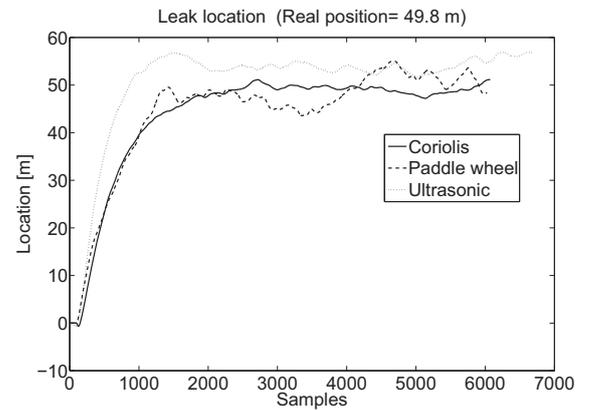


Figure 6: Leak location with the three sensors

Table 2: Leak location errors

Sensor	Error [%]	Error STD [%]	Accuracy [% FS]
Paddle wheel	-0.28	3.36	0.50
Ultrasonic	2.12	1.39	2.00
Coriolis	0.28	0.84	0.05

One of our goals in the SCADA expansion project was to deliver results in real time. For this, sensors experiments were performed to determine which one would have the faster response. An index to take into account is the time response, it can be appreciated in Figure 6 but is practically the same, therefore we measured the settling time from the moment when the leakage valve is opened. In Figures 7, 8 and 9 the flow development is observed, dotted line indicates the time when the leakage valve is opened to 100%. In Table 3 are the measured times, being the ultrasonic sensor which requires more time (this by the number of points used to calculate a mean value).

Table 3: Sensors settling time

Sensor	t_s [s]
Paddle wheel	3
Ultrasonic	35
Coriolis	4

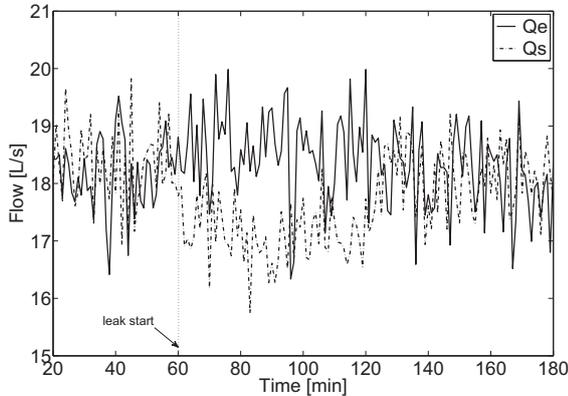


Figure 7: Flow measurement at the pipe ends, paddle wheel sensors

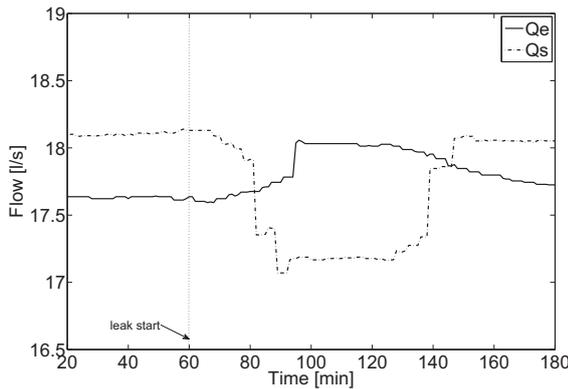


Figure 8: Flow measurement at the pipe ends, ultrasonic sensors

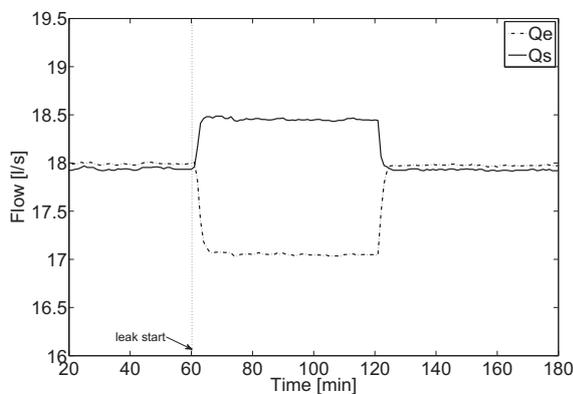


Figure 9: Flow measurement at the pipe ends, Coriolis sensors

Considering the settling time and noise in measurements (taking the STD as the measure for that), Coriolis sensor has the best performance. Experiments showed in this section were made with 1 s sampling period.

6 Asynchronous data and data bases

In the academy, we are used to work with benchmark systems or laboratory facilities with *ad hoc* data acquisition systems, sufficient sensors, controlled environments, etc. But these conditions are not necessarily in the practice, as was the case of the SCADA expansion, where the access to flow and pressure sensors of the pipeline were not available, but through a database. So the solution adopted was as follows:

1. The leak locator is on a dedicated computer, independent of the system that regulates the distribution of the fluid, it connects to the database server, see Figure 10, via intranet or VPN (Virtual Private Network) connection in a LAN (Local Area Network) system.
2. With proper permission a program, task performed with Visual Studio 2010 tool that runs every minute (it is a program without GUI -Graphic User Interfacer-that runs silently), brings system data and creates a database with pipeline flow and pressure information, data required by the locator for proper operation.
3. The locator program (made in the LabVIEW platform, [16]) periodically takes data (through SQL data server of Microsoft), applies the detection algorithm and when detects a leak proceeds to locate it, displays on the screen the location of the leak (Figures 12 and 13), generates a visual warning and creates a file with data leakage.

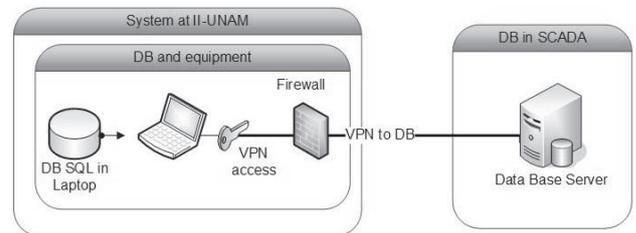


Figure 10: Communication scheme between leak locator and database

But the data acquisition system of SCADA do not meet the condition of sampling the system variables with constant sampling period. The nominal sampling period was 3 min, but in reality this varies from one to several tens of minutes. On the other hand, the locator was assigned a sampling period of 3 min, determined by the condition that nominally SCADA performs a polling of all measuring stations in that time span. To solve the problem of having a value of flow and pressure of each station at all sampling time, it was added to the localizer an algorithm that extrapolates the missing data when it is not available. Two algorithms were tested, one that retains the last data in the following sampling periods and one that generates straight line with the last two values available, that when the value of the variable that is brought from the database is not a new one, then the one determined by straight line is used. In order to compare results with both proposals a simulation with real data with

three leaks was carried on, in Figure 11 the real and extrapolated input flow data are shown. It can be seen that at certain intervals the extrapolation by a straight line delivers values that may be beyond the normal range of measurements, this situation is exacerbated in large intervals with empty data as the line grows monotonically delivering data outside the region of validity. In Figure 12 the location of a leak is shown when extrapolated data are used and in Figure 13 when retained data are used. The pipe length is about 20 km, so that retention has outperformed extrapolation, since the latter yields higher values than the length of the pipe. Original leak location was about 10 km.

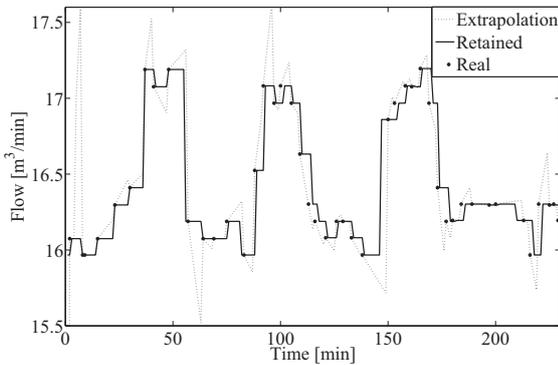


Figure 11: Graphics with original, extrapolated and retained data of input flow with three leaks

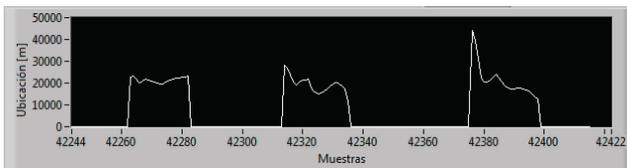


Figure 12: Leaks location with extrapolated data

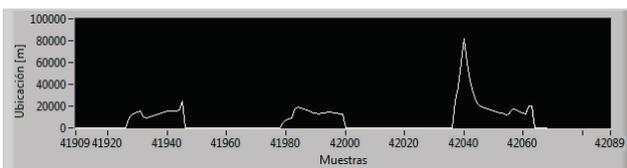


Figure 13: Leaks location with retained data

6.1 Alternate database communication

As part of the project requirements, an alternate way of communication with the SCADA database was experimented. In previous section the communication between leak locator and database was direct through a LAN system, the alternate way was through a third party via internet and VPN connection. Figure 14 shows the principal elements of this scheme.

The client is the computer with the locator program build in LabVIEW platform that performs basically two activities: leak detection and location, and request and sending data to communications broker using JSON strings. The remote client interface is a Java process that runs locally and handles communication, authentication, data formatting, encryption and security of the communication with data server.

It connects to the database in the SCADA through TCP sockets and VPN.

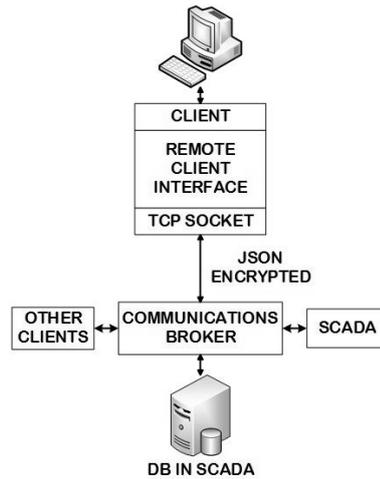


Figure 14: Communications between client and database

For data handling JSON format is used, which is broadly used for information interchange through internet. JSON (Java Script Object Notation) is a data interchange text format, easy for humans to read and write [17]. JSON is a collection of pairs {variable name : value}, realized as an object, record, structure, dictionary, hash table, keyed list, or associated array, see in Figure 15 an object example.

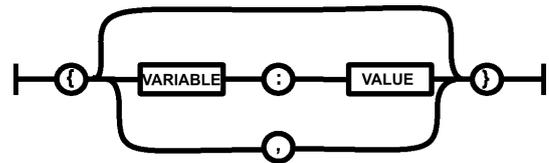


Figure 15: JSON data format for an object

An example of a JSON string for reporting a leak is the following:

```
{ "service": "event",
  "options": {
    "action": "new",
    "vector": {
      "Module": "XXX",
      "EventID": "XXX",
      "Quantity": "XXX",
      "PipeID": "XXX",
      "Location": "XXX",
      "TimeEvent": "yyyymmddhhmmss"
    }
  }
}
```

Communications broker attends clients requests (leaks locator is not the only one) and also SCADA requests. The database attached to the broker contains not only pipeline data but also data generated by the other clients. At the end, the SCADA has an interface in which information of leakage events is displayed.

Figure 16 shows a test ran with real data but off line. That experience showed that locator not always received answers from the broker. But this communications scheme is still in development.

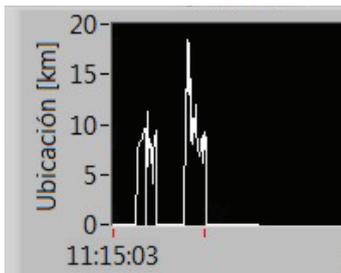


Figure 16: Off line experiment with real data. Detail of the graph, y axis is leak location in km

7 Conclusions

An interesting result is that a pipeline with certain topography may be analyzed as an horizontal pipe in which the piezometric head is a sum of measurements and terrain heights, Equation (10), as seen in section 2.

Compared with traditional methods for locating a leak in a pipe, the method shown here, Equation (22), requires less computational effort and has a simple expression for calculating it.

Another relevant result is the expression for on line calculation of the pipeline friction, Equation (25), as it is enough to measure pressure at the ends and steady state flow. The value of friction was found to be a key parameter for the exact location of the leak. It is to remark that when a leak occurs the pressures change modifying the friction value; in order to avoid wrong location of the leak we keep a delayed value of friction that is frozen when leak alarm occurs.

On the other hand, is to highlight the importance of choosing the appropriate sensor. It is not enough to choose a sensor capable of measuring a certain physical variable, also must be included in the selection process the purpose for which the measurements are needed.

The world of measurements for control targets is not limited to direct measurement of the physical variable, it is possible to achieve the control objectives with indirect measurements, as was the case of reading the variables from the plant via the network to a database. Also, with the partial absence of data we cannot use the plant model to predict data, then the use of extrapolation methods proves to be a powerful tool that helped to achieve the goal of this project; in this paper we use two simple methods, but this is an area that we continue to explore.

The experience with JSON format strings showed that it is easier to work with text characters than with specialized database commands and, no matter the VPN connection and data encryption, the scheme depends strongly on internet conditions. If internet fails leak detection scheme fails, situation that scarcely appears when the locator connects with database through a LAN system.

To the moment this paper was written our FDI system is in the proof stage at the SCADA facilities and we are waiting for in the field results.

8 Acknowledgments

Authors are very thankful to Jonathán Velázquez who helped us by solving the database issues emerged in this project.

References

- [1] I. Bazilescu and B. Lyhus. Russia oil spill. <http://www1.american.edu/ted/KOMI.HTM>, 1994.
- [2] H. Siebert and R. Isermann. Leckerkennung und -lokalisierung bei pipelines durch on-line-korrelation mit einem prozessrechner. *Regelungstechnik*, 25:69–74, 1977.
- [3] R. Isermann. Process fault detection based on modeling and estimation methods - a survey. *Automatica*, 20(4):387–404, July 1984.
- [4] C. Verde, S. Gentil, and R. Morales. *Monitoreo y diagnóstico automático de fallas en sistemas diámicos*. Trillas, 2013.
- [5] M. Hanif Chaudhry. *Applied Hydraulic Transients*. Springer, third edition, 2014.
- [6] C. F. Colebrook and C. M. White. Experiments with fluid friction in roughened pipes. *Proceedings of the Royal Society of London*, 1937.
- [7] J. Saldarriaga. *Hidráulica de acueductos*. Mc Graw-Hill, 2003.
- [8] R. Bansal. *Fluid mechanics and hydraulic machines*. Laxmi Publications (P) LTD, 2005.
- [9] Oke A. Mahgerefteh, H. and O. Atti. Modelling outflow following rupture in pipeline networks. *Chemical Engineering Science*, (61):1811–1818, 2006.
- [10] PipelineStudio. Software in energy solutions international. <http://www.energy-solutions.com/>, 2010.
- [11] R. Carrera and C. Verde. *Prototype for leak detection in pipelines: Users Manual*. Instituto de Ingeniería, UNAM, Ciudad Universitaria, D.F., Noviembre 2010. In Spanish.
- [12] C. Tzimopoulos G. Papaevangelou, C. Evangelides. A new explicit relation for the friction coefficient in the darcy-weisbach equation. In *PRE10: Protection and Restoration of the Environment, Corfu, 05-09 July, 2010*, 2010.
- [13] G. Fisher. *Signet 2540 high performance flow sensor*. Georg Fisher Signet LLC, El Monte, CA, 2004.
- [14] Panametrics. *Two-Channel TransPort Mod. 2PT868 Portable Flowmeter. User's Manual*. PANAMETRICS, Inc., Waltham, MA, USA, 1997.
- [15] E+H. *Proline Promass 83 Operating Instructions*. Endress + Hauser Flowtec, Greenwood, IN, USA, 2010.
- [16] National Instruments. *LabVIEW user manual*. National Instruments Corporation, Austin, 2013.
- [17] JSON Organization. Ecma-404 the json data interchange standard. <http://www.json.org/>.

Automatic Model Generation to Diagnose Autonomous Systems

Jorge Santos Simón¹ and Clemens Mühlbacher¹ and Gerald Steinbauer¹

¹Institute for Software Technology

e-mail: {jsantos, cmuehlba, gstein}@ist.tugraz.at

Abstract

Autonomous systems' dependability can be improved by performing diagnosis during run-time. This can be achieved through model-based diagnosis (MBD) techniques. The required models of the system are for the most part handcrafted. This task is time consuming and error prone. To overcome this issue, we propose a framework to generate formal models out of natural language documents, such as technical requirements or FMEA, using natural language processing (NLP) tools and techniques from the knowledge representation and reasoning (KRR) domain. Therefore, we aim to enable the usage of MBD in autonomous systems with few extra burden. So doing, we expect a significant increase in the usage of MBD techniques on real-world systems.

1 Introduction

Dependability is a key feature of modern autonomous systems. It can be achieved by sound design and implementation, thorough testing and runtime diagnosing. To date, all these processes are still not completely automated and need substantial manual work. However, all these fields can greatly benefit from the use of model-based techniques. Design and implementation can be greatly improved through model-driven engineering, as stated in [1]. Model-based testing (MBT) has been demonstrated [2] to outperform traditional testing techniques in both invested time and number of errors found. Model-based diagnosis (MBD) is the main target of this work. It has been successfully used in industrial settings [3], reducing the need for human intervention. Although it has been increasingly adopted in recent years, we believe that its full potential is still to be developed.

All model-based techniques require appropriate models of the system. As stated in [4; 5], creating these models is the most prevalent limiting factor for their adoption. To overcome this barrier, we propose a method that automates models creation from the documents used during the system design. These comprise requirements documents, architectural designs, FMEA and FTA, among others. The content of these documents is often given in natural language and in semi-structured form and lacks a common semantics. Thus, the contained information is not accessible for a computer. However, advances in natural language processing (NLP) and the availability of common sense and domain-specific knowledge bases (e.g. Cyc [6],

RoboEarth [7]) make semi-automated derivation of models possible. Despite recent advances on this area [8; 9; 10], most techniques focus on very specific applications of the generated formal models. Thus, we pose the problem of generating a common knowledge base as an intermediate representation with a well defined semantics out of documents used during the system design process. From this central repository, different algorithms can extract different formal models for particular needs. We believe that this work can increase the acceptance of model-based techniques and broaden their use.

The motivation for this work came during the development of a model-based diagnosis and repair (MBDR) system for an industrial application. The aim is to improve the dependability of a fleet of robots that automatically deliver goods in a warehouse. As stated in [11], even minor failures often prevent a robot from accomplishing its task, decreasing the overall performance of the system. Moreover, the frequent need of human intervention increases costs and customer dissatisfaction. Using MBDR techniques, many of these failures can be automatically handled, allowing the robot to remain on service, perhaps with its capabilities gracefully degraded [12; 13]. In extreme cases, diagnosing a failure on time can prevent robot behaviors harmful for humans, itself or other elements in the environment.

Confronted with the lack of any formal model of the system, we were forced to manually code the models we need. However, this is both a time-consuming and error prone task, and also impose a maintaining burden as the system evolves. Accordingly, we believe that a mostly automated approach is not only convenient for the intended project but can also help extending the use of MBDR techniques to other projects and domains. Following this idea, we propose a framework that, in a first step, gathers the information from the project together with domain and common-sense knowledge in a machine-understandable knowledge base. Then, a suit of algorithms can extract formal models from this knowledge base for particular purposes. Though our aim is to automate the process as much as possible, human assistance will be requested whenever some pieces of information are missing or contradictory [14; 15].

The novelty of our proposal is two-fold: first, we emphasize the usability of the resulting models for MBD. Second, we aim to integrate all the sources of information typically available in an industrial development process, such as requirements, architecture, and failure modes. As a result, we expect to boost the range and applicability of the automat-

ically generated models. To better illustrate the proposed framework, we will use a small running example extracted from a real-world application. It comes to the robot's box loading operation, performed by the robot's load handling device (LHD).

The remainder of the paper is organized as follows: Related research on model generation is discussed in Section 2. Section 3 provides an overview of the proposed process. Section 4 describes the inputs used, while Section 5 describes the proposed NLP and KRR tool-chain to interpret them. Section 6 provides an example of an output model and its use for MBD. Finally, Section 7 summarizes the presented framework and discusses future work.

2 Related research

We start the brief discussion of related research with the work using NLP methods to derive models. The work of [9] uses NLP methods to derive a formal model out of requirements. This formal model can afterwards be transformed into different representations to test or synthesize the system. The method proposed in [10] uses NLP methods to derive design documents (class diagrams, etc.) out of requirements. These design documents can afterwards be used to implement the system. The authors of [8] proposes a method to extract action receipts from websites. These action receipts comprises the desired behavior in order to achieve a given goal. The method use how-to instructions and NLP tools to derive an action receipt which can be executed by a robot. Missing parts are inferred with the help of common sense knowledge about actions. In contrast to all these approaches, we propose a framework which incorporates different information sources to get a better understanding of the system. Furthermore, our framework generates different models out of an internal formal description depending on the needs of the intended diagnosis and testing tasks.

Beside NLP methods, machine learning can also be used to generate a model of the system. The work in [4] presented a method to statistically learn the model of the system under nominal conditions. The model describes the static interaction of the system components. In contrast, the method proposed in [5] learns the behavior of a system. The method infers from observed events similar/different states and merges similar ones. Furthermore, the variables in the system for each state are estimated. Both methods are only applicable if the system is already built. Instead, we create a model during the design phase, and so the model can be used right at the first stages of the life-cycle.

Missing or contradicting information must be detected and handled when generating models. The method in [15] tries to avoid faults in the requirements document. This is done through the transformation of the requirements into so called boilerplates. Through this semi-structured text, ambiguities are removed and a consistent naming is enforced. A different approach was proposed in [14] to diagnose a knowledge base for consistency. If the knowledge base is inconsistent, the user is asked as an oracle to pinpoint the problem. Afterwards, the user needs to fix this issue. In our framework, we will use ideas from both methods to derive a consistent knowledge base of the system.

3 Framework overview

We propose the framework depicted in Figure 1 to transform informal documents and knowledge into models suitable for MBD. The informal inputs (white squares with solid lines) are processed into intermediate representations (light gray squares with dashed lines) using techniques from NLP and KRR, as well as ontologies (e.g. Cyc). We condense them into a knowledge base together with all our knowledge about the system and its domain. Finally, a variety of algorithms can produce formal models suitable for MBD (gray squares with dot-dash lines).

4 Sources of information

The proposed framework takes artifacts from the design phase as inputs. We propose the use of the following four inputs, though additional sources can be incorporated if available:

1. Requirements document: The technical requirements document describes the expected system behavior. Therefore, it is a mandatory input. The models' quality and so the resulting MBD will heavily depend on the quality of the requirements. Thus, iterative improvement of the requirements and models is used, as proposed in [15]. For our running example, we have taken four requirements that describe the box loading process of a robot:
 - (a) When the robot is docked, it lowers the barrier.
 - (b) When the robot is ready to load, the load handling device starts rotating backward.
 - (c) The load handling device stops rotating backwards when the laser beam is triggered.
 - (d) After stopping the load handling device the barrier is raised.
2. Domain knowledge: This is the most fuzzy input, as it is available not as an artifact but as the knowledge and experience of the engineers involved. We distinguish three kinds of knowledge. Common sense knowledge can be provided by existing ontologies as Cyc [16]. Generic knowledge about the autonomous systems domain can be provided by dedicated ontologies as KnowRob [17]. Particular knowledge about the targeted system itself can be partially inferred from the system architecture, though other parts must be provided by the project engineers. The use of ontologies range from providing meaning to natural language concepts to inferring missing pieces of information.
3. Architecture: The architecture of the system defines its composing elements plus the relations between them. It is typically described as a set of diagrams generated during the design phase of the system. For our running example, we use the architecture excerpt depicted in Figure 2. It states that a robot consists of a LHD and other unspecified elements. Furthermore, the LHD consists of a laser beam, rollers and a barrier.
4. Failure Modes and Effects Analysis: FMEA looks at all potential failure modes, their effects and causes and determines a risk priority factor. FMEA can be used to determine which potential errors are critical, how they can be pinpointed, and how the effects thereof can be avoided [18]. We incorporate the failure modes into the resulting behavior models to diagnose these known

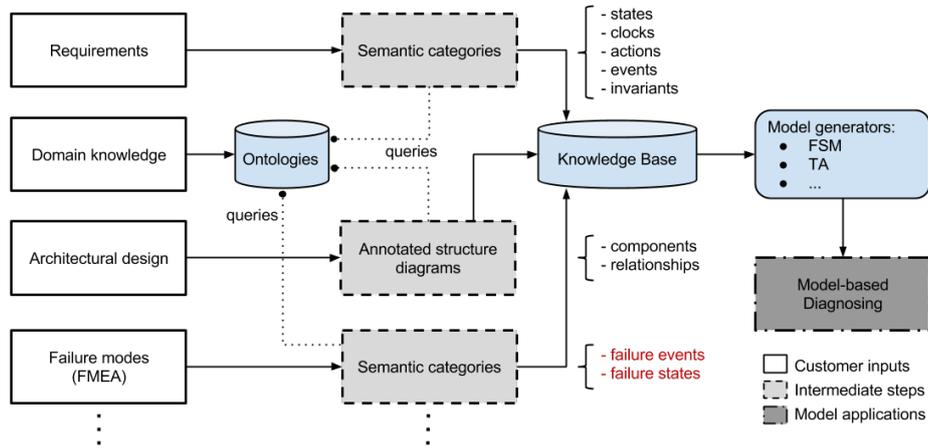


Figure 1: Abstract work-flow for the proposed framework. Starting from left with inputs in natural language, we generate models that can be applied for diagnosis (right).

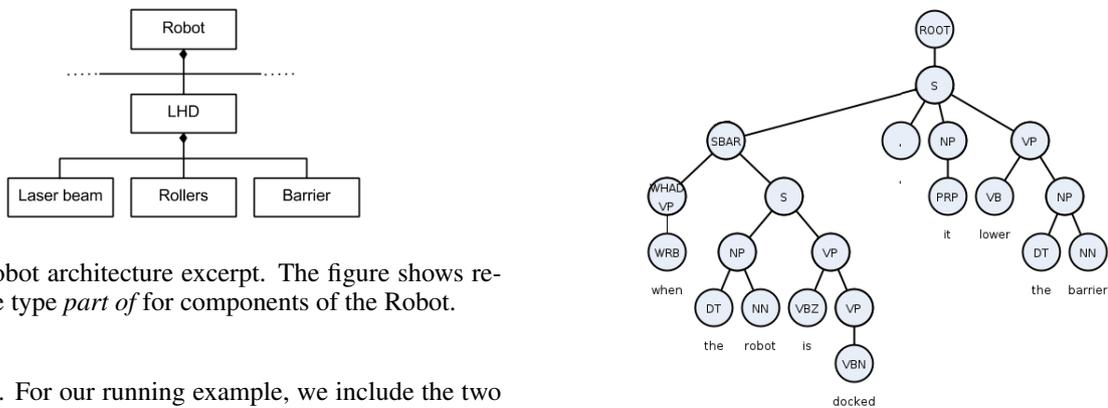


Figure 2: Robot architecture excerpt. The figure shows relations of the type *part of* for components of the Robot.

failures. For our running example, we include the two failure modes that can occur during the load operation, depicted in Table 1.

The biggest challenge for handling all these inputs is to understand semi-structured information. So, we will depict a NLP/KRR tool-chain using state-of-the-art techniques in the following section.

5 NLP/KRR tool chain

The process generates three intermediate artifacts: semi-formal text (boilerplates), syntax trees and semantic categories. As a showcase, we will concentrate on the requirements of our running example, though these techniques can be extended to other textual inputs, as we will see at the end of this section.

5.1 Boilerplates

This is a semi-formal representation where most of the spelling errors, poor grammar and ambiguities have been removed. Boilerplates also enforce the use of a consistent naming scheme. There exist tools such as [19] to perform this task semi-automatically. In our example, the four requirements become the four equivalent boilerplates:

- when the robot is docked, it lower the barrier.
- when the robot is ready to load, the lhd start rotating backward.
- when the lb is triggered, the lhd stop backward rotation.
- after stopping the lhd, the barrier is raised.

Figure 3: Sample syntax tree of the first sentence (a) of the running example.

Note for example that the 3rd person “s” has been removed from the verbs. Furthermore complex terms such as “load handling device” have been replaced by lhd. Finally, the propositions order is rearranged in a consistent structure.

5.2 Syntax trees

A syntax tree comprises the information of the type of each word in the sentence, e.g. “lower” is a verb. Furthermore, the tree specifies how the sentence is constructed with these words. For example, the syntax tree of the first requirement in our running example is depicted in Figure 3. In this syntax tree we can identify that “robot” is a noun and “the robot” is a so called noun phrase. An example of a tool to extract syntax trees is the probabilistic context free grammar parser, described in [20].

5.3 Semantic categories

The semantic categories conceptually describe our system, e.g. a transition describing the motion of an actuator. These semantic categories are hierarchical in nature, as more complex and abstract concepts are composed of simpler ones, e.g. a transition is composed by an action, pre and post conditions, etc. We obtain the semantic categories by parsing the syntax trees and applying transformation rules in a

	Component	Failure	Observations
Failure 1	Barrier	Barrier stuck up	Barrier stuck up regardless commands
Failure 2	Load Handling Device (LHD)	Rotation fail	Laser beam not triggered

Table 1: FMEA from the running example.

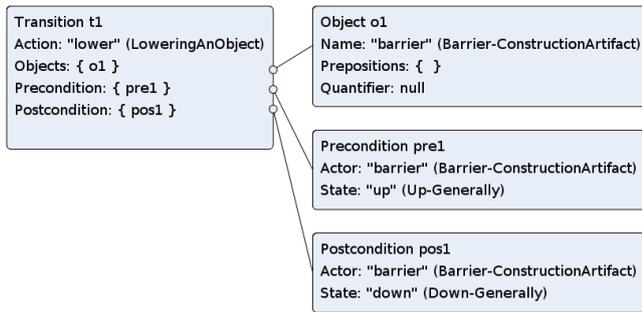


Figure 4: Concepts created from the syntax tree in Figure 3. The word in quotes is the word as it appears in the sentence. The word in parenthesis is the Cyc concept it belongs to.

bottom up fashion, following [8]. We start at the leaves of the syntax tree, containing single words. Each word has assigned a part-of-speech (POS) label describing its grammatical role in the sentence. Furthermore, each word has an additional label with its WordNet [21] synset, used to derive its semantics from the common sense knowledge base. From the leaves, higher level transformations can be applied to create more complex semantic categories. For example, on our running example we create a semantic category for each word in the sentence “lower the barrier”. Then, we can derive that “lower” is an action acting on something. We can after that use the semantic category of the word together with its position in the syntax tree to apply further transformation rules. This process is repeated till the root node is reached. Then, a new semantic category is assigned to the sentence capturing its semantics. For the running example, the semantic category for “lower the barrier” is a transition. A transition must contain a precondition, a post condition, an action and optionally an object of the action. The semantic category specifies that the action “lower” is performed on the object “barrier”. With the help of common sense (Cyc ontology [16]) we can reason that this action causes the “barrier” from state “up” to state “down”. Thus, we can infer the pre and post conditions of “lower”. Finally, the semantic category together with the reasoning results are packed into statements on our knowledge base, as it is depicted in Figure 4.

We can incorporate other documents into the knowledge base by using a similar NLP tool chain. However, how the information is treated depends heavily on the context inherent to each document type.

6 Model generation for behavior diagnosis

To illustrate how the framework can be used to diagnose the behavior of the robot, we create an automaton as output model. To use techniques such as [22], the automaton must describe both nominal and faulty behaviors of the system. To generate this automaton from the knowledge base, we use four different relations stated on it as transitions:

1. Relations representing a direct transition, as depicted in Figure 4. Such a transition can be directly mapped into a transition on the automaton, as can be seen in Figure 5 through the transitions from state 1 to 2.
2. Relations representing an action with a duration. Such a relation must be translated into several transitions: the start of the action, the termination event and a transition to a final state. Such transformed relation is depicted in Figure 5 through the transition from state 2 to 5.
3. Relations representing a failure of the system. The failure event is represented as a divergent path from a normal transition. Thus, the start state is the same as the one of the normal transition. Afterwards, we need a state representing the failure. Finally, we need an observation transition that leads to a final state representing a general failure of the system. The observable transition is cased due to the fact that use a fault model which is derived from the FMEA. Thus every fault has an observable discrepancy to the real system. Additionally it is important to notice that the state representing the general failure is state where the system can exhibit arbitrary behavior. Thus we can model the lack of knowledge which impact the fault has on the system. The transformed failure is depicted in Figure 5 through the transitions from state 2 to 9.
4. Relations representing a failure of a system component. The failure event is represented as a divergent path from a normal transition. To determine all the possible affected transitions, we must perform an inference of the effects each transition has. This inference is based on common sense and domain knowledge. In our running example, we can infer that lowering the barrier causes the barrier to be finally down. A failure such as *barrier_stuck_up* can prevent this transition, and so they can share a common source state. Then, as before we need an observation transition that leads to a final state representing a general failure of the system. Such a sequence is depicted in Figure 5 though the transitions from state 1 to 9 through the states 7 and 8.

7 Conclusion and future work

In this paper we propose a framework to automatically generate formal models out of documents represented in semi-structured form and natural language (requirements, domain knowledge, architecture, failure modes, etc.). The parsed information is gathered together with domain knowledge in a knowledge base. Accessing this common repository, a variety of algorithms can generate different kinds of models for different purposes. Our main target is to derive models suitable for state-of-the-art MBD techniques applied to autonomous systems. We plan to implement this framework to assist us on creating the models required for MBD. Doing so, we expect to improve the dependability in the industrial application of a fleet of transport robots in a warehouse.

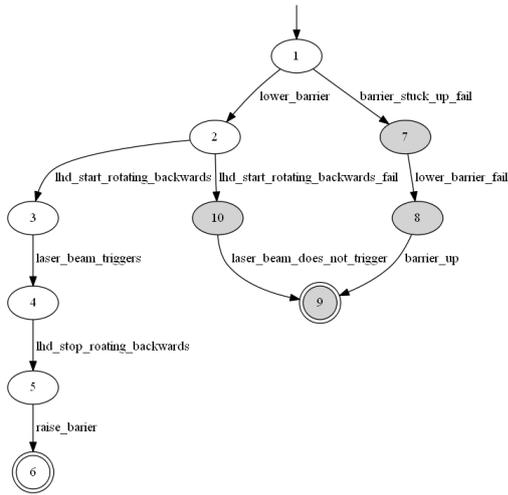


Figure 5: Automaton generated from the running example. Shaded states are reached through some fault. Double circled states represent final states. State number 9 is the general failure state for readability the self loops with all possible labels are omitted.

Besides this immediate result, we expect that the proposed framework will ease the creation of formal models for other applications. Thus, we hope to contribute to the widespread use of MBD techniques, with the consequent improve of autonomous systems dependability.

Acknowledgments

The research presented in this paper has received funding from the Austrian Research Promotion Agency (FFG) under grant 843468 (Guaranteeing Service Robot Dependability During the Entire Life Cycle (GUARD)).

References

- [1] Stuart Kent. Model driven engineering. In Michael Butler, Luigia Petre, and Kaisa Sere, editors, *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer Berlin Heidelberg, 2002.
- [2] Mark Utting and Bruno Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.
- [3] Peter Struss, Raymond Sterling, Jesús Febres, Umbreen Sabir, and Marcus M. Keane. Combining engineering and qualitative models to fault diagnosis in air handling units. In *European Conference on Artificial Intelligence (ECAI) - Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 1185–1190, 2014.
- [4] Safdar Zaman and Gerald Steinbauer. Automated Generation of Diagnosis Models for ROS-based Robot Systems. In *International Workshop on Principles of Diagnosis (DX)*, Jerusalem, Israel, 2013.
- [5] Dennis Klar, Michaela Huhn, and J Gruhser. Symptom propagation and transformation analysis: A pragmatic model for system-level diagnosis of large automation systems. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–9. IEEE, 2011.
- [6] Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.
- [7] Markus Waibel, Michael Beetz, Raffaello D’Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Oliver Zweigle, and René van de Molengraft. RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine*, 18(2):69–82, 2011.
- [8] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1486–1491. IEEE, 2010.
- [9] Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. Automatically extracting requirements specifications from natural language. *arXiv preprint arXiv:1403.3142*, 2014.
- [10] Sven J Körner and Mathias Landhäußer. Semantic enriching of natural language texts with automatic thematic role annotation. In *Natural Language Processing and Information Systems*, pages 92–99. Springer, 2010.
- [11] Gerald Steinbauer. A survey about faults of robots used in robocup. In Xiaoping Chen, Peter Stone, Luis Enrique Sucar, and Tijn van der Zant, editors, *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 344–355. Springer Berlin Heidelberg, 2013.
- [12] Gerald Steinbauer, Franz Wotawa, et al. Detecting and locating faults in the control software of autonomous mobile robots. In *IJCAI*, pages 1742–1743, 2005.
- [13] Mathias Brandstötter, Michael Hofbaur, Gerald Steinbauer, and Franz Wotawa. Model-based fault diagnosis and reconfiguration of robot drives. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.
- [14] Kostyantyn Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss. A direct approach to sequential diagnosis of high cardinality faults in knowledge-bases. In *International Workshop on Principles of Diagnosis (DX)*, Graz, Austria, 2014.
- [15] Bernhard K Aichernig, Klaus Hormaier, Florian Lorber, Dejan Nickovic, Rupert Schlick, Didier Simoneau, and Stefan Tiran. Integration of Requirements Engineering and Test-Case Generation via OSLC. In *Quality Software (QSIC), 2014 14th International Conference on*, pages 117–126. IEEE, 2014.
- [16] Stephen L Reed, Douglas B Lenat, et al. Mapping ontologies into Cyc. In *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, pages 1–6, 2002.

- [17] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1284–1289. IEEE, 2012.
- [18] Hongkun Zhang, Wenjun Li, and Jun Qin. Model-based functional safety analysis method for automotive embedded system application. In *International Conference on Intelligent Control and Information Processing*, 2010.
- [19] Stefan Farfeleder, Thomas Moser, Andreas Krall, Tor Stålhane, Herbert Zojer, and Christian Panis. Dodt: Increasing requirements formalism using domain ontologies for improved embedded systems development. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 271–274. IEEE, 2011.
- [20] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.
- [21] George Miller and Christiane Fellbaum. Wordnet: An electronic lexical database, 1998.
- [22] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *Automatic Control, IEEE Transactions on*, 40(9):1555–1575, 1995.

Methodology and Application of Meta-Diagnosis on Avionics Test Benches

R. Cossé^{1,2}, D. Berdjag², S. Piechowiak², D. Duvivier², C. Gaurel¹

¹AIRBUS HELICOPTERS, Marseille International Airport, 13725 Marignane France
{ronan.cosse, christian.gaurel}@airbus.com

²LAMIH UMR CNRS 8201, University of Valenciennes, 59313 Valenciennes France
{denis.berdjag, sylvain.piechowiak, david.duvivier}@univ-valenciennes.fr

Abstract

This paper addresses Model Based Diagnosis for the test of avionics systems that combines aeronautic computers with simulation software. Just like the aircraft, those systems are complex since additional tools, equipments and simulation software are needed to be consistent with the test requirements. We propose a structural diagnostic framework based on the lattice concept to reduce the time of unscheduled maintenance when the tests cannot be performed. Here, we also describe a diagnosis algorithm that is based on the formal lattice description and designed for test systems. The benefits is to capture the system structure and communication specificities to diagnose the configuration, the equipments, the connections, and the simulation software.

1 Introduction

Avionics systems are complex since tens of subsystems and components interact to achieve required functions. Existing devices for aircraft fault monitoring are based on dedicated avionics functions but the existing solutions are insufficiently flexible for test systems and can be improved. In [1], the framework of an health management algorithms for maintenance is described and implemented on an aircraft. In [2], the diagnostic of avionics equipments is performed through dynamic fault trees. To prevent important failures on the aircraft, avionics systems are checked on rigs called Avionics Test Bench (ATB) composed of the avionics equipments and flight simulation software.

The environment of the ATB needs to be compliant with the configuration of the avionics equipments. Faults of the ATB can concern the avionics equipments, their configurations, or the ATB itself i.e the movable connections and the simulation software. Since it does not exist monitoring functions of the ATB itself, a new method needs to be applied to prevent long periods of unavailability. In fact, during the development of embedded softwares, its architecture and the test environment surrounding the ATB are redesigned by adapting the test means to the specification's requirements. Since the ATB is a test system, and the main knowledge are based on its embedded systems, we need a new approach to deal with the ATB issues. As the embedded systems are already tested on the ATB, and the test results are used to focus on the ATB issues thanks to a new representation based on the model of the test system, the diagnosis of the ATB is what

we call a meta-diagnosis.

Many diagnosis approaches have been proposed to deal with specific avionics problems. Two different classes of representation are applied: data-based diagnosis or model-based diagnosis. The first one, as studied by Berdjag et al. [3] is used to recognize faulty behaviors of an Inertial Reference System (IRS) thanks to normal or faulty categories of input/output data. In this work, data fusion of outputs sensors is computed to eliminate faulty sources. In [2], the time dependency is introduced in data of failure messages to improve problems detection.

In Model Based Diagnosis (MBD), Kuntz et al. [4] have studied an avionics system using minimal cuts notions. Belard et al. have defined a new approach based on the MBD hypotheses called Meta-Diagnosis in [5] dealing with models issues. Berdjag et al. [6] present an algebraic decomposition of the model to reduce the complexity of the required model-based diagnosers. Giap [7] has proposed a formalism of an iterative process to give a solution when models are not complete but it lacks of applications on more complex industrial systems. Nevertheless, it gives clues for an iterative diagnosis. Another diagnostic software has been developed by Pulido et al. in [8] to perform consistency-based diagnosis of dynamic system simulating diagnosis scenarios. The architecture is quite novel and is applied to the three-tank system.

Structural approaches as graph theory are also popular for MBD to describe the structure of the system as with Bayesian Networks in [9]. They enable us to incorporate the system complexity as with the lattice concept to integrate the sub-models dependencies. For example, in [10], the lattice model represents fault modes to compute testable subsystems from redundancy equations. We want to get the main ideas that will serve our proposal. To our knowledge, there is no method for the diagnostic of test systems based on embedded softwares behaviour. Moreover, our proposition has been adapted from embedded systems to the ATB behaviour. Its complexity is relevant to the objectives of the avionics embedded systems certification, as for example high levels of safety requirements, or the simulation of specific test conditions. In our model, we must consider the fact that our representation must put forward the ATB behaviour in case of failures concerning embedded systems, connections, communications, simulation softwares and all settings to configure the test. Considering those features, the high number of needed ATB reconfigurations, it is proposed a structural representation associated with hierarchical verifications that reduce the faulty candidates. The motiva-

tion of the proposed meta-diagnosis approach was presented in [11]. Here, we propose an extended diagnosis methodology originally defined by De Kleer, Williams [12], [13] and Davis [14] and we present a software implementation running on a real ATB. It differs from the Belard et al.'s meta-diagnosis definition because the ATB is still defined as the main system under study. Here, we extend the diagnostic-world tools for a specific system and due to the lack of knowledge and data in case of issues, our proposal is based on a MBD representation with a structural and functional decomposition without fault models.

First, we describe the diagnostic framework, the lattice-based representation used to model the ATB system and the diagnostic algorithm. In the third section, we provide a description of the ATB and the application of the lattice concept. In the fourth section, we illustrate the approach with a case study of the ATB. In the final section, we describe the development of a software application to perform automatically the ATB diagnosis.

2 Diagnostic framework

2.1 System representation

The system is composed of several subsystems that interact together to achieve a global function. The decompositions into subsystems is guided by the communication between components to fulfill this goal. Partitions are used to decompose the system into functional and communications categories. So, there are two classes of partitions: the partitions that represent the structure and the connections of the system; and the partitions that represent the functions of the system. As an example, P_1 is associated with a functionality of the system $P_1 = \{\sigma_1; \sigma_2\}$, $\sigma_1 = \{C_1\}$ and $\sigma_2 = \{C_2, C_3\}$. If a problem appears, i.e the functionality is not performed, then a fault is detected for this partition P and symptoms are seen and linked to subsystems σ .

In the following paragraphs, we use the following notation: P for a partition, σ for a subsystem and c_i for a component. $S = \{c_i, i \in [1, n]\}$ is the set of all the n components of a system. We note Σ the set of all subsystems, i.e the power set of components. A partition P is a set of n_p subsystems $\sigma_i \in \Sigma$: $P = \{\sigma_i, i \in [1, n_p] | \forall i \neq j; \sigma_i \cap \sigma_j = \emptyset, \text{ and } \bigcup_{i=1}^{n_p} \sigma_i = S\}$. We note \mathcal{P} the set of all partitions.

We recall the definition 1 of inclusion relation between partitions and the definition 2 of multiplication.

Definition 1. Two partitions P_1 and P_2 are said to be in inclusion relation $P_1 \subseteq P_2$ if and only if every subsystems of P_1 is contained in a subsystem of P_2 . The relation \subseteq means that P_1 is a sub-partition of P_2 .

Definition 2. The subsystems σ_k of the multiplication of two partitions $P = \{\sigma_i, i \in [1, n_p]\}$ and $Q = \{\sigma_j, i \in [1, n_q]\}$ are defined by: $\forall \sigma_k \in P \times Q, \exists \sigma_i \in P, \exists \sigma_j \in Q, \sigma_k = \sigma_i \cap \sigma_j$.

This operation is used to order subsystems with respect to the proposed diagnostic algorithm. The inclusion relation \subseteq is used to organize the components with the lattice concept $\mathcal{L}(\Sigma, \subseteq)$ with a partial ordering relation. It is different from the concept of partially ordered set (poset) because the arrangement of elements is not based on sets but on partitions.

2.2 Diagnostic function

A basic diagnostic function is defined to help the diagnosis: the *check* function. Depending on the granularity, the *check* function is applied on a component, a subsystem or a partition. First, the *checkC* function is used to determine if a component is faulty or not. However, we do not know precisely how a unique component behaves regarding a fault. So we need to define the *checkS* function of a subsystem. The behaviour of a faulty subsystem may also not be sufficient to explain a fault. In fact, subsystems are interconnected making the system structure and the partitioning concept allows us to focus on different levels of abstraction that we call granularities. In our study, we only focus on faults with observable and measurable symptoms. These faults can only be localized by testing a functionality on a specific architecture. That is why, functional and structural partitions are used to decompose the system into testable partitions.

Definition 3. The *checkC* function of a component c_i is defined by:

$checkC : COMPS \rightarrow \{0, 1, -1\}$ s.a $checkC(c) = 0$ if the component c is faulty, $checkC(c) = 1$ if the component c is unfaulty and $checkC(c) = -1$ if the component state is unknown.

Definition 4. The *checkP* function of a partition P is defined by:

$checkP : \mathcal{P} \rightarrow \{0, 1, -1\}$ s.a $checkP(P) = 1 \Leftrightarrow \forall \sigma_i \in P, checkS(\sigma_i) = 1$, $checkP(P) = 0 \Leftrightarrow \exists \sigma_i \in P, checkS(\sigma_i) = 0$, and $checkP(P) = -1 \Leftrightarrow$ the checked value is unknown.

Some partitions cannot be checked. The set of possible checked partitions is *Cons*. It defined a constraint. A constraint *Cons* is a subset of \mathcal{P} s.a: $\forall P \in Cons, checkP(P) \neq -1$.

Once the *checkP* value of a partition is known, we have to define the *checkS* function of subsystems that are not singletons $\sigma_i \neq \{c_i\}$. If the partition is faulty, either it exists a component $c_i \in \sigma_i$ such as $checkC(c_i) = 0$, or the communication between the components in σ_i is faulty. This is modeled by $checkCom(\sigma_i) = 0$. If the partition is unfaulty, then all communications between the components in $\sigma_i \neq \{c_i\}$ are unfaulty and all singletons $\sigma_i = \{c_i\}$ are unfaulty.

Definition 5. The *checkCom* function of a subsystem $\sigma_i \subseteq COMPS$ is defined by:

$checkCom : \Sigma \rightarrow \{0, 1, -1\}$ s.a $checkCom(\sigma_i) = 1 \Leftrightarrow$ the communication between components in σ_i is unfaulty; $checkCom(\sigma_i) = 0 \Leftrightarrow$ the communications between components in σ_i is faulty.

To help the diagnosis of the system, we decompose it into subsystems and we introduce the *checkS* function of a subsystem $\sigma_i \subseteq COMPS$ defined by:

Definition 6. $checkS : \Sigma \rightarrow \{0, 1, -1\}$ s.a $checkS(\sigma_i) = 1 \Leftrightarrow \forall c_i \in \sigma_i, checkC(c_i) = 1 \wedge checkCom(\sigma_i) = 1$; $checkS(\sigma_i) = 0 \Leftrightarrow \exists c_i \in \sigma_i, checkC(c_i) = 0 \vee checkCom(\sigma_i) = 0$ and $checkS(\sigma_i) = -1 \Leftrightarrow \exists c_i \in \sigma_i, checkC(c_i) = -1 \wedge checkCom(\sigma_i) = -1$.

With the above definitions, it is now time to define the diagnosis problem. Given a system representation with the lattice concept $\mathcal{L}(\Sigma, \subseteq)$ and the set of constraints $Cons =$

$\{P \in \mathcal{P}, \text{check}P(P) \neq -1\}$, the problem is defined by the consistency between $\mathcal{L}(\Sigma, \subseteq)$ that contains the system representation, and $Cons$ that describes system issues.

Definition 7. *The problem formulation is to find the faulty components whose current state may explain the constraints. It is defined as a function $DIAG(\mathcal{L}(\Sigma, \subseteq))$ under the constraints $Cons$.*

There are two kinds of faults: the fault of a component C_i modeled with $\text{check}C(C_i) = 0$, and the communication fault of a subsystem $\sigma_i = \{C_i, C_j, \dots\}$ modeled with $\text{check}Com(\sigma_i) = 0$. With the P_1 partition, suppose that C_2 and C_3 are linked with an ARINC 429 link that is not working. The constraint is $\text{check}P(P_1) = 0$ because the global function is broken. The reason is that $\text{check}Com(\sigma_2) = 0$. Knowing that $\text{check}Com(\sigma_2) = 0$ for the P_1 functionality is giving the information to fix the system.

2.3 Diagnostic algorithm

It is now necessary to introduce a diagnostic method whose aim is to solve the above problem. The algorithm is based on the following proposition that extends the verification from the multiplication of partitions to partitions, see Proposition 1. Then, a functional verification is propagated from partitions to subsystems, and from subsystems to components.

Proposition 1. $\forall P, Q \in \mathcal{P}^2, \text{check}P(P \times Q) = 0 \Rightarrow \text{check}P(P) = 0 \wedge \text{check}P(Q) = 0$.

In order to increase the readability of the algorithm, it has been split into three: $DIAG(\mathcal{L}(\Sigma, \subseteq))$ is the main algorithm, it initializes the framework with the partitions of the system $\{p_i, i \in [1, n]\}$ and the constraints $Cons = \{P \in \mathcal{P}, \text{check}P(P) \neq x\}$.

$FindFaultyElements$ checks the partitions that are defined as a constraint. If the checked value of a partition p_{mult} is faulty (*resp.* unfauly), we add it to the faulty (*resp.* unfauly) partitions set P^- (*resp.* P^+), and every subsystem σ_i of the partition is possibly faulty (*resp.* unfauly), we add it in Σ^+ , (*resp.* Σ^-). If another partition p_{mult} can help to get more faulty or unfauly components, a new constraint is proposed and added to $NCons$.

$Verification$ is used to check the possible components that may be faulty, i.e include in F_c with the $\text{check}C$ function, and the communication of the subsystems in Σ^- with the $\text{check}Com$ function.

Two functions have been introduced: the $\text{check}P(p_i)$ value of a partition p_i and the $\text{check}Com(\sigma_i)$ of a subsystem. Their values can be automatically computed thanks to a program developed on the system to automate the diagnosis. This is performed by the GET function whose purpose is to model the computation of $\text{check}P(p_i)$ or $\text{check}Com(\sigma_i)$.

2.4 Formal example

In order to illustrate the problem formulation and the diagnostic algorithm, a formal example is provided. It is composed of eight components $\{C_i, i \in [1, 8]\}$ organized into three partitions:

$$P_1 = \{ \{C_1, C_2, C_3, C_4\}, \{C_5, C_6, C_7, C_8\} \},$$

$$P_2 = \{ \{C_1, C_2\}, \{C_3, C_4, C_5, C_6, C_7, C_8\} \},$$

$$P_3 = \{ \{C_1\}, \{C_2, C_4, C_6, C_8\}, \{C_3, C_5, C_7\} \}.$$

P_3 describes the topology of the system. P_1 and P_2 describe functionalities. We set the C_2 component as faulty. The idea is to combine the topology of the system with its functionalities to find the faulty component or subsystem. A choice

Algorithm 1: $DIAG(\mathcal{L}(\Sigma, \subseteq))$

Input: $d = \{p_i, i \in [1, n]\}$, $Cons = \{cons_i\}$

Output: $\Delta(Diagnosis)$

Global variables: *End*

F_c (faulty components), U_c (unfaulty components),

Σ^- (faulty subsystems), Σ^+ (unfaulty subsystems),

P^- (faulty partitions), P^+ (unfaulty partitions)

$\Delta, F_c, U_c, P^+, P^-, \Sigma^-, \Sigma^+ \leftarrow \{\}; End \leftarrow false;$

$NCons \leftarrow \{\};$

while $\neg End$ **do**

$FindFaultySubsystems(d, Cons);$

$Verification(F_c, \Sigma^-);$

if $\neg End$ **then**

foreach $p_i \in NCons$ **do**

 GET $\text{check}P(p_i)$

$Cons \leftarrow Cons \cup \{p_i\}$

Algorithm 2: $FindFaultyElements$

Input: $d = \{p_i\}$, $Cons = \{cons_i\}$

Outputs: $F_c, P^-, \Sigma^-, \Sigma^+$

foreach $(p_j, p_k) \in P^2 : p_i \neq p_j$ **do**

$p_{mult} \leftarrow p_j \times p_k$

if $p_{mult} \in Cons$ **then**

if $\text{check}P(p_{mult}) = 0$ **then**

$P^- \leftarrow P^- \cup \{p_i\}$

foreach $\sigma_i \in p_i$ **do**

foreach $c_k \in U_c$ **do**

$\sigma_i \leftarrow \sigma_i \setminus \{c_k\}$

if $\sigma_i = \{c_i\}$ **then**

$F_c \leftarrow F_c \cup \sigma_i$

else if $\sigma_i \notin \Sigma^+$ **then**

$\Sigma^- \leftarrow \Sigma^- \cup \{\sigma_i\}$

if $\text{check}P(p_{mult}) = 1$ **then**

$P^+ \leftarrow P^+ \cup \{p_i\}$

foreach $\sigma_i \in p_i$ **do**

if $\sigma_i = \{c_i\}$ **then**

$U_c \leftarrow U_c \cup \sigma_i$

else

$\Sigma^+ \leftarrow \Sigma^+ \cup \{\sigma_i\}$

if $p_{mult} \notin Cons$ **then**

if $\exists \{c_i\} \in p_{mult}$ **then**

if $\neg(c_i \in U_c \cup F_c)$ **then**

$NCons \leftarrow NCons \cup \{p_{mult}\}$

function is introduced to choose the next topology and the next functionality to be tested. It is guided by the minimum of tests to perform in order to fix the system. For a set of partitions \mathcal{P} , we define $Choose : \{\mathcal{P}\} \rightarrow \mathcal{P} \times \mathcal{P}$.

As the two functionalities are modeled by P_1 and P_2 , and the the topology is modeled by P_3 , we have two possibilities. We assume that P_2 is prior to P_1 , the first iteration is defined with $Choose(\mathcal{P}) = (P_1, P_3)$. We begin with $\text{check}P(P_1 \times P_3) = 0$, s.a $P_1 \times P_3 = \{ \{C_1\}, \{C_2, C_4\}, \{C_3\}, \{C_6, C_8\}, \{C_5, C_7\} \}$. The possible faulty component are C_1 and C_3 . We check the C_1 and C_3 components and

Algorithm 3: Verification

Inputs: F_c
Outputs: Δ, U_c, End
Initialization: $\sigma_+, \sigma_- \leftarrow I$;
foreach $c_i \in F_c$ **do**
 if $checkC(c_i) = 0$ **then**
 $\Delta \leftarrow \Delta \cup \{c_i\}$
 $End \leftarrow true$
 else
 $F_c \leftarrow F_c \setminus \{c_i\}$
 $U_c \leftarrow U_c \cup \{c_i\}$
foreach $\Sigma_i \in \Sigma^-$ **do**
 GET $checkCom(\Sigma_i)$
 if $checkCom(\Sigma_i) = 0$ **then**
 $\Delta \leftarrow \Delta \cup \{\Sigma_i\}$
 $End \leftarrow true$
 else
 $\Sigma^- \leftarrow \Sigma^- \setminus \{\Sigma_i\}$
 $\Sigma^+ \leftarrow \Sigma^+ \cup \{\Sigma_i\}$

find them as unfaulty, see Tables 1. The possible faulty subsystems are $\{C_2, C_4\}$, $\{C_6, C_8\}$ and $\{C_5, C_7\}$ and they are unfaulty. The diagnosis is not sufficient, we must relax the constraint $P_2 \times P_3$.

The second iteration is defined with $Choose(\mathcal{P}) = (P_2, P_3)$, s.a $P_2 \times P_3 = \{\{C_1\}, \{C_2\}, \{C_4, C_6, C_8\}, \{C_3, C_5, C_7\}\}$. We get $checkP(P_2 \times P_3) = 0$, the possible faulty components are C_1 and C_2 but C_1 has already been checked in the previous iteration. So, the possible faulty subsystems are $\{C_3, C_5, C_7\}$ and $\{C_4, C_6, C_8\}$. We check the C_2 component and find it as faulty. For this example, the computed faulty or unfaulty components is, see Table 2, C_2 in $P_2 \times P_3$.

If no components has been found faulty, the upper topological level is treated i.e subsystems: $\{C_2, C_4\}$, $\{C_6, C_8\}$, $\{C_5, C_7\}$, $\{C_4, C_6, C_8\}$ and $\{C_3, C_5, C_7\}$. Here, they are unfaulty.

Components	CheckC
C_1	1
C_2	-1
C_3	1
C_4	-1
C_5	-1
C_6	-1
C_7	-1
C_8	-1

 Table 1: Diagnostic results for components in $P_1 \times P_3$

The method has permitted to detect quickly the faulty component using functional partition and a structural partitioning. Thanks to this result, possible faults regarding either the topology or the functionality are checked.

3 The Automatic Test Benchmark

3.1 Avionics system

The avionics system of the NH90 helicopter is designed to support multiple hardware and software platforms from

Components	CheckC
C_1	1
C_2	0
C_3	1
C_4	-1
C_5	-1
C_6	-1
C_7	-1
C_8	-1

 Table 2: Diagnostic results for components in $P_2 \times P_3$

more than twelve national customers in over twenty different basic helicopter configurations. The NH90 Avionics System consists of two major subsystems: the CORE System and the MISSION System. A computer is the bus controller and manages each subsystem communications: the Core Management Computer (CMC) for the CORE System and the Mission Tactical Computer (MTC) for the MISSION System. Each computer is connected to one or both subsystems via a multiplex data bus (MIL-STD-1553), point to point connections (ARINC429) and serial RS-485 lines. Additional redundant computers are used as backup. One of the two CMC is the Bus Controller (BC) of the CORE multiplex data bus. The avionics system of the ATB is composed of fourteen computers and the above connections: two CMC: $c_1 = CMC1$ and $c_2 = CMC2$; two Plant Management Computer (PMC): $c_3 = PMC1$ and $c_4 = PMC2$; five Multifunction Display (MFD): $c_5 = MFD1$, $c_6 = MFD2$, $c_7 = MFD3$, $c_8 = MFD4$, $c_9 = MFD5$; two Display and Keyboard Unit (DKU): $c_{10} = DKU1$, $c_{11} = DKU2$; two IRS: $c_{12} = IRS1$, $c_{13} = IRS2$; one Radio Altimeter (RA): $c_{14} = RA$. Formally, $COMPS_{ATB} = \{c_i, i \in [1, 14]\}$. The avionics system under test $COMPS_{SUT}$ is a subsystem of $COMPS_{ATB}$. It is described Figure 1. $COMPS_{SUT} = \{c_1, c_2, c_3, c_4, c_5, c_{10}, c_{12}, c_{14}\}$. For the rest of the article, $COMPS_{SUT}$ will be the primary system under study.

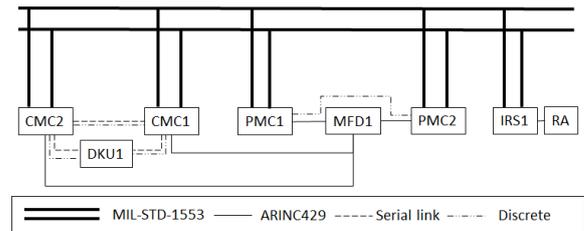


Figure 1: Architecture of the avionics subsystem

From	To	Messages	Subsystems
$DKU1$	$CMC1$	Mode on	$\sigma_{Serial1}$
$CMC1$	$IRS1$	Mode on	σ_{MIL}
$IRS1$	RA	Mode on	$\sigma_{NAV}; \sigma_{ARINC}$
RA	$IRS1$	Alert	$\sigma_{NAV}; \sigma_{ARINC}$
$IRS1$	$CMC1$	Alert	$\sigma_{MIL}; \sigma_{NAV}$
$CMC1$	$DKU1$	Alert	$\sigma_{Serial1}; \sigma_{NAV}$

Table 3: Messages

The PMC is used to monitor the status of all the avionics computers. It displays the alert informations on the MFD. We define the performances partition $p_{PERF} = \{\sigma_{PERF}, \sigma_{-PERF}\}$ with:

$$\sigma_{PERF} = \{PMC1, PMC2, RA, IRS1, MFD1\}$$

$\sigma_{-PERF} = \{CMC1, CMC2, DKU1\}$ and the navigation partition $p_{NAV} = \{\sigma_{NAV}, \sigma_{-NAV}\}$ with:

$$\sigma_{NAV} = \{RA, IRS1, MFD1\}$$

$$\sigma_{-NAV} = \{CMC1, CMC2, DKU1, PMC1, PMC2\}.$$

The test consists in the simulation of a high roll. Normally the RA should be deactivated above the value of forty degrees. The procedure contains the following actions: engage the RA with the DKU1; simulating a roll of 50 degrees; check that the RA functionality is deactivated on the DKU1. Several messages are sent to achieve this functionality, see Table 3, defining a data-flow for two messages: "Mode on" and "Alert" messages: from DKU1 to CMC1 via serial communication to activate the radioaltimeter's specific mode ("Mode on" message); from CMC1 to IRS1 via MIL-STD-1553 communication to relay the activation information; from IRS1 to RA via ARINC communication to send a request to the RA to get the roll angle; from RA to IRS1 via ARINC communication to send the response to the IRS that compute the angle; from IRS1 to CMC1 via ARINC communication, from CMC to DKU via serial communication to display the alert and disable the functionality ("Alert" message).

3.2 System Under Test (SUT) decomposition

The ATB is used to perform the realization of the avionics functions with the necessary equipments and a simulated environment needed to check the system specification.

The ATB is described as a structural decomposition with components subsets. These sets provide partitions of the whole system. We define subsystems σ_i and the partitions p_i with regards to the connections of the avionics system of Figure 1, the serial communication:

$$\sigma_{Serial1} = \{CMC1, CMC2, DKU1\}$$

$$\sigma_{Serial2} = \{PMC1, PMC2\}$$

$$\sigma_{-Serial} = \{MFD1, IRS1, RA\}$$

$$p_{Serial} = \{\sigma_{Serial1}; \sigma_{Serial2}; \sigma_{-Serial}\}$$

the ARINC communications:

$$\sigma_{ARINC} = \{CMC1, CMC2, PMC1, PMC2, MFD1, IRS1, RA\}$$

$$\sigma_{-ARINC} = \{DKU1\}$$

$$p_{ARINC} = \{\sigma_{ARINC}; \sigma_{-ARINC}\}$$

the MIL-STD-1553 communications:

$$\sigma_{MIL} = \{CMC1, CMC2, PMC1, PMC2, IRS1\}$$

$$\sigma_{-MIL} = \{MFD1, DKU1, RA\}$$

$$p_{MIL} = \{\sigma_{MIL}; \sigma_{-MIL}\}$$

The above partitions describe the topology of the problem. We classify the partitions into two categories: functional partitions and communication partitions. The functional partitions contain the subsystems that compute and send the informations. The communication partitions contain the subsystems that relay these informations. In our example, the navigation functionality is tested. Functional partition are: $\{p_{NAV}, p_{PERF}\}$, connection partitions are: $\{p_{MIL}, p_{Serial}, p_{ARINC}\}$. We need to define additional partitions that can be checked with the *check* function on the system thanks to this representation:

$$p_{NAV.MIL} = p_{NAV} \times p_{MIL} = \{\{MFD1, RA\}; \{IRS1\}; \{CMC1, CMC2, PMC1, PMC2\}; \{DKU1\}\};$$

$$p_{NAV.Serial} = p_{NAV} \times p_{Serial} = \{\{CMC1, CMC2,$$

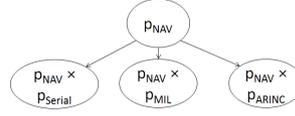


Figure 2: Navigation function decomposition with $d_{protocol}$

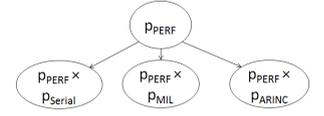


Figure 3: Performance function decomposition with $d_{protocol}$

$DKU1\}; \{PMC1, PMC2\}; \{MFD1, IRS1, RA\}\};$
 $p_{NAV.ARINC} = p_{NAV} \times p_{ARINC} = \{\{MFD1, IRS1, RA\}; \{CMC1, CMC2, PMC1, PMC2\}; \{DKU1\}\}.$

The performance function can give insights about the fault. We compute the partitions with this functionality:

$$p_{PERF.MIL} = p_{PERF} \times p_{MIL} = \{\{MFD1, RA\}; \{DKU1\}; \{CMC1, CMC2\}; \{PMC1, PMC2, IRS1\}\}$$

$$p_{PERF.Serial} = p_{PERF} \times p_{Serial} = \{\{CMC1, CMC2, DKU1\}; \{PMC1, PMC2\}; \{MFD1, IRS1, RA\}\}$$

$$p_{PERF.ARINC} = p_{PERF} \times p_{ARINC} = \{\{PMC1, PMC2, MFD1, IRS1, RA\}; \{CMC1, CMC2\}; \{DKU1\}\}.$$

Those partitions will serve to improve the diagnosis.

3.3 Outlooks about the decompositions

We describe an iterative method to update the diagnostic result by providing new topologies of the system. We need to get precise observations to find the faulty components. The subsystems are computed with the framework of the previous section.

Given the components, the messages sent between them, and the protocol of these messages, we can obtain an overview of the system decomposition: p_{SUT} can be decomposed into $d_{protocol} = \{p_{SUT} \times p_{MIL}; p_{SUT} \times p_{Serial}; p_{SUT} \times p_{ARINC}\}$. This hierarchical structure is provided with a dependency graph, see Figures 2 and 3.

The following partitions are used:

$$\sigma_{com1} = \{\{DKU1, CMC1, IRS1, RA\}\};$$

$$\sigma_{-com1} = \{\{MFD1, CMC2, PMC1, PMC2\}\};$$

$$p_{com1} = \{\sigma_{com1}, \sigma_{-com1}\}.$$

The path of the informations "RA mode on" and "RA alert" on copilot side defines another decomposition: $\sigma_{com2} = \{\{CMC2, IRS1, RA, DKU1\}\}; \sigma_{-com2} = \{\{MFD1, CMC1, PMC1, PMC2\}\}; p_{com2} = \{\sigma_{com2}, \sigma_{-com2}\}.$

We describe the decomposition $d_{com} = \{p_{com1}, p_{com2}\}$ on Figures 4 and 5. We compute partitions with the navigability functionality and this structural decomposition:

$$p_{NAV.com1} = p_{NAV} \times p_{com1} = \{\{RA, IRS1\}; \{MFD1\}; \{CMC1, DKU1\}; \{CMC2, PMC1, PMC2\}\};$$

$$p_{NAV.com2} = p_{NAV} \times p_{com2} = \{\{RA, IRS1\}; \{DKU1, CMC2\}; \{MFD1\}; \{CMC1, PMC1, PMC2\}\};$$

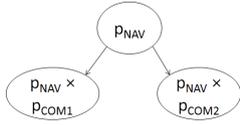
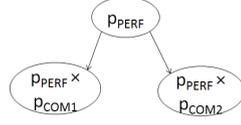
$$p_{PERF.com1} = p_{PERF} \times p_{com1} = \{\{RA, IRS1\}; \{CMC2\}; \{CMC1, DKU1\}; \{MFD1, PMC1, PMC2\}\};$$

$$p_{PERF.com2} = p_{PERF} \times p_{com2} = \{\{RA, IRS1\}; \{DKU1, CMC2\}; \{CMC1\}; \{MFD1, PMC1, PMC2\}\}.$$

4 Illustration of the Meta-Diagnostic Approach

4.1 Application of the meta-diagnosis approach

An iterative approach is very helpful in this case of distributed systems since diagnosis can use new subsystems and partitions. The results of the diagnosis are re-injected in the upper system to refine the results.


 Figure 4: Navigation function decomposition with d_{com}

 Figure 5: Performance function decomposition with d_{com}

The first symptom is the misbehavior of the navigation functionality. We describe the iterations of the algorithms with two topologies. We have launched the meta-diagnostic algorithm with the topology: $d_{NAV.protocol} = \{P_{NAV.MIL}, P_{NAV.ARINC}, P_{NAV.SERIAL}\}$ and $d_{NAV.com} = \{P_{NAV.com1}, P_{NAV.com2}\}$. The constraint is $CONS = \{checkP(p_i), \forall p_i \in d_{NAV.protocol} \cup d_{NAV.com}\}$. The iterations of the algorithms are described in Tables 4, and 5.

p_i	$checkP(p_i)$	U_c	F_c
$P_{NAV.ARINC}$	0	\emptyset	$\{DKU1\}$
$P_{NAV.SERIAL}$	1	\emptyset	$\{DKU1\}$
$P_{NAV.MIL}$	0	\emptyset	$\{IRS1, DKU1\}$

 Table 4: Iterations of *CheckMultiplicationPartition* with $d_{protocol}$

The third step gives a state of the components in F_c set that can be faulty: $DKU1$ and $IRS1$ in Table 5. If the components are faulty, this may explain the system behavior and the algorithm ends. At the same time, the communications of subsystems in Σ^- can be faulty. They are checked in Table 6.

c_i	$checkC(c_i)$	F_c	U_c
$DKU1$	1	$\{IRS1\}$	$\{DKU1\}$
$IRS1$	0	$\{IRS1\}$	$\{DKU1\}$

 Table 5: Iterations of the *CheckComponents* with $d_{protocol}$

Subsystems	checkCom	Partition
$\{MFD1, RA\}$	1	$P_{NAV.ARINC}$
$\{CMC1, CMC2, PMC1, PMC2\}$	1	$P_{NAV.ARINC}$

Table 6: Diagnostic results for subsystems

The $IRS1$ is not faulty, the algorithm is relaunched with $U_c = \{DKU1, IRS1\}$ and the other decomposition $d_{com} = \{P_{NAV.com1}, P_{NAV.com2}\}$. The algorithm iterations are described in Tables 7 and 8.

Once $checkP(p_{NAV.com2}) = 1$, we deduce that $MFD1$ is not faulty, see Table 7. At this step, the unfaulty components are $\{DKU1, IRS1, MFD1\}$, and the diagnosis is $\{RA\}$.

Here the RA is faulty with $p_{NAV.com1}$, and the algorithm ends. The solution is RA for $p_{NAV.com1}$. The data flow of the messages are checked as the impacted connections, wiring and, routing. The system specificities of the communication modeled with $com1$ five clues of the possible

p_i	$checkP(p_i)$	U_c	F_c
$p_{NAV.com1}$	0	$\{DKU1, IRS1\}$	$\{RA, MFD1\}$
$p_{NAV.com2}$	1	$\{DKU1, IRS1, MFD1\}$	$\{RA\}$

 Table 7: Iterations of *CheckMultiplicationPartition* with d_{com}

Subsystems	checkCom	Partition
$\{RA, IRS1\}$	1	$P_{NAV.com1}$
$\{CMC1, DKU1\}$	1	$P_{NAV.com1}$
$\{CMC2, PMC1, PMC2\}$	1	$P_{NAV.com1}$

 Table 8: Diagnostic results of subsystems with $p_{NAV.com1}$

faults. Thanks to the impacted functionality, we know that only messages concerning the IRS roll are concerned. At this stage, the simulation of the message or the bad connection of the IRS are the two main solutions.

4.2 Application with updated constraints

We describe a new problem: the navigation functionality and the performance function do not behave normally. The new constraint is $CONS = \{checkP(p_i), \forall p_i \in d_{NAV.protocol} \cup d_{NAV.com} \cup d_{PERF.protocol} \cup d_{PERF.com}\}$. The algorithm is loaded from *CheckMultiplicationPartition* with the decomposition d_{com} . The algorithm iterations are described in Table 9. Once $checkP(p_{PERF.com2}) = 1$, we deduce that $CMC1$ is not faulty. We continue with $d_{protocol}$ knowing the $CMC1$ is not faulty in Table 10. We deduce that we have to check $DKU1$ and $CMC2$.

p_i	$checkP(p_i)$	U_c	F_c
$p_{PERF.com1}$	0	\emptyset	$\{CMC2\}$
$p_{PERF.com2}$	1	$\{CMC1\}$	$\{CMC2\}$

 Table 9: Algorithm 2's iterations with d_{com}

p_i	$checkP(p_i)$	U_c	F_c
$p_{PERF.ARINC}$	0	$\{CMC1\}$	$\{DKU1, CMC2\}$
$p_{PERF.SERIAL}$	1	$\{CMC1\}$	$\{DKU1, CMC2\}$
$p_{PERF.MIL}$	0	$\{CMC1\}$	$\{DKU1, CMC2\}$

 Table 10: Iterations of *CheckMultiplicationPartition* with $d_{protocol}$

At this state, we check the components on the system. Since the reparation of $CMC2$ has fixed the problem, we conclude that $CMC2$ has been faulty. We also check the $DKU1$ configuration, and find nothing. The diagnosis is $\Delta = \{CMC2\}$.

The evolution of the number of faulty and unfaulty components is reviewed on figure 6. As expected, the number of unfaulty components is increasing with new tests, i.e tests

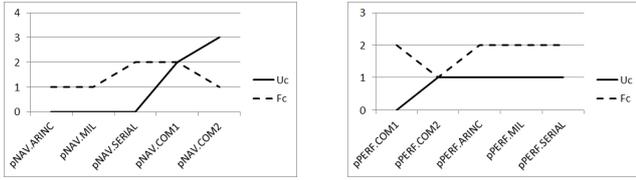


Figure 6: Evolution of the number of faulty and unfaulty components

of partitions. It reveals that the algorithm is converging to a solution because the number of components is limited.

5 Software implementation

5.1 Diagnostic software architecture

The algorithms are implemented in a spy software of ARINC and MIL-STD-1553 buses, see Figure 7. They are developed using C++ for effective diagnosis, and to be implemented in the AIRBUS software. The user interfaces are developed with Java 1.7 and the *Swing* Graphical User Interface (GUI) widget toolkit. The architecture of the diagnostic

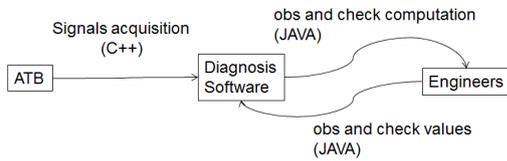


Figure 7: Data flow of the diagnosis software

framework has been adapted to the ATB specificities as described with the Model-View-Controller (MVC) paradigm on Figure 8. Three main objects are defined for the Model: the *Component*, the *Set*, and the *Partition* objects. Four main objects are defined in the View to define specific panels: the *diagnosisPanel*, the *constraintsPanel*, the *initialStatePanel* and the *resultsPanel* objects. The model is implemented with the *ArrayList* class. It is used to define the list of components, the subsystems and the list of partitions. eXtensible Markup Language (XML) files have been used to describe the system structure. The *Controller* dispatches the user requests and selects the panels for presentation. The diagnosis algorithm is implemented in it. A GUI is provided for handling user inputs such as partitions check values and components observations values.

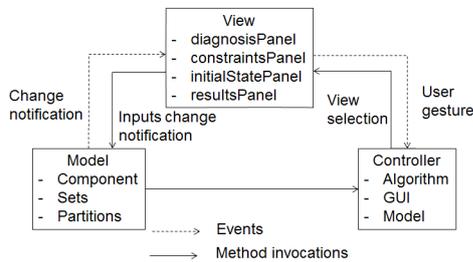


Figure 8: Architecture of the diagnosis software

5.2 User interfaces

The panels are displayed one after the others for each step of the algorithm defined in the *Controller*. The

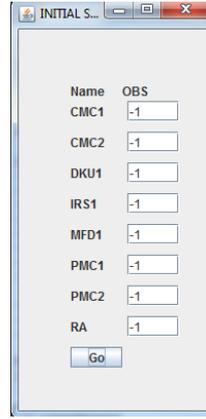


Figure 9: Initial state of the diagnosis

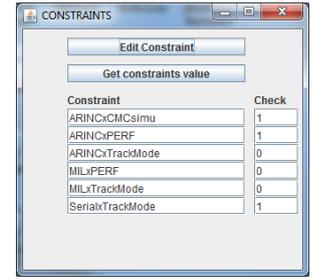


Figure 10: State of the constraints

initialStatePanel panel, Figure 9 defines the status of equipments before launching the diagnosis and a button the run the algorithm. The *check* values computed by the algorithm defined in the *Controller* are provided to the operator in Figure 11. The *constraintsPanel* panel lets to edit and update constraints, see Figure 10. The result of the diagnostic algorithm is provided on Figures 11. It gives the faulty components (observation equal to zero) and the impacted functionality. If a component is suspected, the data

Name	OBS	Partition	Check
CMC1	-1		
CMC2	0	COMonexPERF=[CMC1,DKU1];[RA,IRS1];[CMC2];[PMC1,PMC2,MFD1]	0
DKU1	0	ARINCxTrackMode=[CMC1,CMC2,PMC1,PMC2];[RA,IRS1,MFD1];[DKU1] MILxTrackMode=[CMC1,CMC2,PMC1,PMC2];[IRS1];[DKU1];[RA,MFD1] MILxPERF=[CMC1,CMC2];[PMC1,PMC2,IRS1];[DKU1];[RA,MFD1]	0
IRS1	0	MILxTrackMode=[CMC1,CMC2,PMC1,PMC2];[IRS1];[DKU1];[RA,MFD1]	0
MFD1	1	COMonexTrackMode=[CMC1,DKU1];[RA,IRS1];[CMC2,PMC1,PMC2];[MFD1] COMonxTrackMode=[CMC2,DKU1];[RA,IRS1];[CMC1,PMC1,PMC2];[MFD1]	1
PMC1	-1		
PMC2	-1		
RA	0	MILxTrackMode=[CMC1,CMC2,PMC1,PMC2];[IRS1];[DKU1];[RA,MFD1] MILxPERF=[CMC1,CMC2];[PMC1,PMC2,IRS1];[DKU1];[RA,MFD1]	0

Figure 11: Diagnosis results

flow of the functional chain described by the partition must be checked. As described in the case study, it gives insights about the possible connections, wiring and, routing that can be wrong.

We compute the results $\Delta = \{ IRS1, DKU1, CMC2, RA \}$ and display them on Figure 11. If some components are unfaulty, we can update their status in Figure 9. The algorithm is relaunched using the "GO" button in Figure 9. The good diagnosis rate is evaluated on Figure 12. It is defined by the number of faulty components that the operator has to fix over the number of proposed faulty components.

5.3 Discussion

We have proposed a solution for the diagnosis of a complex system in aeronautics based on the MBD paradigm and the

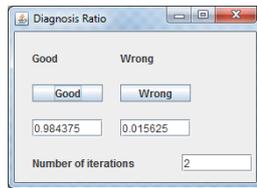


Figure 12: Good diagnosis rate

lattice concept. It is an other solution for the meta-diagnosis problem as described in [5] since we consider the test system environment as the main system. Belard has extended the framework, here we use the original one with the lattice concept to represent the system description. It is also provided a diagnostic algorithm implemented on the system to evaluate our method. Since hundreds of diagnosis are possible on the ATB, since it is not possible to check all those possibilities, we have introduced a methodology for the ATB diagnosis that reduce the number of iterations to get the diagnosis. We have upgraded the applications of MBD for avionics systems evaluated in [4] and [2]. It is proposed the integration and evaluation of a diagnostic algorithm for an ATB, taking the test systems environment into account. It differs from other applications of MBD like [8] because the model decomposition is driven by the test systems specificities that are represented with the lattice concept.

6 Conclusion

This paper extends the MBD approach to propose a diagnostic software that is developed for the diagnosis of test systems. The current framework is based on the lattice decomposition and is used to model a test system. First, the lattice decomposition has been used to decompose the system into its functionalities and connections. The second contribution consists in the proposal of an algorithm that reduce the diagnostic ambiguity. The lattice description has been implemented with JAVA native packages. The software architecture and diagnostic iterations are provided for a formal example and an industrial case study. The diagnostic algorithm has shown to reduce the number of faulty candidates. The results is either faulty equipment or a group of equipments with the associated system functionality that is unable to meet its goal. Together, they are sufficient to point out the reparations that will fix the system. The tests on the Avionics Test Systems in AIRBUS HELICOPTERS have shown good results. The development of models may confront our solution to many others real problems. In future works, algorithms will be improved with adaptable decompositions and automatic tests. Furthermore, as the method is generic, we want to demonstrate the validity of our method for others test systems used in AIRBUS HELICOPTERS.

References

- [1] Canh Ly, Kwok Tom, Carl S. Byington, Romano Patrick, and George J. Vachtsevanos. Fault Diagnosis and Failure Prognosis for Engineering Systems: A Global Perspective. In *Proceedings of the Fifth Annual IEEE International Conference on Automation Science and Engineering, CASE'09*, pages 108–115, Piscataway, NJ, USA, 2009. IEEE Press.
- [2] Arnaud Lefebvre, Zineb Simeu-Abazi, Jean-Pierre Derain, and Mathieu Glade. Diagnostic of the avionics equipment based on dynamic fault tree. In *Proceedings of the IFAC-CEA conference*, October 2007.
- [3] Denis Berdjag, Jérôme Cieslak, and Ali Zolghadri. Fault detection and isolation of aircraft air data/inertial system. pages 317–332. EDP Sciences, 2013.
- [4] Fabien Kuntz, Stéphanie Gaudan, Christian Sannino, Éric Laurent, Alain Griffault, and Gérald Point. Model-based diagnosis for avionics systems using minimal cuts. *DX 2011 22nd International Workshop on Principles of Diagnosis*, 2011.
- [5] Nuno Belard, Yannick Pencole, and Michel Combaucou. A theory of meta-diagnosis: reasoning about diagnostic systems. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence, IJCAI'11*, pages 731–737, Barcelona, Catalonia, Spain, 2011.
- [6] Denis Berdjag, Vincent Cocquempot, Cyrille Christophe, Alexey Shumsky, and Alexey Zhirabok. Algebraic approach for model decomposition: Application for fault detection and isolation in discrete-event systems. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 21(1):109–125, March 2011.
- [7] Quang-Huy Giap, Stéphane Ploix, and Jean-Marie Flaus. Managing Diagnosis Processes with Interactive Decompositions. In *Artificial Intelligence Applications and Innovations III*, IFIP International Federation for Information Processing, pages 407–415. 2009.
- [8] Belarmino Pulido, Carlos Alonso-González, Anibal Bregon, Alberto Hernández Cerezo, and David Rubio. DXPCS: A software tool for consistency-based diagnosis of dynamic systems using Possible Conflicts. *25th Annual Workshop Proceedings, DX-14*, 2014.
- [9] Veronique Delcroix, Mohamed-Amine Maalej, and Sylvain Piechowiak. Bayesian Networks versus Other Probabilistic Models for the Multiple Diagnosis of Large Devices. *International Journal on Artificial Intelligence Tools*, 16(3):417–433, 2007.
- [10] Mattias Krysander, Jan Aslund, and Erik Frisk. A Structural Algorithm for Finding Testable Sub-models and Multiple Fault Isolability Analysis. *21st Annual Workshop Proceedings, DX-10*, 2010.
- [11] Ronan Cossé, Denis Berdjag, David Duvivier, Sylvain Piechowiak, and Christian Gaurel. Meta-Diagnosis for a Special Class of Cyber-Physical Systems: the Avionics Test Benches. In *The 28th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, [Accepted]*, IEA/AIE 2015, Seoul, Korea, 2015.
- [12] Johan de Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.
- [13] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.
- [14] Randall Davis and Walter C. Hamscher. Model-Based Reasoning: Troubleshooting. pages 297–346, July 1988. San Francisco, CA, USA.

SAT-Based Abductive Diagnosis

Roxane Koitz^{1*} and Franz Wotawa¹

¹Graz University of Technology, Graz, Austria

e-mail: {rkoitz, wotawa}@ist.tugraz.at

Abstract

Increasing complexity and magnitude of technical systems demand an accurate fault localization in order to reduce maintenance costs and system down times. Resting on solid theoretical foundations, model-based diagnosis provides techniques for root cause identification by reasoning on a description of the system to be diagnosed. Practical implementations in industries, however, are sparse due to the initial modeling effort and the computational complexity. In this paper, we utilize a mapping function automating the modeling process by converting fault information available in practice into propositional Horn logic sentences to be used in abductive model-based diagnosis. Furthermore, the continuing performance improvements of SAT solvers motivated us to investigate a SAT-based approach to abductive diagnosis. While an empirical evaluation did not indicate a computational benefit over an ATMS-based algorithm, the potential to diagnose more expressive models than Horn theories encourages future research in this area.

1 Introduction

Fault identification of technical systems is becoming increasingly difficult due to their rising complexity and scale. Economic and safety considerations have put accurate diagnosis not only into research focus but has led to a growing interest in practice as well.

Model-based diagnosis has been presented as a method to derive root causes for observable anomalies utilizing a description of the system to be diagnosed [1, 2]. Reiter [1] proposed a component-oriented model encompassing the correct system behavior and structure. Discrepancies, i.e. conflicts, arise when the observed and expected system performance diverge. Based on the minimal conflict sets, root causes for the inconsistencies are obtained by hitting set computation. Hence, fault diagnosis is a two step process, where first contradicting assumptions on component health, given a set of symptoms and the model, are identified. Then the sets intersecting all conflict sets are computed which

constitute the diagnoses. At the same time [2] presents the General Diagnosis Engine (GDE) for multiple fault identification, drawing on the connection between inconsistencies and causes as well. Their approach employs an assumption-based truth maintenance system (ATMS) to detect conflicts and thereon compute diagnoses. Over the years much work has concentrated on model-based diagnosis applications in various domains, such as space probes [3] or the automotive industry [4].

Besides the consistency-based approach, a second method emerged within the field of model-based diagnosis, which exploits the concept of entailment to infer explanations for given observables. While related to the more traditional technique based on consistency, abductive model-based diagnosis requires a system formalization representing faults and their manifestations [5].

Even though based on a well-defined theory, a widespread acceptance of model-based diagnosis among industries has not been accounted for yet. Two main contributing factors can be identified: the initial model development and the computational complexity of diagnosis [6]. In order to diminish the modeling effort, [7] formulates a conversion of failure assessments available in practice into a propositional logic representation suitable for abductive diagnosis. Failure mode and effect analysis (FMEA) is an established reliability evaluation method utilized in various industrial fields. It considers possible component faults as well as their implications on the system's behavior [8]. Whereas there has been extensive research on the automatic generation of FMEAs from system models [9], we argue in favor of the inverse process. As these assessments report on failures and how they reveal themselves in the artifact's behavior, they provide knowledge requisite for abductive reasoning. In this paper, we present a compilation of FMEAs to models which can be used in abductive diagnosis.

Apart from discovering inconsistencies, an ATMS is capable of inferring abductive diagnoses. However, it may face computational challenges and is restricted to operate on propositional Horn clauses. In the case of the models we are extracting from the FMEAs, this is not a limitation so far. Nevertheless, as we anticipate to exploit more expressive representations, a different approach is required.

The performance of Boolean satisfiability (SAT) solvers has improved immensely over the last years and

*Authors are listed in alphabetical order.

several applications of SAT solvers in practice have proven successful. Furthermore, we are able to encode a greater variety of models in SAT. Thus, we propose a SAT-based approach to abductive diagnosis and empirically compare its performance to a procedure dependent on an ATMS.

The remainder of this paper is structured as follows. After formally providing the theoretical background on abductive diagnosis as well as relevant definitions in the context of SAT, we formulate the modeling process based on FMEAs and give information on the properties of the obtained system descriptions. In Section 5 we describe our SAT-based approach to abductive diagnosis and present an algorithm computing explanations for a given abduction problem. An empirical evaluation comparing our method to an ATMS-based diagnosis engine follows in Section 6. Subsequently, we provide some concluding remarks and give an outlook on future research possibilities.

2 Related Work

Mechanizing logic-based abduction has been an active research field for several decades with different approaches for generating explanations emerging, such as proof tree completion [10] and consequence finding [11]. While the former exploits a refutation proof involving hypotheses, the latter computes causes as logical consequences of the theory. As resolution is not consequence finding complete, [12] devised a procedure based on linear resolution which is sound and complete for consequence finding for propositional as well as first order logic.

While the number of practical applications in the context of abductive model-based diagnosis is rather small, in [13] the authors describe abductive reasoning in environmental decision support systems.

Most recently [14] present a SAT encoding for consistency-based diagnosis. The system description is compiled into a Boolean formula, such that the formula's satisfying assignments correspond to the solutions of the diagnosis problem. Based on the encoding, a SAT solver directly computes the diagnoses. In order to improve the solver's performance, the authors utilize several preprocessing techniques. An empirical comparison of their approach to other model-based diagnosis algorithms indicates that their SAT encoding yields performance benefits. Contrasting these results, [15] propose a translation to Max-SAT which could not outperform the stochastic model-based diagnosis algorithm SAFARI [16].

In [17] the authors present an algorithm which ties constraint solving to diagnosis, thus renders the detection of inconsistencies and subsequent hitting set computation unnecessary. Another direct approach by [18] computes minimal diagnoses for over-constrained problems by finding the sets of constraints to be relaxed in order to restore consistency. For Boolean formulas, those relaxations correspond to Minimal Correction Subsets (MCSes). Their hitting set dual, minimal unsatisfiable subsets (MUSes), constitute the set of subformulas explaining the unsatisfiability, i.e. refer to conflicts. While there are several algorithms for efficiently computing MCSes, most recently [19] develop three techniques for reducing the number of SAT solver

calls for existing methods as well as a novel algorithm for MCSes computation.

As stated by [20] the complexity of abduction suspends of a polynomial-time transformation to SAT. Thus, in their work the authors present a fixed-parameter tractable transformation from propositional abduction to SAT exploiting backdoors and describe how to use their transformations to enumerate all solutions for a given abduction instance.

3 Preliminaries

This section provides a brief introduction to abductive model-based diagnosis. In particular, we describe the propositional Horn clause abduction problem (PHCAP) which provides the basis for our research. Note that throughout the paper we consider the closed-world assumption. In addition to the background on abductive model-based diagnosis, we formally define MUSes and MCSes.

3.1 Abductive Diagnosis

In contrast to the traditional consistency-based approach, abductive model-based diagnosis depends on a stronger relation between faults and observable symptoms, namely entailment. Hence, whereas consistency-based diagnosis reasons on the description of the correct system operation, abductive reasoning requires the model to capture the behavior in presence of a fault. By exploiting the notion of entailment and the causal links between defects and their corresponding effects, we can reason about explanations for observed anomalies. In general, abductive diagnosis is an NP-hard problem. However, there are certain subsets of logic, such as propositional definite Horn theory, which are tractable [21]. On these grounds we consider the PHCAP as defined in [22], which represents the connections between causes and effects as propositional Horn sentences. Similar to [22], we define a knowledge base as a set of Horn clauses over a finite set of propositional variables.

Definition 1 (Knowledge base (KB)). *A knowledge base (KB) is a tuple (A, Hyp, Th) where A denotes the set of propositional variables, $Hyp \subseteq A$ the set of hypotheses, and Th the set of Horn clause sentences over A .*

The set of hypotheses contains the propositions, which can be assumed to either be true or false and refer to possible causes. In order to form an abduction problem, a set of observations has to be considered for which explanations are to be computed.

Definition 2. (Propositional Horn Clause Abduction Problem (PHCAP)) *Given a knowledge base (A, Hyp, Th) and a set of observations $Obs \subseteq A$ then the tuple (A, Hyp, Th, Obs) forms a Propositional Horn Clause Abduction Problem (PHCAP).*

Definition 3 (Diagnosis; Solution of a PHCAP). *Given a PHCAP (A, Hyp, Th, Obs) . A set $\Delta \subseteq Hyp$ is a solution if and only if $\Delta \cup Th \models Obs$ and $\Delta \cup Th \not\models \perp$. A solution Δ is parsimonious or minimal if and only if no set $\Delta' \subset \Delta$ is a solution.*

A solution to a PHCAP is equivalent to an abductive diagnosis, as it comprises the set of hypotheses

explaining the observations. Even though Definition 3 does not impose the constraint of minimality on a solution, in practice only parsimonious explanations are of interest. Hence, we refer to minimal diagnoses simply as diagnoses. Notice that finding solutions for a given PHCAP is NP-complete [22].

As aforementioned an ATMS derives abductive explanations for propositional Horn theories, thus it can be utilized to find solutions to a PHCAP. Based on a graph structure where hypotheses, observations, and contradiction are represented as nodes, the Horn clause sentences defined in Th determine the directed edges in the graph. Each node is assigned a label containing the set of hypotheses said node can be inferred from. By updating the labels, the ATMS maintains consistency.

Algorithm `abductiveExplanations` exploits an ATMS and returns consistent abductive explanations for a set of observations [23]. In case the observation consists of a single effect, the label of the corresponding proposition already contains the abductive diagnoses. To account for multiple observables, i.e. $Obs = \{o_1, o_2, \dots, o_n\}$, an individual implication is added, such that $o_1 \wedge o_2 \dots \wedge o_n \rightarrow obs$, where obs is a new proposition not yet considered in A . Every set contained in the label of obs constitutes a solution to the particular PHCAP.

Algorithm 1 `abductiveExplanations` [23]

procedure	ABDUCTIVEEXPLANATIONS
(A, Hyp, Th, Obs)	
Add Th to $ATMS$	
Add $(\bigwedge_{o \in Obs} o \rightarrow obs)$ to $ATMS$ $\triangleright obs \notin A$	
return the label of obs	
end procedure	

3.2 Minimal Unsatisfiable Subset and Minimal Correction Subset

We assume standard definitions for propositional logic [24]. A propositional formula ϕ in CNF, defined over a set of Boolean variables $X = \{x_1, x_2, \dots, x_n\}$, is a conjunction of m clauses (C_1, C_2, \dots, C_m) . A clause $C_i = (l_1, l_2, \dots, l_k)$ is a disjunction of literals, where each literal l is either a Boolean variable or its complement. A truth assignment is a mapping $\mu : X \Rightarrow \{0, 1\}$ and a satisfying assignment for ϕ is a truth assignment μ such that ϕ evaluates to 1 under μ . Given a formula ϕ , the decision problem SAT consists of deciding whether there is a satisfying assignment for the formula.

In case ϕ is unsatisfiable there are subsets of ϕ , which are of special interest in the diagnosis context, namely the MUSes and MCSes. A Minimal Unsatisfiable Subset (MUS) comprises a subset of clauses which cannot be satisfied simultaneously. Notice that every proper subset of MUS is satisfiable. A Minimal Correction Subset (MCS) is the set of clauses which corrects the unsatisfiable formula, i.e. by removing any MCS the formula becomes satisfiable.

Given an unsatisfiable formula ϕ , an MUS and MCS are defined as follows [25]:

Definition 4. (Minimal Unsatisfiable Subset (MUS)) A subset $U \subseteq \phi$ is an MUS if U is unsatisfiable and $\forall C_i \in U, U \setminus \{C_i\}$ is satisfiable.

Definition 5. (Minimal Correction Subset (MCS)) A subset $M \subseteq \phi$ is an MCS if $\phi \setminus M$ is satisfiable and $\forall C_i \in M, \phi \setminus (M \setminus \{C_i\})$ is unsatisfiable.

Since an MCS is a set of clauses correcting the unsatisfiable formula when removed, a single clause of an MUS is an MCS for this MUS. Note that the hitting set duality of MUSes and MCSes has been established [26].

Example. Consider the unsatisfiable formula ϕ in CNF.

$$\phi = \overbrace{(-a \vee -b \vee c)}^{C_1} \wedge \overbrace{(-c \vee d)}^{C_2} \wedge \overbrace{(c)}^{C_3} \wedge \overbrace{(-d)}^{C_4}$$

It is apparent that the combination of clauses C_2, C_3 and C_4 results in ϕ being unsatisfiable, hence

$$\text{MUSes}(\phi) = \{\{C_2, C_3, C_4\}\}.$$

By hitting set computation we arrive at the following set of MCSes:

$$\text{MCSes}(\phi) = \{\{C_2\}, \{C_3\}, \{C_4\}\}.$$

Removing any MCS of ϕ results in the formula being satisfiable.

It is worth noticing that utilizing subsets of unsatisfiable formulas has been proposed in regard to consistency-based diagnosis. In this context, a diagnosis is defined as the set of components which assumed faulty retains the consistency of the system. Thus, a consistency-based diagnosis corresponds to an MCS. For instance, [18] presents a direct diagnosis method computing MCSes for over-constrained systems. In conflict-directed algorithms, as proposed by Reiter [1], the minimal conflicts, arising from the deviations of the modeled to the experienced behavior, equate to the MUSes. In Section 5 we discuss our abductive diagnosis approach based on MUSes and MCSes.

4 Modeling Methodology

As mentioned before model-based diagnosis depends on a formal description of the system to be examined. The generation of appropriate models, however, is still an issue preventing a wide industrial adoption, since the modeling process is time-consuming and typically demanding for system engineers.

Therefore, we present a modeling methodology relying on FMEAs available in practice. An FMEA comprises a systematic component-oriented analysis of possible faults and the way they manifest themselves in the artifact's behavior and functionality [8]. This type of assessment is gaining importance and has become a mandatory task in certain industries, especially for systems that require a detailed safety analysis. Due to the knowledge capturing the causal dependencies between specific fault modes and symptoms, an FMEA provides information suitable for abductive reasoning [7].

Definition 6 (FMEA). An FMEA is a set of tuples (C, M, E) where $C \in COMP$ is a component, $M \in MODES$ is a fault mode, and $E \subseteq PROPS$ is a set of effects.

Running Example. In order to illustrate our modeling process, we use the converter of an industrial wind turbine as our running example [27]. Table 1

illustrates a simplified FMEA neglecting all parts affiliated with reliability analysis, such as severity ratings. Each row specifies a particular failure mode, (i.e. Corrosion, Thermo-mechanical fatigue (TMF) or High-cycle fatigue (HCF)) of a subsystem and determines its corresponding symptoms, such as $P_turbine$ referring to a deviation between expected and measured turbine power output.

Component	Fault Mode	Effect
Fan	Corrosion	T_cabinet, P_turbine
Fan	TMF	T_cabinet, P_turbine
IGBT	HCF	T_inverter_cabinet, T_nacelle, P_turbine

Table 1: Excerpt of the FMEA of the converter

Consider the FMEA of the converter in Table 1. We can map the columns to their corresponding representations from Definition 6. The entries in the column *Component* constitute the elements of $COMP$, the entries in *Fault Mode* of $MODES$ and $PROPS$ subsumes the entries of $Effect$.

$$COMP = \{ Fan, IGBT \}$$

$$MODES = \{ Corrosion, TMF, HCF \}$$

$$PROPS = \left\{ \begin{array}{l} T_cabinet, P_turbine, \\ T_inverter_cabinet, T_nacelle \end{array} \right\}$$

Through Definition 6 we obtain $FMEA_{Converter} =$

$$\left\{ \begin{array}{l} (Fan, Corrosion, \{T_cabinet, P_turbine\}), \\ (Fan, TMF, \{T_cabinet, P_turbine\}), \\ (IGBT, HCF, \{T_inverter_cabinet, T_nacelle, \\ P_turbine\}) \end{array} \right\}$$

Since the FMEA already represents the relation between defects and their manifestations the conversion to a suitable abductive KB is straightforward. It is worth noting that FMEAs usually consider single faults; thus, the resulting diagnostic system holds the single fault assumption. Let HC be the set of horn clauses. We define a mapping function $\mathfrak{M} : 2^{FMEA} \mapsto HC$ generating a corresponding propositional Horn clause for each entry of the FMEA [7].

Definition 7 (Mapping function \mathfrak{M}). *Given an FMEA, the function \mathfrak{M} is defined as follows:*

$$\mathfrak{M}(FMEA) =_{def} \bigcup_{t \in FMEA} \mathfrak{M}(t)$$

where $\mathfrak{M}(C, M, E) =_{def} \{mode(C, M) \rightarrow e \mid e \in E\}$.

We utilize the proposition $mode(C, M)$ to denote that component C experiences fault mode M . Thus, the set of component-fault mode couples forms the set of hypotheses.

$$Hyp =_{def} \bigcup_{(C, M, E) \in FMEA} \{mode(C, M)\}.$$

In regard to the running example the following elements compose the set Hyp :

$$Hyp = \left\{ \begin{array}{l} mode(Fan, Corrosion), \\ mode(Fan, TMF), \\ mode(IGBT, HCF) \end{array} \right\}$$

The set of propositional variables A is defined as the union of all effects stored in the FMEA as well as all hypotheses, that is the set of component-fault mode pairs, i.e.:

$$A =_{def} \bigcup_{(C, M, E) \in FMEA} E \cup \{mode(C, M)\}$$

Continuing our converter example:

$$A = \left\{ \begin{array}{l} T_cabinet, P_turbine, \\ T_inverter_cabinet, T_nacelle, \\ mode(Fan, Corrosion), \\ mode(Fan, TMF), \\ mode(IGBT, HCF) \end{array} \right\}$$

Applying \mathfrak{M} results in the following set of propositional Horn clauses representing Th and thus completing $KB_{Converter}$:

$$Th = \left\{ \begin{array}{l} mode(Fan, Corrosion) \rightarrow T_cabinet, \\ mode(Fan, Corrosion) \rightarrow P_turbine, \\ mode(Fan, TMF) \rightarrow T_cabinet, \\ mode(Fan, TMF) \rightarrow P_turbine, \\ mode(IGBT, HCF) \rightarrow T_inverter_cabinet, \\ mode(IGBT, HCF) \rightarrow T_nacelle, \\ mode(IGBT, HCF) \rightarrow P_turbine \end{array} \right\}$$

On account of the mapping function \mathfrak{M} and the underlying structure of the FMEAs, the compiled models feature a certain topology. First, the set of hypotheses and symptoms are disjoint sets. Second, since there is a causal link from faults to effects but not vice versa, the descriptions exhibit a forward and acyclic structure. Specifically, each implication connects one hypothesis to one effect, thus are biconjunctive clauses. In order to account for impossible observations, we append additional implications to KB stating that an effect and its negation cannot occur simultaneously, i.e. $e \wedge \neg e \models \perp$.

The question remains whether the generated models are suitable for the diagnostic task. Abductive explanations are consistent by definition and complete given an exhaustive search. Thus, the appropriateness of the system description is determined by whether a single fault diagnosis can be obtained given all necessary information is available.

Definition 8. (One Single Fault Diagnosis Property (OSFDP)) *Given a $KB(A, Hyp, Th)$. KB fulfills the OSFDP if the following hold:*

$$\forall m \in Hyp : \exists Obs \subseteq A : \{m\} \text{ is a diagnosis of } (A, Hyp, Th, Obs) \text{ and } \neg \exists m' \in Hyp : m' \neq m \text{ such that } \{m'\} \text{ is a diagnosis for the same PHCAP.}$$

The property ensures that under the assumption enough knowledge is available all single fault diagnoses can be distinguished and subsequently unnecessary replacement activities are avoided. To verify whether the OSFDP holds or not, we compute the set of propositions $\delta(h)$ implied by each hypothesis h and the theory. It is not fulfilled if we can record for two or more hypotheses the same $\delta(h)$. [7] describes a polynomial algorithm testing for the property. Note that the OSFDP check can be done on side of the FMEA before compiling the model. This is advantageous as the absence of the property indicates that internal variables or observations have not been considered in the FMEA.

Assume the set of hypotheses $\{h_1, h_2, \dots, h_n\}$ share the same $\delta(h)$. We cannot distinguish h_1, h_2, \dots, h_n from one another and thus all corresponding components have to be repaired or replaced in case they are part of the diagnosis. Therefore, we can treat them as a unit by replacing h_1, h_2, \dots, h_n with a new hypothesis h' . Once all indistinguishable hypotheses have been removed, the KB satisfies the OSFDP. Regarding the hypotheses, which cannot be differentiated, as one cause during diagnosis has an effect on the computational effort as fewer hypotheses are to be considered.

Algorithm `distinguishHypotheses` replaces all indistinguishable causes and ensures that after termination the given KB satisfies the OSFDP. Evidently, the algorithm's complexity is determined by the three nested loops, hence $O(|Hyp|^2|A - Hyp|)$. Since there is a finite number of hypotheses and effects possibly included in $\delta(h)$ the algorithm must terminate.

Algorithm 2 `distinguishHypotheses`

```

procedure DISTINGUISHHYPOTHESES ( $A, Hyp, Th$ )
   $\Psi[|Hyp|] \leftarrow Hyp$ 
  for all  $h_1 \in \Psi$  do
    for all  $h_2 \in \Psi$  do
      if  $h_1 \neq h_2$  then
        if  $\delta(h_1) = \delta(h_2)$  and  $\delta(h_1) \neq \emptyset$  then
          Create new hypothesis  $h' \triangleright h' \notin Hyp$ 
          Add  $h'$  to  $\Psi$ 
          Add  $h'$  to  $A$ 
          for all  $e \in \delta(h_1)$  do
            Add  $(h' \rightarrow e)$  to  $Th$ 
            Remove  $(h_1 \rightarrow e)$  from  $Th$ 
            Remove  $(h_2 \rightarrow e)$  from  $Th$ 
          end for
          Remove  $h_1 \wedge h_2$  from  $\Psi$ 
          Remove  $h_1 \wedge h_2$  from  $A$ 
        end if
      end if
    end for
  end for
  return  $KB(A, \Psi, Th)$ 
end procedure

```

Our running example of the converter does not fulfill the OSFDP, since $mode(Fan, Corrosion)$ and $mode(Fan, TMF)$ are not distinguishable. By removing both hypotheses and introducing $h' = mode((Fan, Corrosion), (Fan, TMF))$ the property is fulfilled.

Notice that abductive diagnosis is premised on the assumption that the model is complete; thus, we presume that all significant fault modes for each contributing part of the system have been contemplated in the FMEA. Furthermore, we expect on the one hand that the symptoms described within the FMEA are detectable in order to constitute observations. On the other hand, the automated mapping demands a consistent effect denotation throughout the analysis.

5 Abductive Diagnosis via SAT

Although an ATMS derives abductive diagnoses, it is limited to propositional Horn theories and subject to performance issues. Both problems have been accommodated through ATMS extensions and focus strategies. Nevertheless, the advances in the development

of SAT solvers and their application to a vast number of different AI problems and industrial domains have motivated us to consider a SAT-based approach for abductive diagnosis.

Recall Definition 3 of a diagnosis: Δ is an abductive explanation if $\Delta \cup Th \models Obs$ and $\Delta \cup Th \not\models \perp$. Through logical equivalence we recast the first condition to $\Delta \cup Th \cup \{-Obs\} \models \perp$, where $\{-Obs\}$ denotes the set containing the complement of each observation in Obs , i.e. $\forall o \in Obs : \neg o \in \{-Obs\}$ [10]. In general, we can state the relation as follows: given the theory and assuming the hypotheses to be true whereas stating the absence of a set of observations, results in an inconsistency due to the fact that the causes entail the effects, i.e. $Hyp \cup Th \cup \{-Obs\} \models \perp$. Thus, we draw on this relationship and reformulate the problem of generating minimal abductive explanations for a set of observations to computing minimal unsatisfiable subformulas.

Since MUSes contain several unsatisfiable subsets irrelevant for the diagnostic task, we define the set $MUSes_{Hyp}$, which only contains subset minimal MUS comprising clauses referring to hypotheses:

Definition 9. ($MUSes_{Hyp}$) *Let MUSes be the set of MUSes of $Hyp \cup Th \cup \{-Obs\}$, then $\forall M \in MUSes_{Hyp} : \exists U \in MUSes : M = U \cap Hyp$ and $\neg \exists M' \in MUSes_{Hyp} : M' \subset M$.*

Corollary 1. *Given a PHCAP(A, Hyp, Th, Obs), let $MUSes_{Hyp}$ be the set of interesting MUSes. A set $\Delta \subseteq Hyp$ is a minimal abductive diagnosis if $\exists M \in MUSes_{Hyp} : \Delta = M$ and $\Delta \cup Th \not\models \perp$.*

Proof. We can restate the problem of computing inconsistencies to finding the set of prime implicates of $Th \wedge Hyp \wedge \{-Obs\}$. By definition, the prime implicates are equivalent to the MUSes of said formula. \square

Deriving a minimal abductive explanation corresponds to computing a minimal subset of the hypotheses, which cannot be simultaneously satisfied with the theory and the negation of observations.

We devised the algorithm `satAB`, which computes the set of abductive diagnoses for a given PHCAP based on MUS enumeration. First, in order to take advantage of the MUSes, which correspond to the solutions of the PHCAP, we create an unsatisfiable CNF encoding of the problem. Since the Th consists of Horn clauses a conversion into CNF is straightforward. Note that we are, however, not limited to Horn clause models, as we can create a CNF representation based on Tseitin transformation [28]. We refer to the set of clauses associated with the theory as \mathcal{T} . For each $h \in Hyp$ we create a single clause assuming h to be true. Additionally, we generate a disjunction containing the negated observations. The resulting unsatisfiable formula is referred to as ϕ . $\Delta - Set$ is the set of diagnoses obtained from the PHCAP.

The diagnostic task consists in computing the sets of hypotheses which are responsible for the unsatisfiability of ϕ , i.e. $MUSes_{Hyp}(\phi)$. Since finding satisfiable subsets is an NP-hard problem whereas UNSAT resides in Co-NP, we employ an MCSes enumeration algorithm on the unsatisfiable formula and then derive the diagnoses via hitting set computation [25]. As we are only

$C_1 : \neg mode(Fan, Corrosion) \vee T_cabinet$	$C_2 : \neg mode(Fan, Corrosion) \vee P_turbine$
$C_3 : \neg mode(Fan, TMF) \vee T_cabinet$	$C_4 : \neg mode(Fan, TMF) \vee P_turbine$
$C_5 : \neg mode(IGBT, HCF) \vee T_inverter_cabinet$	$C_6 : \neg mode(IGBT, HCF) \vee T_nacelle$
$C_7 : \neg mode(IGBT, HCF) \vee P_turbine$	$C_8 : mode(Fan, Corrosion)$
$C_9 : mode(Fan, TMF)$	$C_{10} : mode(IGBT, HCF)$
$C_{11} : \neg P_turbine \vee \neg T_cabinet$	

Figure 1: SAT encoding of the running example

Algorithm 3 satAB

```

procedure SATAB ( $A, Hyp, Th, Obs$ )
     $MCSes \leftarrow \emptyset$ 
     $MCSes_{Hyp} \leftarrow \emptyset$ 
     $\mathcal{T} \leftarrow CNF(Th)$   $\triangleright$  CNF representation of  $Th$ 
     $\phi \leftarrow \mathcal{T} \cup Hyp \cup \bigvee_{o \in Obs} \neg o$ 
     $MCSes \leftarrow MCSes(\phi)$   $\triangleright$  MCS enumeration algorithm
    for all  $m \in MCSes$  do
        if  $m \subseteq Hyp$  and  $m \cup Th$  is consistent then
             $MCSes_{Hyp} \leftarrow m \cup MCSes_{Hyp}$ 
        end if
    end for
     $\Delta - Set \leftarrow MHS(MCSes_{Hyp})$   $\triangleright$  Minimal hitting set
    algorithm
    return  $\Delta - Set$ 
end procedure

```

interested in the conflicts stemming from the assumptions that all hypotheses are true, we select each MCS only containing clauses referring to explanations. For this reason, we create the set $MCSes_{Hyp}$ such that $\forall m \in MCSes_{Hyp} : m \subseteq Hyp$. This has one practical rationale: it diminishes the number of sets to be considered by the hitting set algorithm. The corresponding MUSes derived via hitting set computation of $MCSes_{Hyp}$ already constitute the abductive diagnoses.

Consider again our running example of the converter. We already obtained the KB via the mapping function \mathfrak{M} . Let us assume that the condition monitoring system of the wind turbine encountered that the turbine's power output is lower than expected ($P_turbine$) and that the cabinet temperature exceeds a certain threshold ($T_cabinet$), i.e. $Obs = \{P_turbine, T_cabinet\}$. In Figure 1 we depict the CNF representation ϕ of the abduction problem. Clauses C_1 to C_7 refer to \mathcal{T} , C_8 to C_{10} to the set Hyp and clause C_{11} contains the negation of the set of observations.

Computing the $MCSes$ of ϕ we obtain: $MCSes =$

$$\left\{ \begin{array}{l} \{C_{11}\}, \{C_1, C_3\}, \{C_1, C_9\}, \{C_3, C_8\}, \{C_9, C_8\}, \\ \{C_4, C_7, C_2\}, \{C_4, C_{10}, C_2\}, \{C_4, C_7, C_8\}, \\ \{C_4, C_{10}, C_8\}, \{C_2, C_9, C_7\}, \{C_2, C_9, C_{10}\} \end{array} \right\}.$$

Extracting the MCSes, which only contain clauses from Hyp and are consistent with regard to the theory, results in

$$MCSes_{Hyp} = \{\{C_9, C_8\}\}.$$

By computing the hitting set of $MCSes_{Hyp}$, we obtain the set of MUSes solely referring to explanations, which is in fact the set of diagnoses:

$$\Delta - Set = \{\{C_9\}, \{C_8\}\}.$$

Hence the abductive diagnoses are $\Delta_1 = \{mode(Fan, Corrosion)\}$ and $\Delta_2 = \{mode(Fan, TMF)\}$.

6 Empirical Evaluation

To determine whether computing abductive diagnoses via SAT yields any computational advantages in the case of our models, we conducted an empirical evaluation, comparing `abductiveExplanations` to `satAB` on several instances of FMEAs. In case of the former we employed a Java implementation of an unfocused ATMS. The algorithm `satAB` exploits on the one hand an MCS enumeration procedure and on the other hand an implementation of a hitting set algorithm. We utilized the MCS_{LS} tool by [19] to compute the MCSes. MCS_{LS} is written in C++, employs Minsat 2.2 as the SAT solver, and provides the possibility to apply several MCS enumeration algorithms. We decided for the CLD approach of MCS_{LS} , which takes advantage of disjoint unsatisfiable cores and showed the best overall performance in a preliminary experimental set-up. Regarding the hitting set computation, we engaged a Java implementation of the Binary Hitting Set Tree algorithm [29] which performed well in a comparison of minimal hitting set algorithms [30]. All the numbers presented in this section were obtained from a Lenovo ThinkPad T540p Intel Core i7-4700MQ processor (2.60 GHz) with 8 GB RAM running Ubuntu 14.04 (64-bit).

Several publicly available as well as project internal FMEAs provide the basis for our evaluation. They cover various technical systems and subsystems with different underlying structures. In particular they describe faults in electrical circuits, a connector system by Ford (FCS), the Focal Plane Unit (FPU) of the Heterodyne Instrument for the Far Infrared (HIFI) built for the Herschel Space Observatory, printed circuit boards (PCB), the Anticoincidence Detector (ACD) mounted on the Large Area Telescope of the Fermi Gamma-ray Space Telescope, the Maritim ITStandard (MiTS), and rectifier, inverter, transformer, backup components, as well as main bearing of an industrial wind turbine. By applying the mapping function \mathfrak{M} , we generated the corresponding abductive knowledge bases KB for each FMEA. Table 2 provides an overview of the FMEAs' structure and the evaluation results. It is worth noting that the FMEAs vary in the number of hypotheses, i.e. component-fault mode couples, the number of effects, and the number of rules, i.e. the links between faults and symptoms. Due to Th of an abductive KB comprising Horn clauses, a conversion into a CNF representation, suitable for the MCS_{LS} tool, is straightforward. We do not address the model compilation times, since the system description would be compiled offline and

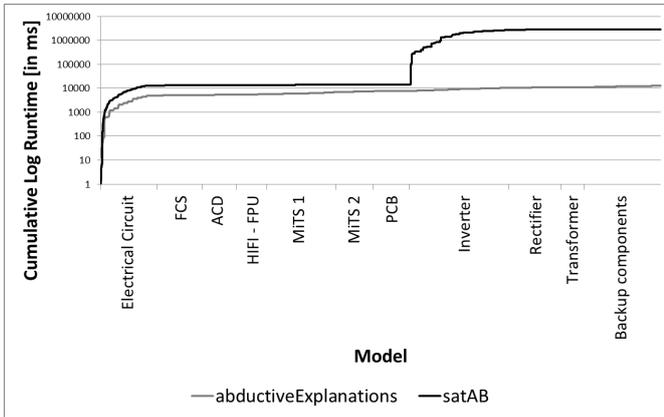


Figure 2: Cumulative runtimes of `abductiveExplanations` and `satAB` for the FMEA instances

the mapping execution consumed less than one second for the examples we utilized so far.

Table 2 shows that none, except of the model resulting from the transformer’s FMEA, of the original models satisfy the OSFDP. Therefore, we compiled a second set of models fulfilling the property by exchanging each set of indistinguishable hypotheses with a new single hypothesis representing said set. For example, Algorithm `distinguishHyp` ensures that the resulting KB satisfies the OSFDP. In Table 2 the original models are identified accordingly, and the adapted models are provided with the label *OSFDP*. Note that the number of hypotheses and rules diminishes for the adapted models.

In the experiments, we computed the abductive explanations for $|Obs|$ from one to the maximum number of effects possible. The observations were generated randomly; however, the same set was used for `satAB` and `abductiveExplanations` on the original as well as adapted model. The results reported in Table 2 have been obtained from ten trials and both algorithms faced a 200 seconds runtime limit. Whereas some of the small runtimes are arguable due to the measurement in the milliseconds range, Table 2 reveals that `satAB` (Mean = 703.73 ms, SD = 8432.07 ms, Median = 0.59 ms, Skewness = 18.61) does not outperform `abductiveExplanations` (Mean = 3.08 ms, SD = 16.38 ms, Median = 1 ms, Skewness = 12.68) in general. From the statistical data we can infer that the underlying distribution of both algorithms is highly right skewed, thus the bulk of values is located towards the lower runtimes. We can even observe that for certain instances, the SAT-based approach performs rather poorly. Amongst these are the model of an inverter and a rectifier of an industrial wind turbine. `satAB` exceeded the given timeout four times for the former. Notice that in all these cases the MCSes generation already reached the time threshold. According to [19] CLD requires $|\phi| - p + 1$ SAT solver calls, where p refers to the size of the smallest MCS of ϕ . In our case $p = 1$, as the clause representing the set of negated observations always constitutes an MCS. Thus, $|\phi|$ SAT solver calls are necessary, where $|\phi|$ is determined by $|Th| + |Hyp| + 1$, with 1 referring to the clause containing the observations. Unsurprisingly, the larger FMEAs are more computationally demanding. It is worth mentioning that in the majority of cases

the hitting set computation accounted for a negligible fraction of the total runtime.

Figure 2 illustrates the cumulative log runtimes for `satAB` and `abductiveExplanations` on the FMEA models generated. Although `abductiveExplanations` performs on average better, the first model requires a longer computation time for both algorithms. Moreover, the illustration reveals the high computational effort necessary for `satAB` to compute the diagnoses for the model of the inverter. As expected we observe particularly high runtimes when the set of observations contains effects corresponding to different hypotheses. This has a greater impact on `satAB` than on the ATMS implementation. For the section from the models FCS to PCB in Figure 2, however, we can see that the cumulative runtime for `abductiveExplanations` rises at a steeper angle. Generally, the data gathered in the experiment do not suggest a performance benefit of the SAT-based approach over an ATMS implementation.

7 Conclusion and Future Work

In the course of the paper, we presented a mapping from failure assessments available to propositional Horn clause models. The modeling methodology relies on FMEAs as they comprise information on faults and their symptoms. Hence, they provide a suitable source for model compilation. Although in our case an ATMS can be used to compute abductive diagnoses, it is limited to propositional Horn theories. We proposed a SAT-based approach to abductive model-based diagnosis which allows us to reason on more expressive representations. Our method is based on computing conflict sets, i.e. MUSes, resulting from a rewritten, unsatisfiable system description. Subsets of these unsatisfiable cores constitute the minimal abductive explanations. Since the computation of MUSes is computationally demanding our proposed algorithm exploits its hitting set dual, MCSes, in order to derive minimal diagnoses.

We empirically compared an implementation of a diagnosis engine employing an ATMS to our SAT-based algorithm. The results indicate that while for some of the models, the algorithm performs well, in general we could not observe a performance advantage. Particular examples led to even longer computation times than the ATMS-based implementation. Despite the fact that the data provided no evidence of a computational benefit in employing a SAT-based approach, we believe that the possibility to utilize more expressive models provides an interesting incentive for future research in this area.

Since the evaluation results, did not indicate a superiority of the SAT-based approach on grounds of MCSes enumeration, we currently investigate direct conflict generation methods. Additionally, due to the model structure and the experiment data we are planning on employing compilation methods [31, 32], in order to divert some of the computational inefficiency to the model generation process.

Acknowledgments

The work presented in this paper has been supported by the FFG project Applied Model Based Reasoning (AMOR) under grant 842407. We would further like to express our gratitude to our industrial partner, Uptime Engineering GmbH.

Component	Model Structure			#Diagnoses					Runtime [in ms]			
	#Hyp	#Effects	#Rules	MAX	AVG	SF	DF	TF	Algorithm	MIN	MAX	AVG
Electrical circuit												
Original	32	17	52	792	197.15	11	11	66	abductive Explanations	< 1	425	27.87
									satAB	< 1	181.33	76.05
OSFDP	15	17	35	1	1	1	1	1	abductive Explanations	< 1	8	0.33
									satAB	< 1	1.91	0.16
FCS												
Original	17	17	51	18	2.93	3	6	18	abductive Explanations	< 1	1	0.42
									satAB	< 1	6.41	1.28
OSFDP	15	17	49	18	2.75	3	6	18	abductive Explanations	< 1	61	2.04
									satAB	< 1	4.73	0.56
ACD												
Original	13	16	41	15	2.89	5	15	15	abductive Explanations	< 1	84	1.38
									satAB	< 1	2.89	0.35
OSFDP	12	16	39	10	2.04	5	10	10	abductive Explanations	< 1	1	0.29
									satAB	< 1	2.435	0.28
Main bearing												
Original	3	5	20	3	2.54	3	0	0	abductive Explanations	< 1	1	0.16
									satAB	< 1	1	0.09
OSFDP	2	5	15	2	1.54	2	0	0	abductive Explanations	< 1	1	0.12
									satAB	< 1	0.61	0.03
HIFI - FPU												
Original	17	11	36	63	8.64	3	7	21	abductive Explanations	< 1	86	2.54
									satAB	< 1	8.33	3
OSFDP	9	11	27	6	1.55	2	2	3	abductive Explanations	< 1	1	0.15
									satAB	< 1	1	0.09
MiTS 1												
Original	18	21	48	24	8.40	3	2	6	abductive Explanations	< 1	94	3.40
									satAB	< 1	3.02	0.39
OSFDP	13	21	43	1	1	1	1	1	abductive Explanations	< 1	100	1.54
									satAB	< 1	2.15	0.16
MiTS 2												
Original	22	15	48	288	39.98	4	8	18	abductive Explanations	< 1	109	4.49
									satAB	< 1	15.16	3.43
OSFDP	14	15	37	5	2.02	1	5	2	abductive Explanations	< 1	1	0.33
									satAB	< 1	1.68	0.20
PCB												
Original	10	11	24	2	1.49	2	2	2	abductive Explanations	< 1	1	0.21
									satAB	< 1	1.49	0.1
OSFDP	9	11	23	1	1	1	1	1	abductive Explanations	< 1	1	0.11
									satAB	< 1	1	0.1
Inverter												
Original	30	38	144	450	23.73	19	5	50	abductive Explanations	< 1	107	6.15
									satAB	< 1	166593	5007.37
OSFDP	23	38	124	66	5.89	14	3	6	abductive Explanations	< 1	94	1.67
									satAB	< 1	1110.82	38.23
Rectifier												
Original	20	17	93	88	10.83	8	24	32	abductive Explanations	< 1	6	1.07
									satAB	< 1	24236.9	1070.88
OSFDP	14	17	66	22	3.06	5	18	8	abductive Explanations	< 1	1	0.63
									satAB	< 1	44.74	4.88
Transformer												
Original	5	8	22	2	1.06	2	2	1	abductive Explanations	< 1	1	0.16
									satAB	< 1	1.69	0.06
OSFDP	5	8	22	2	1.06	2	2	1	abductive Explanations	< 1	1	0.13
									satAB	< 1	1.91	0.08
Backup components												
Original	25	30	114	252	23.06	8	12	21	abductive Explanations	< 1	138	5.24
									satAB	< 1	41.98	12.89
OSFDP	19	30	95	48	3.29	7	7	10	abductive Explanations	< 1	4	0.79
									satAB	< 1	10.06	3.09

Table 2: Features of the FMEAs and experimental results. For each component we conducted the experiment using an implementation of `abductiveExplanations` and `satAB`. The columns *SF*, *DF*, *TF* display the maximum number of single faults, double faults, and triple faults, respectively.

References

- [1] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
- [2] Johan de Kleer and Brian C Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32(1):97–130, 1987.
- [3] Brian C Williams and P Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 971–978, 1996.
- [4] Peter Struss, Andreas Malik, and Martin Sachenbacher. Case studies in model-based diagnosis and fault analysis of car-subsystems. In *Proc. 1st Int'l Workshop Model-Based Systems and Qualitative Reasoning*, pages 17–25, 1996.
- [5] Luca Console, Daniele Theseider Dupre, and Pietro Torasso. On the Relationship Between Abduction and Deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.
- [6] Peter Zoetewij, Jurryt Pietersma, Rui Abreu, Alexander Feldman, and Arjan JC Van Gemund. Automated fault diagnosis in embedded systems. In *Secure System Integration and Reliability Improvement, 2008. SSIRI'08. Second International Conference on*, pages 103–110. IEEE, 2008.
- [7] Franz Wotawa. Failure mode and effect analysis for abductive diagnosis. In *Proceedings of the International Workshop on Defeasible and Ampliative Reasoning (DARe-14)*, volume 1212. CEUR Workshop Proceedings, ISSN 1613-0073, 2014. <http://ceur-ws.org/Vol-1212/>.
- [8] Peter G. Hawkins and Davis J. Woollons. Failure modes and effects analysis of complex engineering systems using functional models. *Artificial Intelligence in Engineering*, 12:375–397, 1998.
- [9] Chris Price and Neil Taylor. Automated multiple failure fmea. *Reliability Engineering & System Safety*, 76:1–10, 2002.
- [10] Sheila A McIlraith. Logic-based abductive inference. *Knowledge Systems Laboratory, Technical Report KSL-98-19*, 1998.
- [11] Pierre Marquis. Consequence finding algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 41–145. Springer, 2000.
- [12] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2):301–353, 1992.
- [13] Franz Wotawa, Ignasi Rodriguez-Roda, and Joaquim Comas. Environmental decision support systems based on models and model-based reasoning. *Environmental Engineering and Management Journal*, 9(2):189–195, 2010.
- [14] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel SAT-based approach to model based diagnosis. *Journal of Artificial Intelligence Research*, pages 377–411, 2014.
- [15] Alexander Feldman, Gregory Provan, Johan de Kleer, Stephan Robert, and Arjan van Gemund. Solving model-based diagnosis problems with Max-SAT solvers and vice versa. In *DX-10, International Workshop on the Principles of Diagnosis*, 2010.
- [16] Alexander Feldman, Gregory M Provan, and Arjan JC van Gemund. Computing minimal diagnoses by greedy stochastic search. In *AAAI*, pages 911–918, 2008.
- [17] Iulia Nica and Franz Wotawa. ConDiag-computing minimal diagnoses using a constraint solver. In *International Workshop on Principles of Diagnosis*, pages 185–191, 2012.
- [18] Alexander Felfernig and Monika Schubert. Fastdiag: A diagnosis algorithm for inconsistent constraint sets. In *Proceedings of the 21st International Workshop on the Principles of Diagnosis (DX 2010), Portland, OR, USA*, pages 31–38, 2010.
- [19] Joao Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 615–622. AAAI Press, 2013.
- [20] Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1046–1052. AAAI Press, 2013.
- [21] Gustav Nordh and Bruno Zanuttini. What makes propositional abduction tractable. *Artificial Intelligence*, 172:1245–1284, 2008.
- [22] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Hypothesis classification, abductive diagnosis and therapy. In *Expert Systems in Engineering Principles and Applications*, pages 69–78. Springer, 1990.
- [23] Franz Wotawa, Ignasi Rodriguez-Roda, and Joaquim Comas. Abductive Reasoning in Environmental Decision Support Systems. In *AIAI Workshops*, pages 270–279, 2009.
- [24] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving*. Academic press, 1973.
- [25] Mark H Liffiton and Kareem A Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning*, 40(1):1–33, 2008.
- [26] Elazar Birnbaum and Eliezer L Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *Journal of Experimental & Theoretical Artificial Intelligence*, 15(1):25–46, 2003.
- [27] Christopher S Gray, Roxane Koitz, Siegfried Psutka, and Franz Wotawa. An abductive diagnosis and modeling concept for wind power plants. In *International Workshop on Principles of Diagnosis*, 2014.
- [28] Gregory Tseitin. On the complexity of proofs in propositional logics. In *Seminars in Mathematics*, volume 8, pages 466–483, 1970.
- [29] Li Lin and Yunfei Jiang. The computation of hitting sets: review and new algorithms. *Information Processing Letters*, 86(4):177–184, 2003.
- [30] Ingo Pill, Thomas Quaritsch, and Franz Wotawa. From conflicts to diagnoses: An empirical evaluation of minimal hitting set algorithms. In *22nd Int. Workshop on the Principles of Diagnosis*, pages 203–210, 2011.
- [31] Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM (JACM)*, 48(4):608–647, 2001.
- [32] Pietro Torasso and Gianluca Torta. Computing minimum-cardinality diagnoses using OBDDs. In *KI 2003: Advances in Artificial Intelligence*, pages 224–238. Springer, 2003.

Fault Tolerant Control for a 4-Wheel Skid Steering Mobile Robot

George K. Fourlas¹, George C. Karras² and Kostas J. Kyriakopoulos²

¹ Department of Computer Engineering, Technological Educational Institute (T. E. I.) of Central Greece, Lamia, Greece

email: gfourlas@teiste.gr

² Control Systems Laboratory, School of Mechanical Eng. National Technical University of Athens (NTUA) Athens, Greece

email: karrasg@mail.ntua.gr, kkyria@mail.ntua.gr

Abstract

This paper studies a fault tolerant control strategy for a four wheel skid steering mobile robot (SSMR). Through this work the fault diagnosis procedure is accomplished using structural analysis technique while fault accommodation is based on a Recursive Least Squares (RLS) approximation. The goal is to detect faults as early as possible and recalculate command inputs in order to achieve fault tolerance, which means that despite the faults occurrences the system is able to recover its original task with the same or degraded performance. Fault tolerance can be considered that it is constituted by two basic tasks, fault diagnosis and control redesign. In our research using the diagnosis approach presented in our previous work we addressed mainly to the second task proposing a framework for fault tolerant control, which allows retaining acceptable performance under systems faults. In order to prove the efficacy of the proposed method, an experimental procedure was carried out using a Pioneer 3-AT mobile robot.

1 Introduction

The higher demands to achieve more reliable performance in modern robotic systems have necessitated the development of appropriate fault diagnosis methods. The appearance of faults is inevitable in all systems, such as wheeled robots, either because their elements are worn out or because the environment in which they operate, presents unanticipated situations [4].

In a large number of applications, as for example search and rescue, planetary exploration, nuclear waste cleanup or mine decommissioning, the wheeled robots operate in environments where human intervention can be very costly, slow or even impossible. They can move freely in such dynamic environments. It is therefore essential for the robots to monitor their behavior so that faults may be addressed before they result in catastrophic failures.

A wheeled mobile robot is usually an embedded control platform, which consists of an on-board computer, power, motor control system, communications, sonars, cameras, laser radar system and sensors such as gyroscope, encoders, accelerometers etc, Fig. 1.



Figure 1. 4-Wheel Skid Steering Mobile Robot.

Fault diagnosis and accommodation for wheeled mobile robots is a complex problem due to the large number of faults that can be present such as faults of sensors and actuators [10] - [20].

Model based fault detection and isolation is a method to perform fault diagnosis using a certain model of the system. The goal is to detect faults as early as possible in order to provide a timely warning [8]. The aim of timely handling the fault occurrence is to accommodate their consequences so that the system remains functional. This can be achieved with fault tolerance.

In cases where fault could not be tolerated, it is necessary to use redundant hardware. In practice there exist two different approaches for fault tolerance control, static redundancy and dynamic redundancy [8].

In [10] and [16], the research is focused only on the problem of fault detection and identification in a mobile robot and different approaches related to state estimation were introduced. In [9] and [15], the research interest is focused only on the problem of fault detection which is a separate problem in the fault diagnosis domain. The research efforts in [7] and [12] - [14] are primarily intended to detect faults in the sensors of a wheeled robot. Concerning the research area of detection and accommodation on wheeled robots there is also a small number of efforts [18] with different approaches and methodologies.

As a fault, it can be considered any unpermitted deviation from the normal behavior of a system. Fault diagnosis is the procedure of determination of the component which is faulty. Consequently, the aim of fault diagnosis is to produce the suitable fault statement regarding the malfunction of a wheeled robot.

Fault diagnosis includes fault detection, which is the indication that something is going wrong in the system and fault isolation, which is the determination of the magnitude of the fault, by evaluating symptoms. Follows fault detection. Fault detection and isolation tasks together are referred to as fault diagnosis (FDI - Fault Detection and Isolation).

Among the various methods in the design of a residual generator, only few deal with nonlinear systems. Structural analysis is a technique that provides feasible solutions to the residual generation of nonlinear systems

Structural analysis methods are used in research publications [2] and [6]. Paper [3] presents a structural analysis for complex systems such as a ship propulsion benchmark. In [13] and [14] the authors discuss how structural analysis technique is applied to an unmanned ground vehicle for residual generation.

In this research, a model based fault diagnosis for a four wheel skid steering mobile robot (SSMR) is presented. The basic idea is to use structural analysis based technique in order to generate residuals. For this purpose we use the kinematic model of the mobile robot that serves to the design of the structural model of the system. This technique provides the parity equations which can be used as residual generators. The advantage of the proposed method is that offers feasible solution to the residual generation of nonlinear systems. Additionally, we propose a fault accommodation technique based on RLS approximation in order to provide recalculated control inputs in the case that the left or right set of the robot tires becomes flat.

The mobile robot is supposed to be equipped with two high resolution optical quadrature shaft encoders mounted on reversible-DC motors which provide rotational speeds of the left and right wheels ω_L and ω_R respectively and an inertial measurement unit (IMU) which provides the forward linear acceleration and the angular velocity well as the angle θ between the mobile robot axle and the x axis of the mobile robot. The absolute pose (horizontal position and orientation) of the robot is available via a camera system mounted on the workspace of the robot. A distinctive marker is placed at the top side of the robot.

The paper is organized as follows. We start by presenting the mathematical model of a Pioneer 3-AT mobile robot in section 2. Section 3 describes the fault diagnosis procedure. Section 4 describes the methodology of fault accommodation. In section 5 we present the application results of the proposed method to the robotic platform. Conclusions and directions for future work are presented in Section 6.

2 Mathematical Model of Pioneer 3-AT Mobile Robot

In this work, the mobile robot Pioneer 3-AT was used as a robotic platform. This robot is a four wheel skid – steering vehicle actuated by two motors, one for the left sided wheels and the other for the right sided wheels. The wheels on the same side are mechanically coupled and thus have the same velocity. Also, they are equipped with encoders and the angular readings are available through routine calls.

The kinematic model describes the motion constrains of the system, as well as the relationship of the sensors measurements with the system states and it is crucial for the fault diagnosis procedure.

2.1 Kinematic Model

The geometry of the robot is presented in Fig.2. To consider the model of the four wheel skid steering mobile robot (SSMR) it is assumed that the robot is placed on a plane surface where (X_I, Y_I) is the inertial reference frame and (X, Y) is a local coordinate frame fixed on the robot at its center of mass (COM). The position of the COM is (x, y) with respect to the inertial frame and θ is the orientation of the local coordinate frame with respect to the inertial frame.

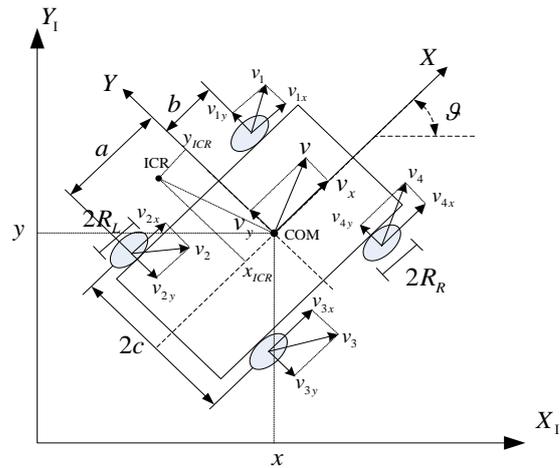


Figure 2. Mobile Robot Geometry.

As depicted in Fig. 2, a is the distance between the center of mass and the front wheels axle along X , b is the distance between the center of mass and the rear wheels axle along X , c is half distance between wheels along Y and R_L, R_R are the radii of left and right wheels respectively. The coordinates of the instantaneous center of rotation (ICR) are (x_{ICR}, y_{ICR}) .

Assuming that the robot moves on a horizontal plane the linear velocity with respect to the local frame is given by

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} \quad (1)$$

and its angular velocity is given by

$$\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \quad (2)$$

The state vector with respect to the inertial frame is

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad (3)$$

The time derivatives of (3) denotes the robot's velocity vector and is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = \begin{bmatrix} \cos \vartheta & -\sin \vartheta & 0 \\ \sin \vartheta & \cos \vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (4)$$

Assuming that longitudinal slip between the wheels and the surface can be neglected we have the following equation,

$$v_{ix} = R_i \omega_i \quad (5)$$

where v_{ix} is the longitudinal component of the total velocity vector v_i of the i -th wheel expressed with respect to the local frame and R_i is the rolling radius of that wheel.

If we take into account all wheels (Fig. 2), the following relationships between the wheels can be obtained [11],

$$\begin{aligned} v_L &= v_{1x} = v_{2x} \\ v_R &= v_{3x} = v_{4x} \\ v_F &= v_{1y} = v_{4y} \\ v_B &= v_{2y} = v_{3y} \end{aligned} \quad (6)$$

where v_L refers to the longitudinal coordinates of the left wheels velocities, v_R refers to the longitudinal coordinates of the right wheels velocities, v_F refers to the lateral coordinates of the front wheels velocities and v_B refers to the lateral coordinates of the rear wheels velocities.

Unlike other mobile robots, lateral velocities of the four wheel skid steering mobile robot are generally nonzero since from its mechanical structure the lateral skidding is necessary if the robot changes its orientation. Therefore, in order to complete the kinematic model, the following non-holonomic constrain in Pfaffian form is introduced

$$\begin{bmatrix} -\sin \vartheta & \cos \vartheta & -x_{ICR} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0 \quad (7)$$

Then we have

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\boldsymbol{\eta} \quad (8)$$

where

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix} \cos \vartheta & x_{ICR} \sin \vartheta \\ \sin \vartheta & -x_{ICR} \cos \vartheta \\ 0 & 1 \end{bmatrix} \quad (9)$$

$$\boldsymbol{\eta} = \begin{bmatrix} v_x \\ \omega \end{bmatrix} \quad (10)$$

$\mathbf{S}(\mathbf{q})$ is a full rank matrix, whose columns are in the null space of $\mathbf{A}(\mathbf{q})$,

$$\mathbf{S}^T(\mathbf{q})\mathbf{A}^T(\mathbf{q}) = 0 \quad (11)$$

It is noted that since $\dim(\boldsymbol{\eta}) = 2 < \dim(\mathbf{q}) = 3$, equation (8) describes the kinematic of a sub-actuated robot with the nonholonomic constraint given by (7).

We suppose that the mobile robot localization is calculated via the following measurement devices:

- two high resolution optical quadrature shaft encoder mounted on reversible-DC motors which provide rotational speeds of the left and right wheels ω_L and ω_R respectively,
- an Inertial Measurement Unit (IMU) which provides the forward linear acceleration and the angular velocity as well as the angle ϑ between the mobile robot axle and the x axis of the mobile robot.
- A camera system, which calculates the pose of the robot, by tracking a marker placed at the top side of it.

In this work we are only interested in abrupt faults which occur in the actuators of the mobile robot and as consequence, we make the following assumptions.

- *Assumption 1:* When the mobile robot starts functioning all its components are in normal mode.
- *Assumption 2:* The magnitude of the noise is assumed to be significantly smaller than the magnitude of the faults.
- *Assumption 3:* Regarding the wheel radius the following inequalities are satisfied:

$$R_R + \delta R_R > 0 \quad \& \quad R_L + \delta R_L > 0$$

According to this assumption, faults that result in the complete loss of the wheel are not considered.

3 Fault Detection and Isolation

Between several techniques for generating residuals, limited number of them concerns nonlinear systems. Such one is structural analysis. Using this method we can extract information about system components that we are not able to measure. Also we can take the parity equations that allow generating residuals.

The structure of the mobile robot is described using the following sets of constrains C and variables V

$$C = \{c_1, c_2, \dots, c_9\} \quad (12)$$

$$V = X \cup K \quad (13)$$

X is a subset of the unknown ones and K is a subset of known that are measurements and inputs.

The above subsets are

$$X = \{\dot{x}, \dot{y}, \dot{\vartheta}, v_x, v_y\} \quad (14)$$

$$K = \{x, y, \vartheta, \ddot{x}, \ddot{y}, \omega, \omega_L, \omega_R\} \quad (15)$$

The constrain set of the mobile robot is

$$c_1 : \dot{x} = \cos \vartheta v_x - \sin \vartheta v_y \quad (16)$$

$$c_2 : \dot{y} = \sin \vartheta v_x + \cos \vartheta v_y \quad (17)$$

$$c_3 : \dot{\vartheta} = \omega \quad (18)$$

$$c_4 : \ddot{x} = \frac{d\dot{x}}{dt} \quad (19)$$

$$c_5 : \ddot{y} = \frac{d\dot{y}}{dt} \quad (20)$$

$$c_6 : \vartheta = \int_0^t \omega d\tau \quad (21)$$

$$c_7 : v_x = \frac{r}{2}(\omega_R + \omega_L) \quad (22)$$

$$c_8 : \dot{x} = \frac{dx}{dt} \quad (23)$$

$$c_9 : \dot{y} = \frac{dy}{dt} \quad (24)$$

Through the above technique we create the following incidence matrix that describes the robot structure, Table 1.

Table 1. Incidence Matrix

	KNOWN							UNKNOWN					
	x	y	θ	\ddot{x}	\ddot{y}	ω	ω_L	ω_R	\dot{x}	\dot{y}	$\dot{\theta}$	v_x	v_y
c_1			1						1			1	①
c_2			1							①		1	1
c_3						1					①		
c_4				1					①				
c_5					1					1			
c_6			1								1		
c_7							1	1				①	
c_8	1								1				
c_9		1								1			

Applying matching algorithm [1] to the incidence matrix, we take out the following matched M and unmatched U constrains

$$M = \{c_1, c_2, c_3, c_4, c_7\} \quad (25)$$

$$U = \{c_5, c_6, c_8, c_9\} \quad (26)$$

In order to have residual generators we use the following parity equations

$$c_5 (\dot{y}, \ddot{y}) = 0 \quad (27)$$

$$c_6 (\vartheta, \dot{\vartheta}) = 0 \quad (28)$$

$$c_8 (x, \dot{x}) = 0 \quad (29)$$

$$c_9 (y, \dot{y}) = 0 \quad (30)$$

By starting from the unknown variables through backtracking to known variables, the residuals are:

$$r_1 = \ddot{y} - \frac{d}{dt} \left(\sin \vartheta \frac{r}{2} (\omega_R + \omega_L) + \frac{\left(\cos \vartheta \frac{r}{2} (\omega_R + \omega_L) - \int \ddot{x} d\tau \right)}{\sin \vartheta} \right) \quad (31)$$

$$r_2 = \vartheta - \int_0^t \omega d\tau \quad (32)$$

$$r_3 = \int \ddot{x} d\tau - \frac{dx}{dt} \quad (33)$$

$$r_4 = \int \ddot{y} d\tau - \frac{dy}{dt} \quad (34)$$

4 Fault Accommodation

Fault accommodation is the phase that follows the fault diagnosis. One of the most important issues to consider for the design of fault tolerant control is relative to the performance and functionality of the system under consideration. More specific it should take into consideration, the degree of performance degradation that is acceptable. There are two aspects of system performance, dynamic and steady state. In our approach we take into account the second one. We also use the aforementioned fault diagnosis method to monitor the system. The goal is to have the necessary information about the fault occurrence for timely counteraction. Figure 3 shows the overall structure of the proposed fault tolerant mechanism. It consists of two parts: i) the fault detection module which accepts as inputs the measurement of the linear and angular velocity of the SSMR and decides about the type of fault according to the method described in Section 3, and ii) the fault accommodation module which accepts as inputs the type of fault as well as the measurement of the linear and angular velocity and recalculates accordingly the command inputs in order to compensate for the fault.

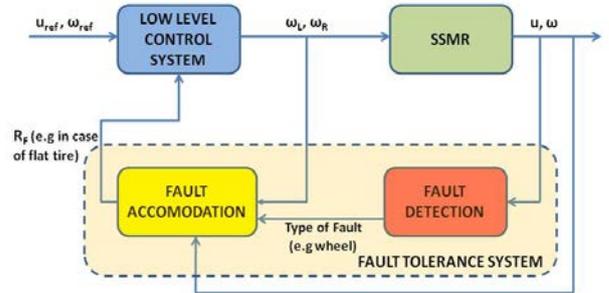


Figure 3. Fault Tolerance System Architecture.

When a fault occurs the appropriate action is undertaken (e.g. maintenance, repair, reconfiguration, stop operation) in such a way to prevent system failures. In that level the performance degradation that is acceptable is relative to the minimum requirements that ensure the system functionality. There is always the case that the malfunction may cause hazard for the process or the environment, and a decision for stopping the operation is unavoidable.

In this work, we propose a fault accommodation technique which is employed when either the left or the right set of tires becomes flat during the operation of a SSMR. It is obvious that when a flat tire fault occurs, the total nominal radius R_{NOM} (rim and tire) of the fault wheel changes to R_F , where $R_F < R_{NOM}$. The proposed fault accommoda-

tion strategy relies on the online estimation of the new radius R_F , in order to correct the commanded rotational speeds of the faulty wheel and compensate for the fault which otherwise will inevitably lead the vehicle to diverge from its nominal course.

As explained in [11], the kinematic model of the SSMR can be considered equivalent with the unicycle differential drive one, mainly due to the existence of a single motor drive and a transmission belt for each set of wheels (left and right), which impose the same rotational speed for each set of wheels. According to this assumption we can safely assume that:

$$\begin{bmatrix} u_x \\ \omega \end{bmatrix} = \frac{1}{2c} \begin{bmatrix} c & c \\ -1 & 1 \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \quad (35)$$

where $v_L = \omega_L R_L$, $v_R = \omega_R R_R$ are the equivalent linear velocities of the left and right wheels respectively in relation to the rotational speeds and radii. If we consider that the fault will occur only at the one set of the wheels (left or right), we may consider only the angular velocity equation for the accommodation. Thus, only the angular velocity measurement is needed. The fault accommodation is based on the online estimation of the new radius R_F employing a Recursive Least Squares algorithm. More specific, we may consider the following linear equation for the measurement of the mobile's robot body angular velocity, in case a left side fault occurs:

$$\begin{aligned} \hat{\omega}_k - \frac{0.5}{c} \omega_R R_R &= H_k \hat{R}_{F_k} + v_k \\ H_k &= H_{L_k} = -\frac{0.5}{c} \omega_L \\ v_k &\sim (0, R_k) \end{aligned} \quad (36)$$

while in the case of a right side fault:

$$\begin{aligned} \hat{\omega}_k + \frac{0.5}{c} \omega_L R_L &= H_k \hat{R}_{F_k} + v_k \\ H_k &= H_{R_k} = \frac{0.5}{c} \omega_R \\ v_k &\sim (0, R_k) \end{aligned} \quad (37)$$

Having defined the measurement model of the robot angular velocity in the body frame, we proceed to the online estimation of the fault wheel radius employing the following Recursive Least Squares approximation algorithm:

1. Initialize the estimator:

$$\begin{aligned} \hat{R}_{F_0} &= E(R_F) = R_{L/R} \\ P_0 &= E\left(\left(R_F - \hat{R}_{F_0}\right)^2\right) \end{aligned} \quad (38)$$

where $R_{L/R}$ is the nominal radius of the left or right wheel set.

2. Obtain a new measurement ω_k , assuming that it is given by the equation (36), or (37).

3. Update the estimate \hat{R}_{F_k} and the covariance P_k of the estimation error sequentially according to:

$$\begin{aligned} K_k &= P_{k-1} H_k^T \left(H_k P_{k-1} H_k^T + R_k \right)^{-1} \\ \hat{R}_{F_k} &= \hat{R}_{F_{k-1}} + K_k \left(\omega_k - H_k \hat{R}_{F_{k-1}} \right) \\ P_k &= \left(I - K_k H_k \right) P_{k-1} \end{aligned} \quad (39)$$

where ω_k is the actual measurement of the body angular velocity as delivered by the IMU sensor.

4. Using the estimated wheel radius \hat{R}_{F_k} we correct the commanded wheel angular velocity as follows:

$$\omega_{L_cor} = \frac{-2\omega_k c + \omega_R R_R}{\hat{R}_{F_k}} \quad (40)$$

$$\omega_{R_cor} = \frac{2\omega_k c + \omega_L R_L}{\hat{R}_{F_k}} \quad (41)$$

in case there is a left or a right wheel fault respectively.

5 Application Results

The proposed method has been implemented and tested experimentally on Pioneer 3-AT mobile robot. All experiments have been performed indoors. We consider a faulty situation where the right wheel set is flat (forward and backward wheels). We apply a command of $\omega_L = \omega_R = 5 \text{ rad/s}$ for both set of wheels. In the nominal situation (no faults) the robot should move (almost) straight forwards without any deviation. The robot starts from the origin of the inertial frame and moves for 2.5m. The time interval dt between successive IMU measurements is 2.5msec. The nominal radius of the wheels (proper inflation) is $R_L = R_R = 0.115 \text{ m}$.

In the first experiment (Fig. 4), the fault accommodation algorithm is not enabled and as we can observe from the trajectory of the vehicle, the SSMR significantly diverges from its nominal course to the right.

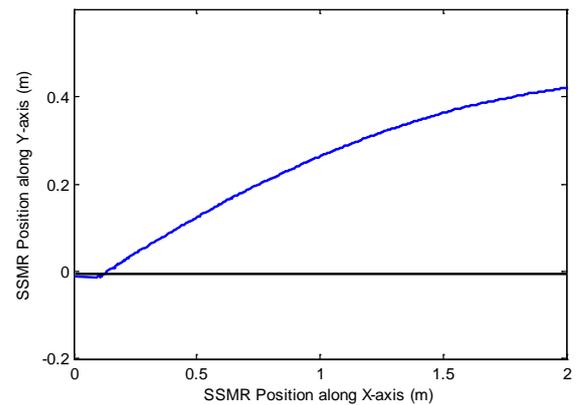


Figure 4. Robot's position while the right wheel set is flat.

The fault detection algorithm is enabled, and as we can see from Fig. 5 the fault was successfully detected by the proposed structural analysis algorithm.

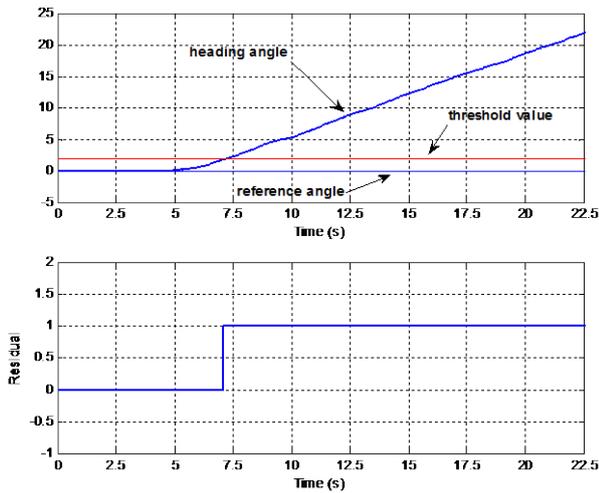


Figure 5. Fault signal as the right wheel set is flat.

In the second experiment we impose the same control inputs to the SSMR $\omega_L = \omega_R = 5 \text{ rad/s}$, but this time not only the fault detection but also the proposed fault accommodation algorithm is enabled. As we can see in Fig. 6 the on line estimation algorithm quickly converge to the new radius of the faulty wheel set and consequently the fault accommodation algorithm provides modified inputs to the right wheel set (Fig. 7).

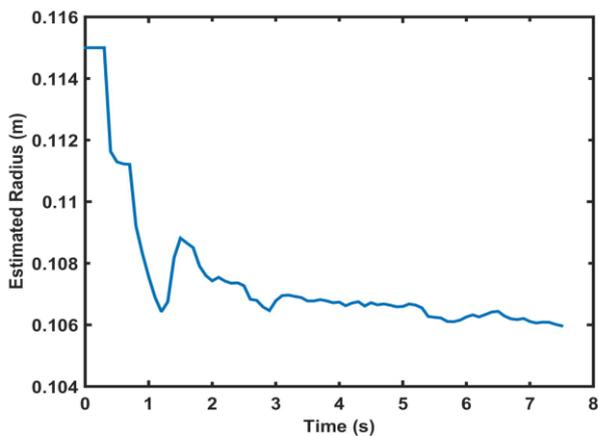


Figure 6. On line estimation of faulty radius.

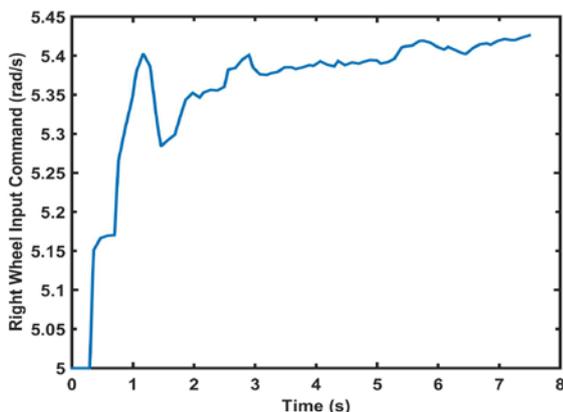


Figure 7. Recalculated input from the fault accommodation algorithm.

As we can observe from Fig. 8 the trajectory of the SSMR was successfully detained in an almost straight line form.

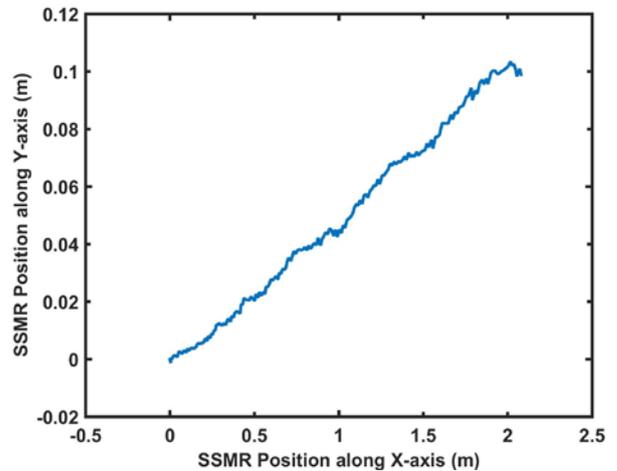


Figure 8. SSMR Corrected Planar Trajectory.

6 Conclusion

The notion of fault tolerant control for a 4-wheel skid steering mobile robot is an important problem to deal with, since faults appearance is inevitable in such systems. The most significant challenge arises from the complexity of the system. In this paper we have introduced the underlying concepts for our approach to fault tolerant control for mobile robots focusing our attention mainly to control re-configuration. As concerning the issue of fault diagnosis the structural analysis based technique is used in order to generate residuals. We use the kinematic model of the mobile robot that serves to the development of the structural model of the system. The above technique provides the parity equations which can be used as residual generators since model based fault diagnosis approach is based on residuals. The advantage of the above method is that it can offer a feasible solution to the residual generation of non-linear systems. The fault accommodation procedure targets in the case where one of the two wheel tire sets becomes flat. The proposed accommodation method is based on a RLS approximation of the new faulty wheel radius and via this information a new control input is calculated in order to compensate for the fault.

The efficacy of the proposed method is demonstrated through an extensive experimental procedure using a mobile robot Pioneer 3-AT.

Acknowledgments

This research is implemented through the Operational Program "Education and Lifelong Learning" and is co-financed by the European Union (European Social Fund) and Greek national funds. The work is part of the research project entitled «DIAGNOR - Fault Diagnosis and Accommodation for Wheeled Mobile Robot» of the Act "Archimedes III - Strengthening Research Groups in TEI Lamia".

References

- [1] M. Blanke, M. Kinnaert, J. Luzne, M. Staroswiecki, «Diagnosis and Fault Tolerant Control», ser. Heidelberg, Springer-Verlag, 2003.
- [2] M. Blanke, H. Niemann, and T. Lorentzen, “Structural analysis – a case study of the Rømer satellite,” in Proc. of IFAC Safeprocess 2003, Washington, DC, USA, 2003.
- [3] M. Blanke, V. Cocquempot, R. I. Zamanabadi, and M. Staroswiecki, “Residual generation for the ship benchmark using structural approach,” in Proc. of Int. Conference on CONTROL’98, Swansea, UK, Sep 1998.
- [4] Steven X. Ding, “Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools, Springer-Verlag Berlin, 2008.
- [5] G.K. Fourlas, K.J. Kyriakopoulos, N.J. Krikelis, “Fault Diagnosis of Hybrid Systems”, Proceedings of the 2005 IEEE International Symposium on Intelligent Control, 13th IEEE Mediterranean Conference on Control and Automation, Limassol, Cyprus, 2005.
- [6] G.K. Fourlas, “Theoretical Approach of Model Based Fault Diagnosis for a 4 - Wheel Skid Steering Mobile Robot, 21st IEEE Mediterranean Conference on Control and Automation (MED '13), Plataniass-Charania, Crete, GREECE, June 25-28, 2013.
- [7] G. Ippoliti, S. Longhi, A. Monteriù, “Model-based sensor fault detection system for a smart wheelchair”, IFAC 2005.
- [8] R. Isermann, “Fault Diagnosis Systems – An Introduction from Fault Detection to Fault Tolerant”, Springer-Verlag Berlin Heidelberg, 2006.
- [9] B. Halder, and N. Sarkar, “Robust Fault Detection of Robotic Systems: New Results and Experiments”, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida - May 2006.
- [10] Z. Kira, “Modeling Cross-Sensory and Sensorimotor Correlations to Detect and Localize Faults in Mobile Robots”, Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego, CA, USA, Oct 29 - Nov 2, 2007.
- [11] K. Kozłowski and D. Pazderski, “Modeling and Control of a 4-Wheel Skid-Steering Mobile Robot”, Int. J. Appl. Math. Comput. Sci., 2004, vol. 14, no. 4, 477-496.
- [12] Y. Morales, E. Takeuchi and T. Tsubouchi, “Vehicle Localization in Outdoor Woodland Environments with sensor fault detection”, 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, May 19-23, 2008.
- [13] A. Monteriù, P. Asthan, K. Valavanis and S. Longhi, “Model-Based Sensor Fault Detection and Isolation System for Unmanned Ground Vehicles: Theoretical Aspects (part I)”, 2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.
- [14] A. Monteriù, P. Asthan, K. Valavanis and S. Longhi, “Model-Based Sensor Fault Detection and Isolation System for Unmanned Ground Vehicles: Experimental Validation (part II), 2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.
- [15] P. Sundvall and P. Jensfelt, “Fault detection for mobile robots using redundant positioning systems” Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida - May 2006.
- [16] C. Valdivieso, and A. Cipriano, “Fault Detection and Isolation System Design for Omnidirectional Soccer-Playing Robots”, Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design Munich, Germany, October 4-6, 2006.
- [17] R. Izadi-Zamanabadi, “Structural Analysis Approach to Fault Diagnosis with Application to Fixed-wing Aircraft Motion” in Proc. of American control Conference, USA, 2002.
- [18] D. Zhuo-hua, CAI Zi-xing, YU Jin-xia “Fault Diagnosis and Fault Tolerant Control for Wheeled Mobile Robots under Unknown Environments: A Survey”, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005.
- [19] S.Zaman, G. Steinbauer, J.Maurer, P.Lepej, and S.Uran, “An integrated model-based diagnosis and repair architecture for ROS-based robot systems”, IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013.
- [20] D. Portugal, and Rui P. Rocha, “Scalable, Fault-Tolerant and Distributed Multi-Robot Patrol in Real World Environments” 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), November 3-7, Tokyo, Japan, 2013.

Data-Driven Monitoring of Cyber-Physical Systems Leveraging on Big Data and the Internet-of-Things for Diagnosis and Control

Oliver Niggemann^{1,3}, Gautam Biswas², John S. Kinnebrew², Hamed Khorasgani²,
Sören Volgmann¹ and Andreas Bunte³

¹ Fraunhofer Application Center Industrial Automation, Lemgo, Germany
e-mail: {oliver.niggemann, soeren.volgmann}@iosb-ina.fraunhofer.de

² Vanderbilt University and Institute for Software Integrated Systems, Nashville, TN, USA
e-mail: {john.s.kinnebrew, hamed.g.khorasgani, gautam.biswas}@vanderbilt.edu

³ Institute Industrial IT, Lemgo, Germany
e-mail: {andreas.bunte}@hs-owl.de

Abstract

The majority of projects dealing with monitoring and diagnosis of Cyber Physical Systems (CPSs) relies on models created by human experts. But these models are rarely available, are hard to verify and to maintain and are often incomplete. Data-driven approaches are a promising alternative: They leverage on the large amount of data which is collected nowadays in CPSs, this data is then used to learn the necessary models automatically. For this, several challenges have to be tackled, such as real-time data acquisition and storage solutions, data analysis and machine learning algorithms, task specific human-machine-interfaces (HMI) and feedback/control mechanisms. In this paper, we propose a cognitive reference architecture which addresses these challenges. This reference architecture should both ease the reuse of algorithms and support scientific discussions by providing a comparison schema. Use cases from different industries are outlined and support the correctness of the architecture.

1 Motivation

The increasing complexity and the distributed nature of technical systems (e.g. power generation plants, manufacturing processes, aircraft and automobiles) have provided traction for important research agendas, such as Cyber Physical Systems (CPSs) [1; 2], the US initiative on the “Industrial Internet” [3] and its German counterpart “Industrie 4.0” [4]. In these agendas, a major focus is on self-monitoring, self-diagnosis and adaptivity to maintain both operability and safety, while also taking into account humans-in-the-loop for system operation and decision making. Typical goals of such self-diagnosis approaches are the detection and isolation of faults and anomalies, identifying and analyzing the effects of degradation and wear, providing fault-adaptive control, and optimizing energy consumption [5; 6].

So far, the majority of projects and papers for analysis and diagnosis has relied on manually-created diagnosis models of the system’s physics and operations [6; 7; 8]: If a drive is used, this drive is modeled, if a reactor is installed, the associated chemical and physical processes are

modeled. However, the last 20 years have clearly shown that such models are rarely available for complex CPSs; when they do exist, they are often incomplete and sometimes inaccurate, and it is hard to maintain the effectiveness of these models during a system’s life-cycle.

A promising alternative is the use of data-driven approaches, where monitoring and diagnosis knowledge can be learned by observing and analyzing system behavior. Such approaches have only recently become possible: CPSs now collect and communicate large amounts of data (see Big Data [9]) via standardized interfaces, giving rise to what is now called the Internet of Things [10]. This large amount of data can be exploited for the purpose of detecting and analyzing anomalous situations and faults in these large systems: The vision is developing CPSs that can observe their own behavior, recognize unusual situations during operations, inform experts, who can then update operations procedures, and also inform operators, who use this information to modify operations or plan for repair and maintenance.

In this paper, we take on the challenges of proposing a common data-driven framework to support monitoring, anomaly detection, prognosis (degradation modeling), diagnosis, and control. We discuss the challenges for developing such a framework, and then discuss case studies that demonstrate some initial steps toward data-driven CPSs.

2 Challenges

In order to implement data-driven solutions for the monitoring, diagnosis, and control of CPSs, a variety of challenges must be overcome to enable the learning pathways illustrated in Figure 1:

Data Acquisition: All data collected from distributed CPSs, e.g. sensors, actuators, software logs, and business data, must meet real-time requirements, as well as including time synchronization and spatial labeling when relevant. Often sensors and actuators operate at different rates, so data alignment, especially for high-velocity data, becomes an issue. Furthermore, data must be annotated semantically to allow for a later data analysis.

Data Storage, Curation, and Preprocessing: Data will be stored and preprocessed in a distributed way. Environmental factors and the actual system configuration (e.g., for the current product in a production system) must also be stored. Depending on the applications, a relational database format, or increasingly distributed noSQL technologies [11], may

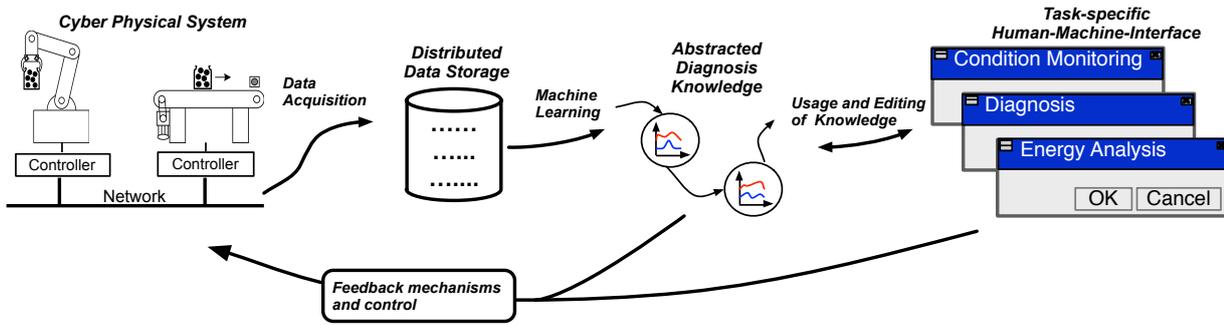


Figure 1: Challenges for the analysis of CPSs.

need to be adopted, so that the right subsets of data may be retrieved for different analyses. Real-world data can also be noisy, partially corrupted, and have missing values. All of these need to be accommodated in the curation, storage, and pre-processing applications.

Data Analysis and Machine Learning: Data must be analyzed to derive patterns and abstract the data into condensed usable knowledge. For example, machine learning algorithms can generate models of normal system behavior in order to detect anomalous patterns in the data [12]. Other algorithms can be employed to identify root causes of observed problems or anomalies. The choice and design of appropriate analyses and algorithms must consider factors like the ability to handle large volumes and sometimes high velocities of heterogeneous data. At a minimum, this generally requires machine learning, data mining, and other analysis algorithms that can be executed in parallel, e.g., using the Spark [13], Hadoop [14], and MapReduce [15] architectures. In some cases, this may be essential to meet real-time analysis requirements.

Task-specific Human-Machine-Interfaces: Tasks such as condition monitoring, energy management, predictive maintenance or diagnosis require specific user interfaces [16]. One set of interfaces may be more tailored for offline analysis to allow experts to interact with the system. For example, experts may employ information from data mining and analytics to derive new knowledge that is beneficial to the future operations of the system. Another set of interfaces would be appropriate for system operators and maintenance personnel. For example, appropriate operator interfaces would be tailored to provide analysis results in interpretable and actionable forms, so that the operators can use them to drive decisions when managing a current mission or task, as well as to determine future maintenance and repair.

Feedback Mechanisms and Control: As a reaction to recognized patterns in the data or to identified problems, the user may initiate actions such as a reconfiguration of the plant or an interruption of the production for the purpose of maintenance. In some cases, the system may react without user interactions; in this case, the user is only informed.

3 Solutions

As Section 4 will show, the challenges from Section 2 reappear in the majority of CPS examples. While details, such as the machine learning algorithms employed or the nature of data and data storage formats can vary, the primary steps are about the same. Most CPS solutions re-implement all of these steps and even employ different solution strategies—

raising the overall efforts, preventing any reuse of hardware/software and impeding a comparison between solutions.

To achieve better standardization, efficiency, and repeatability, we suggest a generic cognitive reference architecture for the analysis of CPSs. Please note that this architecture is a pure reference architecture which does not constraint later implementations and introduction of application-specific methods.

Figure 2 shows its main components:

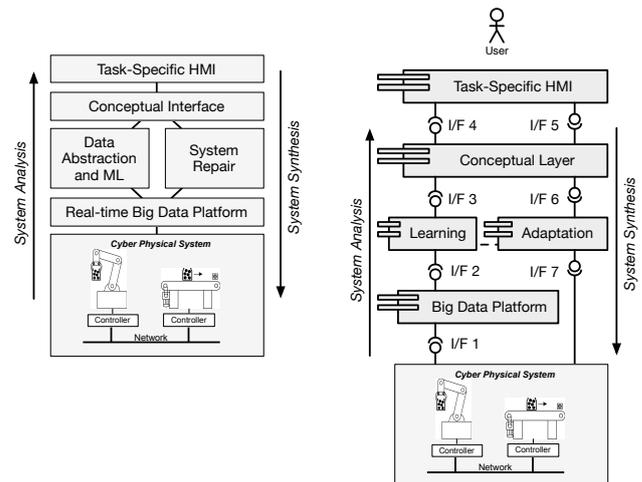


Figure 2: A cognitive architecture as a solution for the analysis of CPSs.

Big Data Platform (I/F 1 & 2): This layer receives all relevant system data, e.g., configuration information as well as raw data from sensors and actuators. This is done by means of domain-dependent, often proprietary interfaces, here called interface 1 (I/F 1). This layer then integrates, often in real-time, all of the data, time-synchronizes them and annotates them with meta-data that will support later analysis and interpretation. For example, sensor meta-data may consist of the sensor type, its position in the system and its precision. This data is provided via I/F 2, which, therefore, must comprise the data itself and also the meta-data (i.e., the semantics). A possible implementation approach for I/F 2 may be the mapping into and use of existing Big Data platforms, such as Sparks or Hadoop, for storing the data and the Data Distribution Service (DDS) for acquiring the data (and meta-data).

Learning Algorithms (I/F 2 & 3): This layer receives all

data via I/F 2. Since I/F 2 also comprises meta-data, the machine learning and diagnosis algorithms need not be implemented specifically for a domain but may adapt themselves to the data provided. In this layer, unusual patterns in the data (used for anomaly detection), degradation effects (used for condition monitoring) and system predictions (used for predictive maintenance) are computed and provided via I/F 3. Given the rapid changes in data analysis needs and capabilities, this layer may be a toolbox of algorithms where new algorithms can be added by means of plug-and-play mechanisms. I/F 3 might again be implemented using DDS.

Conceptual Layer (I/F 3 & 4): The information provided by I/F 3 must be interpreted according to the current task at hand, e.g. computing the health state of the system. Therefore, the provided information about unusual patterns, degradation effects and predictions are combined with domain knowledge to identify faults, their causes and rate them according to the urgency of repair. A semantic notation will be added to the information, e.g. the time for next maintenance or a repair instruction, which will be provided at I/F 4 in a human understandable manner. From a computer science perspective, this layer provides reasoning capabilities on a symbolic or conceptual level and adds a semantic context to the results.

Task-Specific HMI (I/F 4 & 5): The user is in the center of the architecture presented here, and, therefore, requires task-, context- and role-specific Human-Machine-Interfaces (HMIs). This HMI uses I/F 4 to get all needed analysis results and presents them to the user. Adaptive interfaces, rather than always showing the results of the same set of analyses, could allow a wider range of information to be provided, while maintaining efficiency and preventing information overload. Beyond obvious dynamic capabilities like alerts for detected problems or anomalies, the interfaces could further adapt the information displayed to be more relevant to the current user context (e.g. the user's location within a production plant, recognition of tasks the user may be engaged in, observed patterns of the user's previous information-seeking behavior, and knowledge of the user's technical background). If the user decides to influence the system (e.g. shutdown of a subsystem or adaptation of the system behavior), I/F 5 is used to communicate this decision to the conceptual layer. Again, I/F 4 and I/F 5 might be implemented using DDS.

Conceptual Layer (I/F 5 & 6): The user decisions will be received via I/F 5. The conceptual layer will use the knowledge to identify actions which are needed to carry out the users' decisions. For example, a decision to decrease the machine's cycle time by 10 % could lead to actions such as decreasing the robot speed by 10 % and the conveyor speed by 5 % or the decision to shutdown a subsystem. These actions are communicated via I/F 6 to the adaption layer.

Adaption (I/F 6 & 7): This layer receives system adaption commands on the conceptual level via I/F 6—which again might be based on DDS. Examples are the decrease of robot speed by 10 % or a shutdown of a subsystem. The adaption layer takes these commands on the conceptual level and computes, in real-time, the corresponding changes to the control system. For example, a subsystem shutdown might require a specific network signal or a machine's timing is changed by adapting parameters of the control algorithms, again by means of network signals. I/F 7 therefore uses domain-dependent interfaces.

4 Case Studies

We present a set of case studies that cover the manufacturing and process industries, as well as complex CPS systems, such as aircraft.

4.1 Manufacturing Industry

The modeling and learning of discrete timing behavior for manufacturing industry (e.g., automotive industry) is a new field of research. Due to the intuitive interpretation, Timed Automata are well-suited to model the timing behavior of these systems. Several algorithms have been introduced to learn such Timed Automata, e.g. RTI+ [17] and BUTLA [18]. Please note that the expert still has to provide structural information about the system (e.g. asynchronous sub-systems) and that only the temporal behavior is learned.

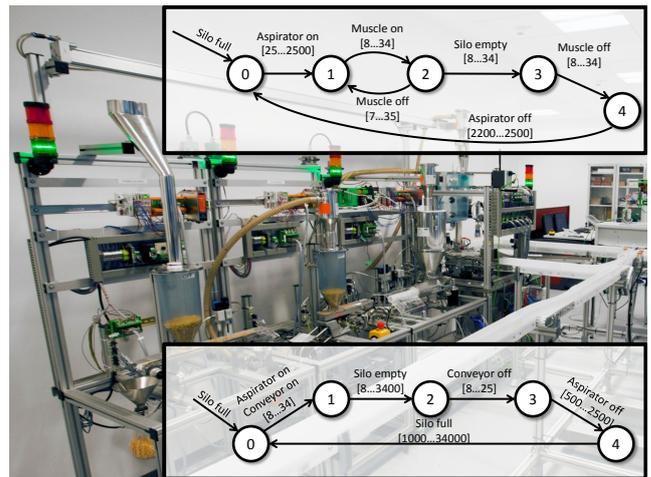


Figure 3: Learned Timed Automata for a manufacturing plant.

The data acquisition for this solution (I/F 1 in Figure 2) has been implemented using a direct capturing of Profinet signals including an IEEE 1588 time-synchronization. The data is offered via OPC UA (I/F 2). On the learning layer, timed automata are learned from historical data and compared to the observed behavior. Also, the sequential behavior of the observed events as well as the timing behavior is checked, anomalies are signaled via I/F 3. On the conceptual layer it is decided whether an anomaly is relevant. Finally, a graphical user interface is connected to the conceptual layer via OPC UA (I/F 4).

Figure 3 shows learned automata for a manufacturing plant: The models correspond to modules of the plants, transitions are triggered by a control signals and are annotated with a learned timing interval.

4.2 Energy Analysis In Process Industry

Analyzing the energy consumption in production plants has some special challenges: Unlike the discrete systems described in Section 4.1, also continuous signals such as the energy consumption must be learned and analyzed. But also the discrete signals must be taken into consideration because continuous signals can only be interpreted with respect to the current system's status, e.g. it is crucial to know whether a valve is open or whether a robot is turned on. And the system's status is usually defined by the history of discrete control signals.

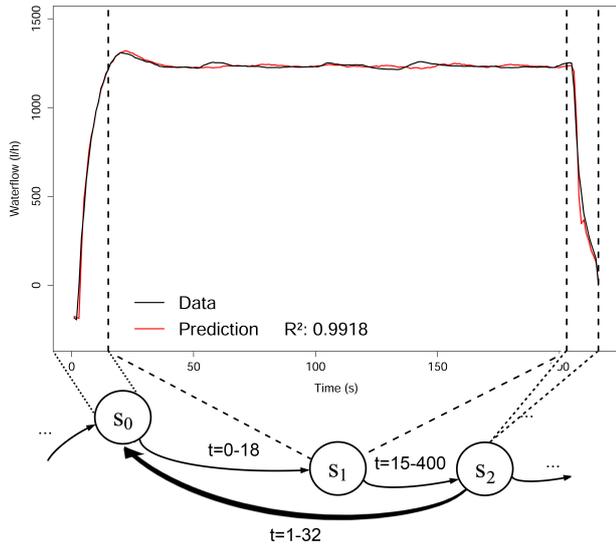


Figure 4: A learned hybrid automaton modeling a pump.

In [19], an energy anomaly detection system is described which analyzes three production plants. Ethercat and Profinet is used for I/F 1 and OPC UA for I/F 2. The collected data is then condensed on the learning layer into hybrid timed automata. Also on this layer, the current energy consumption is compared to the energy prediction. Anomalies in the continuous variables are signaled to the user via mobile platforms using web services (I/F 3 and 4).

In Figure 4, a pump is modeled by means of such automata using the flow rate and switching signals. The three states S_0 to S_2 are separating the continuous function into three linear pieces which can then be learned automatically.

Figure 5 shows a typical learned energy consumption (here for bulk good production).

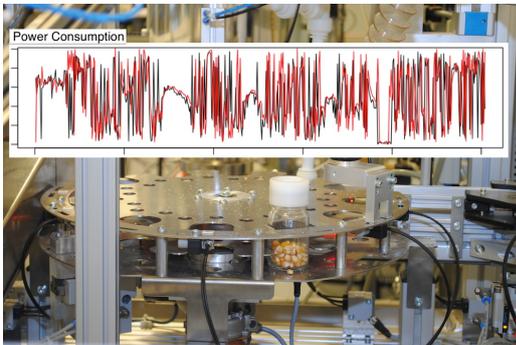


Figure 5: A measured (black line) and a learned power consumption (red line).

4.3 Big Data Analysis in Manufacturing Systems

Analyzing historical process data during the whole production cycle requires new architectures and platforms for handling the enormous volume, variety and velocity of the data. Data analysis pushes the classical data acquisition and storage up to its limits, i.e. big data platforms are need.

In the assembling line of the SmartFactoryOWL, a small factory used for production and research, a big data platform is established to acquire, store and visualize the data from

the production cycles. In Figure 6 the architecture of the big data platform is depicted.

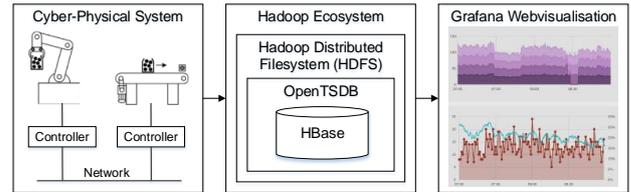


Figure 6: Data Analysis Platform in Manufacturing

The CPS is connected through OPC UA (I/F 1 in Figure 2) with an Hadoop ecosystem. Hadoop itself is an software framework for scalable distributed computing. The process data is stored in a non-relational database (HBase) which is based on a distributed file-system (HDFS). On top of HBase, a time-series database *OpenTSDB* is used as an interface to explore and analyze the data (I/F 2 in Figure 2). Through this database it is possible to do simple statistics such as mean-values, sums or differences, which is usually not possible within the non relational data stores.

Using the interfaces of OpenTSDB or Hadoop, it becomes possible to analyze the data directly on the storage system. Hence, the volume of a historical dataset need not be loaded into a single computer system, instead the algorithms can work distributively on the data. A web interface can be used to visualize the data as well as the computed results. In Figure 6, grafana is used for data visualization. In the SmartFactoryOWL this big data platform is currently being connected to the application scenarios from Sections 4.1 and 4.2.

4.4 Anomaly Detection in Aircraft Flight Data

Fault detection and isolation schemes are designed to detect the onset of adverse events during operations of complex systems, such as aircraft and industrial processes. In other work, we have discussed approaches using machine learning classifier techniques to improve the diagnostic accuracy of the online reasoner on board of the aircraft [20]. In this paper, we discuss an anomaly detection method to find previously undetected faults in aircraft system [21].

The flight data used for improving detection of existing faults and discovering new faults was provided by Honeywell Aerospace and recorded from a former regional airline that operated a fleet of 4-engine aircraft, primarily in the Midwest region of the United States. Each plane in the fleet flew approximately 5 flights a day and data from about 37 aircraft was collected over a five year period. This produced over 60,000 flights. Since the airline was a regional carrier, most flight durations were between 30 and 90 minutes. For each flight, 182 features were recorded at sample rates that varied from 1Hz to 16Hz. Overall this produced about 0.7 TB of data.

Situations may occur during flight operations, where the aircraft operates in previously unknown modes that could be attributed to the equipment, the human operators, or environmental conditions (e.g., the weather). In such situations, data-driven anomaly detection methods [12], i.e., finding patterns in the operations data of the system that were not expected before can be applied. Sometimes, anomalies may represent truly aberrant, undesirable and faulty behavior; however, in other situations they may represent behav-

iors that are just unexpected. We have developed unsupervised learning or clustering methods for off-line detection of anomalous situations. Once detected and analyzed, relevant information is presented to human experts and mission controllers to interpret and classify the anomalies.

Figure 7 illustrates our approach. We started with curated raw flight data (layer "Big Data Platform" in Figure 2), transforming the time series data associated with the different flight parameters to a compressed vector form using wavelet transforms. The next step included building a dissimilarity matrix of pairwise flight segments using the Euclidean distance measure, followed by a subsequent step where the pairwise between flight distances was used to run a 'complete link' hierarchical clustering algorithm [22] (layer "Learning" in Figure 2). Run on the flight data, the algorithm produced a number of large clusters that we considered to represent nominal flights, and a number of smaller clusters and outlier flights that we initially labeled as anomalous. By studying the feature value differences between the larger nominal and smaller anomalous clusters with the help of domain experts, we were able to interpret and explain the anomalous nature ("Conceptual Layer" in Figure 2).

These anomalies or faults represented situations that the experts had not considered before; therefore, this unsupervised or semi-supervised data driven approach provided a mechanism for learning new knowledge about unanticipated system behaviors. For example, when analyzing the aircraft data, we found a number of anomalous clusters. One of them turned out to be situations where one of the four engines of the aircraft was inoperative. On further study of additional features, the experts concluded that these were test flights conducted to test aspects of the aircraft, and, therefore, they represented known situations, and, therefore, not an interesting anomaly. A second group of flights were interpreted to be take offs, where the engine power was set much higher than most flights in the same take off condition. Further analysis of environmental features related to these set of take-off's revealed that these were take-offs from a high altitude airport at 7900 feet above sea level.

A third cluster provided a more interesting situation. The experts when checking on the features that had significantly different values from the nominal flights realized that the auto throttle disengaged in the middle of the aircraft's climb trajectory. The automatic throttle is designed to maintain either constant speed during takeoff or constant thrust for other modes of flight. This was an unusual situation where the auto thruster switched from maintaining speed for a takeoff to a setting that applied constant thrust, implying that the aircraft was on the verge of a stall. This situation was verified by the flight path acceleration sensor shown in Figure 7. By further analysis, the experts determined that in such situations the automatic throttle would switch to a possibly lower thrust setting to level the aircraft and compensate for the loss in velocity. By examining the engine parameters, the expert verified that all the engines responded in an appropriate fashion to this throttle command. Whereas this analysis did not lead to a definitive conclusion other than the fact the auto throttle, and therefore, the aircraft equipment, responded correctly, the expert determined that further analysis was required to answer the question "*why did the aircraft accelerate in such a fashion and come so close to a stall condition?*". One initial hypothesis to explain these situations was pilot error.

4.5 Reliability and Fault Tolerant Control

Most complex CPSs are safety-critical systems that operate with humans-in-the-loop. In addition to equipment degradation and faults, humans can also introduce erroneous decisions, which becomes a new source of failure in the system. Figure 8 represents possible faults and cyber-attacks that can occur in a CPS.

There are several model-based fault tolerant control strategies for dynamic systems in the literature (see for example [23] and [24]). Research has also been conducted to address network security and robust network control problems (see for example [25] and [26]). However, these methods need mathematical models of the system, which may not exist for large scale complex systems. Therefore, data driven control [27] and data driven fault tolerant control [28] have become an important research topic in recent years. For CPSs, there are more aspects of the problem that need to be considered. As it is shown in Figure 8, there are many sources of failure in these systems.

We propose a hybrid approach that uses an abstract model of the complex system and utilizes the data to ensure the compatibility between model and the complex system. Data abstraction and machine learning techniques are employed to extract patterns between different control configurations and system outputs unit by computing the correlation between control signals and the physical subsystems outputs. The highly correlated subsystems (layer "Learning" in Figure 2) become candidates for further study of the effects of failure and degradation at the boundary of these interacting subsystems. For complex systems, all possible interactions and their consequences are hard to pre-determine, and data-driven approaches help fill this gap in knowledge to support more informed decision-making and control. A case-based reasoning module can be designed to provide input on past successes and failed opportunities, which can then be translated by human experts into operational monitoring, fault diagnosis, and control situations ("Conceptual Layer" in Figure 2). Some of the control paradigms that govern appropriate control configurations, such as modifying sequence of mission tasks and switching between different objectives or changing the controller parameters (layer Adaptation in Figure 2) are being studied in a number of labs including ours [29].

Example Fault Tolerant Control of Fuel Transfer System The fuel system supplies fuel to the aircraft engines. Each individual mission will have its own set of requirements. However, common requirements such as saving the aircraft Center of Gravity (CG), safety, and system reliability are always critical. A set of sensors included in the system to measure different system variables such as the fuel quantity contained in each tank, engines fuel flow rates, boost pump pressures, position of the valves and etc.

There are several failure modes such as the total loss or degradation in the electrical pumps or a leakage in the tanks or connecting pipes in the system. Using the data and the abstract model we can detect and isolate the fault and estimate its parameters. Then based on the type fault and its severity the system reconfiguration unit chooses the proper control scenario from the control library. For example in normal situation the transfer pumps and valves are controlled to maintain a transfer sequence to keep the aircraft center of gravity within limits. This control includes maintaining a balance between the left and right sides of the aircraft. When there

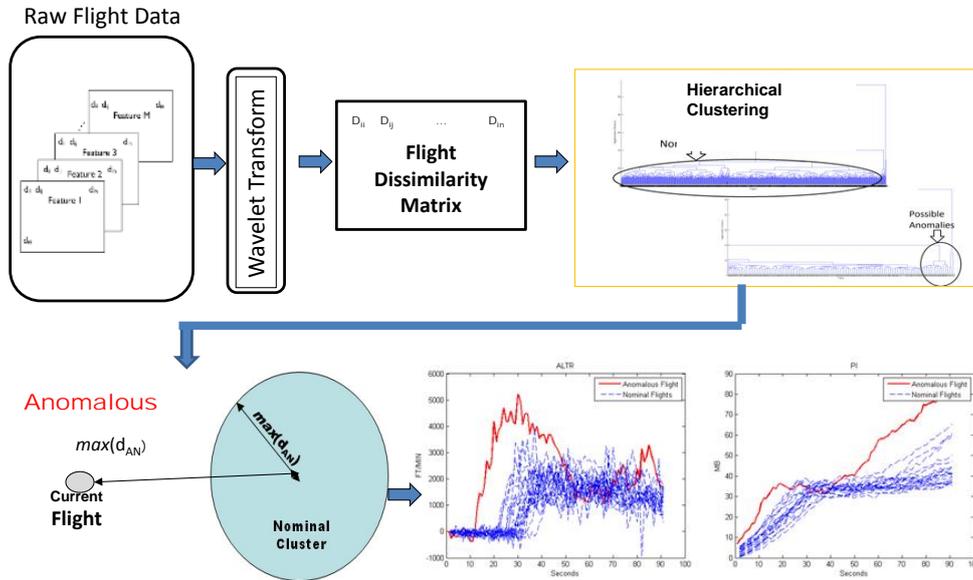


Figure 7: Data Driven Anomaly Detection Approach for Aircraft Flights

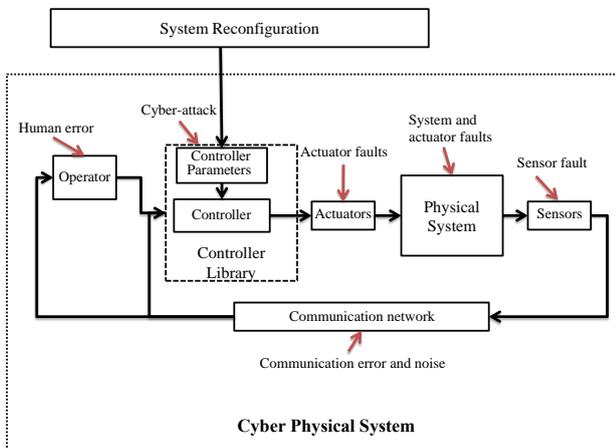


Figure 8: Possible faults in a CPS.

is a small leak, normally the system can tolerate it depending on where the leak is, but the leak usually grows over time. Therefore we need to estimate the leakage rate and re-configure the system to move the fuel from the tank or close the pipe before critical situation.

5 Conclusions

Data-driven approaches to the analysis and diagnosis of Cyber-Physical Systems (CPSs) are always inferior to classical model-based approaches, where models are created manually by experts: Experts have background knowledge which can not be learned from models and experts automatically think about a larger set of system scenarios than can be observed during a system’s normal lifetime.

So the question is not whether data-driven or expert-driven approaches are superior. The question is rather which kind of models can we realistically expect to exist in real-world applications—and which kind of models must therefore be learned automatically. This becomes especially important in the context of CPSs since these systems adapt themselves to their environment and show therefore a changing behavior, i.e. models would also have be

adapted frequently.

In Sections 4.1 and 4.2, structural information about the plant is imported from the engineering chain and the temporal behavior is learned in form of timed automata. In Section 4.5, an abstract system model describing the input/output structure and the main failure types is provided and again the behavior is learned. These approaches are typical because in most applications structural information can be gained from earlier engineer phases while behavior models hardly exist and are almost never validated with the real system.

Looking at the learning phase, the first thing to notice is that all described approaches work and deliver good results: For CPSs, data-driven approaches have moved into the focus of research and industry. And they are well suited for CPSs: They adjust automatically to new system configurations, they do not need manual engineering efforts and they make usage of the now available large number of data signals—connectivity being a typical feature of CPSs.

Another common denominator of the described application examples is that the focus is on anomaly detection rather than on root cause analysis: for data-driven approaches it is easier to learn a model of the normal behavior than learning erroneous behavior. And it is also typical that the only root cause analysis uses a case-based approach (Section 4.5), case-based approaches being suitable for data-driven solutions to diagnosis.

Finally, the examples show that the proposed cognitive architecture (Figure 2) matches the given examples:

Big Data Platform: Only a few examples (e.g. Section 4.3) make usage of explicit big data platforms, so-far solutions often use proprietary solutions. But with the growing size of the data involved, new platforms for storing and processing the data are needed.

Learning: All examples employ machine learning technologies—with a clear focus on unsupervised learning techniques which require no a-priori knowledge such as clustering (Section 4.4) or automata identification (Sections 4.1, 4.2).

Conceptual Layer: In all examples, the learned models are evaluated on a conceptual or symbolic level: In Section 4.4, clusters are compared to new observations and data-cluster

distances are used for decision making. In Sections 4.1 and 4.2, model predictions are compared to observations. And again, derivations are decided on by a conceptual layer.

Task-Specific HMI: None of the given examples works completely automatically, in all cases the user is involved in the decision making.

Adaption: In most cases, reactions to detected problems are up to the expert. The use case from Section 4.5 is an example for an automatic reaction and the usage of analysis results for the control mechanism.

Using such a cognitive architecture would bring several benefits to the community: First of all, algorithms and technologies in the different layers can be changed quickly and can be re-used. E.g. learning algorithms from one application field can be put on top of different big data platforms. Furthermore, currently most existing approaches mix the different layers, making the comparison of approaches to the analysis of CPSs difficult. Finally, such an architecture helps to clearly identify open issues for the development of smart monitoring systems.

Acknowledgments The work was partly supported by the German Federal Ministry of Education and Research (BMBF) under the project "Semantics4Automation" (funding code: 03FH02013), under the project "Analyse großer Datenmengen in Verarbeitungsprozessen (AGATA)" (funding code: 01IS14008 A-F) and by NASA NRA NNL09AA08B from the Aviation Safety program. We also acknowledge the contributions of Daniel Mack, Dinkar Mylaraswamy, and Raj Bharadwaj on the aircraft fault diagnosis work.

References

- [1] E.A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369, 2008.
- [2] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference, DAC '10*, pages 731–736, New York, NY, USA, 2010. ACM.
- [3] Peter C. Evans and Marco Annunziata. Industrial internet: Pushing the boundaries of minds and machines. Technical report, GE, 2012.
- [4] Promotorengruppe Kommunikation. Im fokus: Das industrieprojekt industrie 4.0, handlungsempfehlungen zur umsetzung. Forschungsunion Wirtschaftswissenschaft, March 2013.
- [5] L. Christiansen, A. Fay, B. Opgenoorth, and J. Neidig. Improved diagnosis by combining structural and process knowledge. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, Sept 2011.
- [6] Rolf Isermann. Model-based fault detection and diagnosis - status and applications. In *16th IFAC Symposium on Automatic Control in Aerospace*, St. Petersburg, Russia, 2004.
- [7] Johan de Kleer, Bill Janssen, Daniel G. Bobrow, Tolga Kurtoglu Bhaskar Saha, Nicholas R. Moore, and Saravan Sutharshana. Fault augmented modelica models. *The 24th International Workshop on Principles of Diagnosis*, pages 71–78, 2013.
- [8] D. Klar, M. Huhn, and J. Gruhser. Symptom propagation and transformation analysis: A pragmatic model for system-level diagnosis of large automation systems. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–9, Sept 2011.
- [9] GE. The rise of big data - leveraging large time-series data sets to drive innovation, competitiveness and growth - capitalizing on the big data opportunity. Technical report, General Electric Intelligent Platforms, 2012.
- [10] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Terziyan. Smart semantic middleware for the internet of things. In *5th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2008.
- [11] Michael Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53(4):10–11, 2010.
- [12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41.3:1–72, Sept 2009.
- [13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, page 10, June 2010.
- [14] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10, May 2010.
- [15] M JAYASREE. Data mining: Exploring big data using hadoop and mapreduce. *International Journal of Engineering Sciences Research-IJESR*, 4(1), 2013.
- [16] Friedhelm Nachreiner, Peter Nickel, and Inga Meyer. Human factors in process control systems: The design of human-machine interfaces. *Safety Science*, 44(1):5–26, 2006.
- [17] Sicco Verwer. *Efficient Identification of Timed Automata: Theory and Practice*. PhD thesis, Delft University of Technology, 2010.
- [18] Oliver Niggemann, Benno Stein, Asmir Vodenčarević, Alexander Maier, and Hans Kleine Büning. Learning behavior models for hybrid timed systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.
- [19] Bjoern Kroll, David Schaffranek, Sebastian Schriegel, and Oliver Niggemann. System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants. In *19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep 2014.
- [20] D.L.C. Mack, G. Biswas, X. Koutsoukos, and D. Mylaraswamy. Learning bayesian network structures to augment aircraft diagnostic reference model, "to appear". *IEEE Transactions on Automation Science and Engineering*, 17:447–474, 2015.

- [21] Daniel LC Mack. *Anomaly Detection from Complex Temporal Spaces in Large Data*. PhD thesis, Vanderbilt University, Nashville, TN, USA, 2013.
- [22] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
- [23] Jiang Jin. Fault tolerant control systems - an introductory overview. *Acta Automatica Sinica*, 31(1):161–174, 2005.
- [24] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and fault-tolerant control*. Springer-Verlag, Sep 2003.
- [25] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. In *In Proceedings of the IEEE*, volume 95, pages 163 – 187, Jan 2007.
- [26] B. Schneier. Security monitoring: Network security for the 21st century. In *Computers Security*, 2001.
- [27] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.
- [28] Hongm Wang, Tian-You Chai, Jin-Liang Ding, and Martin Brown. Data driven fault diagnosis and fault tolerant control: some advances and possible new directions. *Acta Automatica Sinica*, 25(6):739–747, 2009.
- [29] Z. S. Hou and J. X. Xu. On data-driven control theory: the state of the art and perspective. *Acta Automatica Sinica*, 35:650–667, 2009.

Diagnosing Advanced Persistent Threats: A Position Paper

Rui Abreu and Danny Bobrow and Hoda Eldardiry and Alexander Feldman and John Hanley and Tomonori Honda and Johan de Kleer and Alexandre Perez

Palo Alto Research Center

3333 Coyote Hill Rd

Palo Alto, CA 94304, USA

{rui,bobrow,hoda.eldardiry,afeldman,john.hanley,tomo.honda,dekleeer,aperez}@parc.com

Dave Archer and David Burke

Galois, Inc.

421 SW 6th Avenue, Suite 300

Portland, OR 97204, USA

{dwa,davidb}@galois.com

Abstract

When a computer system is hacked, analyzing the root-cause (for example entry-point of penetration) is a diagnostic process. An audit trail, as defined in the National Information Assurance Glossary, is a security-relevant chronological (set of) record(s), and/or destination and source of records that provide evidence of the sequence of activities that have affected, at any time, a specific operation, procedure, or event. After detecting an intrusion, system administrators manually analyze audit trails to both isolate the root-cause and perform damage impact assessment of the attack. Due to the sheer volume of information and low-level activities in the audit trails, this task is rather cumbersome and time intensive. In this position paper, we discuss our ideas to automate the analysis of audit trails using machine learning and model-based reasoning techniques. Our approach classifies audit trails into the high-level activities they represent, and then reasons about those activities and their threat potential in real-time and forensically. We argue that, by using the outcome of this reasoning to explain complex evidence of malicious behavior, we are equipping system administrators with the proper tools to promptly react to, stop, and mitigate attacks.

1 Introduction

Today, enterprise system and network behaviors are typically “opaque”: stakeholders lack the ability to assert causal linkages in running code, except in very simple cases. At best, event logs and audit trails can offer some partial information on temporally and spatially localized events as seen from the viewpoint of

individual applications. Thus current techniques give operators little system-wide situational awareness, nor any viewpoint informed by a long-term perspective. Adversaries have taken advantage of this opacity by adopting a strategy of persistent, low-observability operation from inside the system, hiding effectively through the use of long causal chains of system and application code. We call such adversaries *advanced persistent threats*, or APTs.

To address current limitations, this position paper discusses a technique that aims to track causality across the enterprise and over extended periods of time, identify subtle causal chains that represent malicious behavior, localize the code at the roots of such behavior, trace the effects of other malicious actions descended from those roots, and make recommendations on how to mitigate those effects. By doing so, the proposed approach aims to enable stakeholders to understand and manage the activities going on in their networks. The technique exploits both current and novel forms of local causality to construct higher-level observations, long-term causality in system information flow. We propose to use a machine learning approach to classify segments of low-level events by the activities they represent, and reasons over these activities, prioritizing candidate activities for investigation. The diagnostic engine investigates these candidates looking for patterns that may represent the presence of APTs. Using pre-defined security policies and related mitigations, the approach explains discovered APTs and recommends appropriate mitigations to operators. We plan to leverage models of APT and normal business logic behavior to diagnose such threats. Note that the technique is not constrained by availability of human analysts, but can benefit by human-on-the-loop assistance.

The approach discussed in the paper will offer unprecedented capability for *observation* of long-term, subtle system-wide activity by automatically constructing such global, long-term causality observa-

tions. The ability to automatically classify causal chains of events in terms of abstractions such as activities, will provide operators with a unique capability to *orient* to long-term, system-wide evidence of possible threats. The diagnostic engine will provide a unique capability to identify whether groups of such activities likely represent active threats, making it easier for operators to *decide* whether long-term threats are active, and where they originate, even before those threats are identified by other means. Thus, the approach will pave the way for the first automated, long-horizon, continuously operating system-wide support for an effective defender Observe, Orient, Decide, and Act (OODA) loop.

2 Running Example

The methods proposed in this article are illustrated on a *realistic* running example. The attackers in this example use sophisticated and recently discovered exploits to gain access to the victim's resources. The attack is remote and *does not* require social engineering or opening a malicious email attachment. The methods that we propose, however, are not limited to this class of attacks.

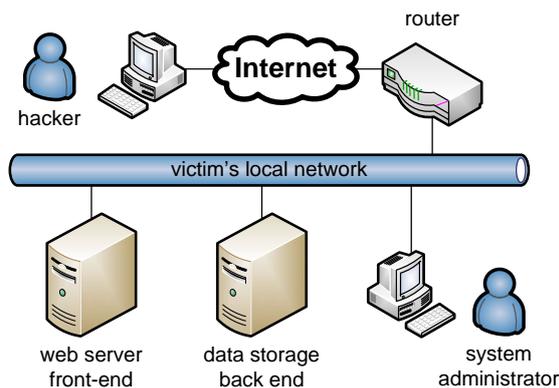


Figure 1: Network topology for the attack

The network topology used for our running example is shown in figure 1. The attack is executed over several days. It starts by (1) compromising the web server front-end, followed by (2) a reconnaissance phase and (3) compromising the data storage back end and ultimately extracting and modifying sensitive information belonging to the victim.

Both the front-end and the back end in this example run unpatched UBUNTU 13.1 LINUX OS on an INTEL® SANDY BRIDGE™ architecture.

What follows is a detailed chronology of the events:

1. The attacker uses the APACHE httpd server, a cgi-bin script, and the SHELLSHOCK vulnerability (GNU bash exploit registered in the Common Vulnerabilities and Exposures database as CVE 2014-6271 (see <https://nvd.nist.gov/>) to gain remote shell access to the victim's front-end. It is now possible for the attacker to

execute processes on the front-end as the non-privileged user www-data.

2. The attacker notices that the front-end is running an unpatched UBUNTU LINUX OS version 13.1. The attacker uses the nc Linux utility to copy an exploit for obtaining root privileges. The particular exploit that the attacker uses utilizes the x32 recvmsg() kernel vulnerability registered in the Common Vulnerabilities and Exposures (CVE) database as CVE 2014-0038. After running the copied binary for a few minutes the attacker gains root access to the front-end host.
3. The attacker installs a root-kit utility that intercepts all input to ssh;
4. A system administrator uses the compromised ssh to connect to the back-end revealing his back-end password to the attacker;
5. The attacker uses the compromised front-end to bypass firewalls and uses the newly acquired back-end administrator's password to access the back-end;
6. The attacker uses a file-tree traversing utility on the back-end that collects sensitive data and consolidates it in an archive file;
7. The attacker sends the archive file to a third-party hijacked computer for analysis.

3 Auditing and Instrumentation

Almost all computing systems of sufficiently high-level (with the exception of some embedded systems) leave detailed logs of all system and application activities. Many UNIX variants such as LINUX log via the syslog daemon, while WINDOWS™ uses the event log service. In addition to the usual logging mechanisms, there is a multitude of projects related to secure and detailed auditing. An audit log is more detailed trail of any security or computation-related activity such as file or RAM access, system calls, etc.

Depending on the level of security we would like to provide, there are several methods for collecting input security-related information. On one extreme, it is possible to use the existing log files. On the other extreme there are applications for collecting detailed information about the application execution. One such approach [1] runs the processes of interest through a debugger and logs every memory read and write access.

It is also possible to automatically inject logging calls in the source files before compiling them, allowing us to have static or dynamic logging or a combination of the two. Log and audit information can be signed, encrypted and sent in real-time to a remote server to make system tampering and activity-hiding more difficult. All these configuration decisions impose different trade-offs in security versus computational and RAM load [2] and depend on the organizational context.

```

:
front_end.secure_access_log:11.239.64.213 - [22/Apr/2014
06:30:24 +0200] "GET /cgi-bin/test.cgi HTTP/1.1" 401 381
:
front_end.rsyslogd.log:recvmmsg(3, msg_name(0) =
NULL, msg_iov(1) = ["29/Apr/2014 22:15:49 ...", 8096],
msg_controllen = 0, msg_flags = MSG_CTRUNC,
MSG_DONTWAIT) = 29
:
back_end.auditctl:type = SYSCALL msg = au-
dit(1310392408.506:36): arch = c000003e syscall = 2
success = yes exit = 3 a0 = 7fff2ce9471d a1 = 0 a2 = 61f768
a3 = 7fff2ce92a20 items = 1 ppid = 20478 pid = 21013 auid
= 1000 uid = 0 gid = 0 euid = 0 suid = 0 fsuid = 0 egid = 0
sgid = 0 fsgid = 0 ses = 1 comm = "grep" exe = "/bin/grep"
:

```

Figure 2: Part of log files related to the attack from the running example

Figure 2 shows part of the logs collected for our running example. The first entry is when the attacker exploits the SHELLSHOCK vulnerability through a CGI script of the web server. The second entry shows `syslog` `strace`-like message resulting from the kernel escalation. Finally, the attacker uses the `grep` command on the back-end server to search for sensitive information and the call is recorded by the audit system.

It is often the case that the raw system and security log files are preprocessed and initial causal links are computed. If we trace the `exec`, `fork`, and `join` POSIX system calls, for example, it is possible to add graph-like structure to the log files computing provenance graphs. Another method for computing local causal links is to consider shared resources, e.g., two threads reading and writing the same memory address [1].

4 Activity Classification

The Activity Classifier continuously annotates audit trails with semantic tags describing the higher-order activity they represent. For example, ‘remote shell access’, ‘remote file overwrite’, and ‘intra-network data query’ are possible activity tags. These tags are used by the APT Diagnostics Engine to enable higher-order reasoning about related activities, and to prioritize activities for possible investigation.

4.1 Hierarchical semantic annotation of audit trails

A key challenge in abstracting low-level events into higher-order activity patterns that can be reasoned about efficiently is that such patterns can be described at multiple levels of semantic abstraction, all of which may be useful in threat analysis. Indeed, higher-order

abstractions may be composed of lower-order abstractions that are in turn abstractions of low-level events. For example, a sequential set of logged *events* such as ‘browser forking bash’, ‘bash initiating Netcat’, and ‘Netcat listening to new port’, might be abstracted as the *activity* ‘remote shell access’. The set of activities, ‘remote shell access’, and ‘escalation of privilege’ can be abstracted as the activity ‘remote **root** shell access’.

We approach activity annotation as a supervised learning problem that uses classification techniques to generate activity tags for audit trails. Table 1 shows multiple levels of activity classifications for the above APT example.

Table 1 represents one possible classification-enriched audit trail for such an APT. There can be many relatively small variations. For example, obscuring the password file could be done using other programs. A single classifier only allows for a single level of abstraction, and a single leap from low-level events to very abstract activities (for example, from ‘bash execute perl’ level to ‘extracting modified file’ level) will have higher error caused by these additional variations.

To obtain several layers of abstraction for reasoning over, and thus reduce overall error in classification, we use a *multi-level learning* strategy that models information at multiple levels of semantic abstraction using multiple classifiers. Each classifier solves the problem at one abstraction level, by mapping from a lower-level (fine) feature space to the next higher-level conceptual (coarse) feature space.

The activity classifier rely on both a vocabulary of activities and a library of patterns describing these activities that will be initially defined manually. This vocabulary and pattern set reside in a Knowledge Base.

In our training approach, results from training lower level classifiers are used as training data for higher level classifiers. In this way, we coherently train all classifiers by preventing higher-level classifiers from being trained with patterns that will never be generated by their lower-level precursors. We use an ensemble learning approach to achieve accurate classification. This involves stacking together both bagged and boosted models to reduce both variance and bias error components [3]. The classification algorithm will be trained using an online-learning technique and integrated within an Active Learning Framework to improve classification of atypical behaviors.

Generating Training Data for Classification To build the initial classifier, training data is generated using two methods. First, an actual deployed system is used to collect normal behavior data, and a Subject Matter Expert manually labels it. Second, a testing platform is used to generate data in a controlled environment, particularly platform dependent vulnerability-related behavior. In addition, to generate new training data of previously unknown behavior, we use an Active Learning framework as described in Section 5.

Table 1: Sample classification problem for running example

Activity 1 <i>Remote Shell Access</i> Shell Shock	Activity 2 <i>Remote File Overwrite</i> Trojan Installation	Activity 3 <i>Modified File Download</i> Password Exfiltration
Browser (Port 80) fork bash	Netcat listen to Port 8443	Netcat listen to Port 8443
bash fork Netcat	Port 8443 receive binary file	Port 8443 fork bash
Netcat listen to port 8080	binary file overwrites libns.so	bash execute perl
		Perl overwrite /tmp/stolen_pw
		Port 8443 send /tmp/stolen_pw

5 Prioritizer

As the Activity Classifier annotates audit trails with activity descriptors, the two (parallel) next steps in our workflow are to 1) prioritize potential threats to be referred to the Diagnostic Engine (see Section 6) for investigation, and 2) prioritize emergent activities that (after suitable review and labeling) are added to the activity classifier training data. This module prioritizes activities by *threat severity* and *confidence level*. This prioritization process presents three key challenges.

5.1 Threat-based rank-annotation of activities

One challenge in ranking activities according to their threat potential is the complex (and dynamic) notion of what constitutes a threat. Rankings based on matching to known prior threats is necessary, but not sufficient. An ideal ranking approach should take known threats into account, while also proactively considering the unknown threat potential of new kinds of activities. Another such challenge is that risk may be assessed at various levels of activity abstraction, requiring that overall ranking must be computed by aggregating risk assessments at multiple abstraction levels.

We implement two ranking approaches: a *supervised* ranker based on previously known threats and an *unsupervised* ranker that considers unknown potential threats.

Supervised ranking using APT classification to catch known threats. The goal of APT classification is to provide the diagnostic engine with critical APT related information such as APT Phase, severity of attack, and confidence level associated with APT tagging for threat prioritization. Since the audit trails are annotated hierarchically into different granularity of actions, multiple classifiers will be built to consider each hierarchical level separately. APT classifiers are used to identify entities that are likely to be instances of known threats or phases of an APT attack. Two types of classifiers are used. The first classifier is hand-coded and the second classifier is learned from training data.

The hand-coded classifier is designed to have high precision, using hand-coded rules, mirroring SIEM and IDS systems. Entities tagged by this classifier are given the highest priority for investigation. The second classifier, which is learned from training data, will provide higher recall at the cost of precision. Activities

are ranked according to their threat level by aggregating a severity measure (determined by classified threat type) and a confidence measure. We complement the initial set of training data to calibrate our classifiers by using an Active Learning Framework, which focuses on improving the classification algorithm through occasional manual labeling of the most critical activities in the audit trails.

Unsupervised ranking using normalcy characterization to catch unknown threats. The second component of the prioritizer is a set of unsupervised *normalcy rankers*, which rank entities based on their statistical “normalcy”. Activities identified as *unusual* will be fed to the Active Learning framework to check if any of them are “unknown” APT activities. This provides a mechanism for detecting “unknown” threats while also providing feedback to improve the APT classifier.

5.2 Combining Multiple Rankings

One of the key issues with combining the outputs of multiple risk ranking is dealing with two-dimensional risk (severity, confidence) scores that may be on very different scales. A diverse set of score normalization techniques have been proposed [4; 5; 6] to deal with this issue, but no single technique has been found to be superior over all the others. An alternative to combining scores is to combine rankings [7]. Although converting scores to rankings does lose information, it remains an open question if the loss in information is compensated for by the convenience of working with the common scale of rankings.

We will develop combination techniques for weighted risk rankings based on probabilistic rank aggregation methods. This approach builds on our own work [8] that shows the robustness of the weighted ranking approach. We also build on principled methods for combining ranking data found in the statistics and information retrieval literature.

Traditionally, the goal of rank aggregation [9; 10] is to combine a set of rankings of the same candidates into a single consensus ranking that is “better” than the individual rankings. We extend the traditional approach to accommodate the specific context of weighted risk ranking. First, unreliable rankers will be identified and either ignored or down-weighted, lest their rankings decrease the quality of the overall consensus [7; 10]. Second, we will discount excessive correlation among rankers, so that a set of

highly redundant rankers do not completely outweigh the contribution of other alternative rankings. To address these two issues, we will associate a probabilistic latent variable Z_i with the i 'th entity of interest, which indicates whether the entity is anomalous or normal. Then, we will build a probabilistic model that allows us to infer the posterior distribution over the Z_i based on the observed rankings produced by each of the input weighted risk rankings. This posterior probability of Z_i being normal will then be used as the weighted risk rank. Our model will make the following assumptions to account for both unreliable and correlated rankers: 1) Anomalies are ranked lower than all normal instances and these ranks tend to be concentrated near the lower rankings of the provided weighted risk rankings, and 2) Normal data instances tend to be uniformly distributed near the higher rankings of the weighted risk rankings.

There are various ways to build a probabilistic model that reflects the above assumptions and allows for the inference of the Z_i variables through Expectation-Maximization [11]. In addition to these assumptions, we will explore allowing other factors to influence the latent Z_i variables, such as features of the entities as well as feedback provided by an expert analysts.

6 Diagnosis

We view the problem of detecting, isolating, and explaining complex APT campaigns behavior from rich activity data is a diagnostic problem. We will use an AI-based diagnostic reasoning to guide the global search for possible vulnerabilities that enabled the breach. Model-based diagnosis (MBD) [12] is a particularly compelling approach as it supports reasoning over complex causal networks (for example, having multiple conjunctions, disjunctions, and negations) and identifies often subtle combinations of root causes of the symptoms (the breach).

6.1 An MBD approach for APT detection and isolation: Motivation

Attack detection and isolation are two distinct challenges. Often diagnostic approaches use separate models for detection and isolation [13]. MBD however uses a *single model*, to combine these two reasonings. The security model contains both part of the security policy (that communicating with certain blacklisted hosts may indicate an information leak) and information about the *possible* locations and consequence of a vulnerability (a privilege escalation may lead to an information leak). The security model also contains abstract security constraints such as if a process requires authentication, a password must be read and compared against.

The diagnostic approach takes into consideration the bootstrapping of an APT which we consider the root-cause of the attack. What enables a successful APT is either a *combination* of software component

vulnerabilities or the combined use of social engineering and insufficiency of the organizational security policies. We use MBD for computing *the set of simultaneously exploited vulnerabilities* that allowed the deployment of the APT. Computing such explanations is possible because MBD reasons in terms of multiple-faults [14]. In our running example this set would include both the fact the the web server has been exploited due to the Shellshock vulnerability and that a the attacker gained privileged access on the front-end due to the use of the X64_32 escalation vulnerability.

The abstract security model is used to gather information about types of attacks the system is vulnerable to, and to aid deciding the set of actions required to stop an APT campaign (policy enforcement). Various heuristics exist to find the set of meaningful diagnosis candidates. As an example, one might be interested in the minimal set of actions to stop the attack [15; 16] or select those candidates that capture significant probability mass [17]. In the rest of this section, for illustration purposes, we use minimality as the heuristic of interest. MBD is the right tool for dealing with computation of diagnosis candidates as it offers several ways to address the modeling and computational complexity [18; 19].

6.2 Detection and Isolation of Attacks from Abstract Security Model and Sensor Data

The abstract security model provides an abstraction mechanism that is originally missing in the audit trails. More precisely what is not in the audit trails and what is in the security model is how to connect (possibly disconnected) activities for the purpose of global reasoning. The abstract security model and the sensor data collected from the audit trails are provided as inputs to an MBD algorithms that performs the high-level reasoning about possible vulnerabilities and attacks similar to what a human security officer would do.

The information in the “raw” audit trails is of too high fidelity [2] and low abstraction to be used by a “crude” security model. That is the reason the diagnostic engine needs the machine learning module to temporally and spatially group nodes in the audit trails and to provide semantically rich variable/value sensor data about actions, suitable for MBD. Notice that in this process, the audit trail structure is translated to semantic categories, i.e., the diagnostic engine receives as observations time-series of sensed actions.

The listing that follows next shows an abstract security model for the running example in the LYDIA language [20]. This bears some resemblance to PROLOG, except that LYDIA is a language for model-based diagnosis of logical circuits while PROLOG is for Horn-style reasoning. The use of LYDIA is for illustration purposes only, in reality computer systems can be much more easily modeled as state machines. There is a significant body of literature dealing with diagnosis of discrete-event systems [21; 22; 23], to name just a few.

```

1  system front_end(bool know_root_password)
2  {
3      bool httpd_shell_vuln; // vulnerability
4      bool buffer_overflow_vuln; // vulnerability
5      bool escalation_vuln; // vulnerability
6
7      bool httpd_shell;
8      bool root_shell;
9      bool leak_passwd;
10
11     // weak-fault models
12     if (!httpd_shell_vuln) { // if healthy
13         !httpd_shell; // forbid shells via httpd
14     }
15
16     if (!escalation_vuln) { // if healthy
17         !root_shell; // no root shell is possible
18     }
19
20     if (!buffer_overflow_vuln) { // if healthy
21         !leak_passwd; // passwords don't leak
22     }
23
24     bool access_passwd;
25     attribute observable(access_passwd) = true;
26
27     !access_passwd => !leak_passwd;
28
29     /**
30     * Knowing the root password can be explained
31     * by a root shell (for example there is a
32     * password sniffer).
33     */
34     know_root_password =>
35     ((httpd_shell || leak_passwd) && root_shell);
36 }
37
38 system back_end(bool know_root_password)
39 {
40     bool comm;
41     attribute observable(comm) = true;
42
43     /**
44     * Normal users can only communicate with a
45     * list of permitted hosts.
46     */
47     if (!know_root_password) {
48         comm == true;
49     }
50 }
51
52 system main()
53 {
54     bool know_root_password;
55
56     system front_end fe(know_root_password);
57     system back_end be(know_root_password);
58 }

```

LYDIA translates the model to an internal propositional logic formula. Part of this internal representation is shown in figure 3, which uses the standard VLSI [24] notation to denote AND-gates, OR-gates, and NOT-gates. Wires are labeled with variable names. Boolean circuits (matching propositional logic), however, have limited expressiveness and modeling secu-

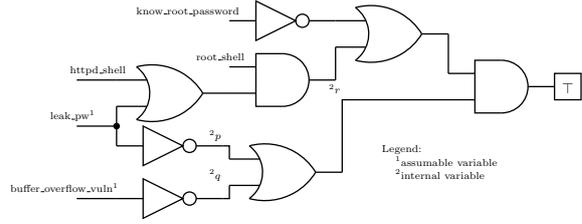


Figure 3: Part of the abstract security model for the running example

rity constraints in it is notoriously difficult, hence we plan to create or use specialized modal logic similar to the one proposed in [25].

Notice that the format of the Boolean circuit shown in figure 3 is very close to the one used in Truth Maintenance System (TMS) [26]. The only assumable variable in figure 3 is `buffer_overflow_vuln` and its default value is false (i.e., there is no buffer overflow vulnerability in the web server process).

We next show how a reasoning engine can discover a conflict through forward and backward propagation. Looking at figure 3, it is clear that r must be true because it is an input to an AND-gate whose output is set to true. Therefore either p or q (or both) must be true. This means that either `buffer_overflow_vuln` or `leak_pw` must be false. If we say that `leak_pw` is assumed to be true (measured or otherwise inferred), then `leak_pw` and `buffer_overflow_vuln` are together part of a conflict. It means that the reasoning engine has to change one of them to resolve the contradiction.

Based on the observation from our running example and a TMS constructed from the security model shown in figure 3, the hitting set algorithm computes two possible diagnostic hypotheses: (1) the attacker gained a shell access through a web-server vulnerability *and* the attacker performed privilege escalation *or* (2) the attacker injected binary code through a buffer overflow *and* the attacker performed privilege escalation.

If we use LYDIA to compute the set of diagnoses for the running example, we get the following two (ambiguous) diagnoses for the root-cause of the penetration:

```

$ lydia example.lm example.obs
d1 = { fe.escalation_vuln,
      fe.httpd_shell_vuln }
d2 = { fe.buffer_overflow_vuln,
      fe.escalation_vuln }

```

MBD uses probabilities to compute a sequence of possible diagnoses ordered by likelihood. This probability can be used for many purposes: decide which diagnosis is more likely to be the true fault explanation, whether there is the need for consider further evidence from the logs or limit the number of diagnoses that need to be identified. Many policies exist to compute these probabilities [27; 28].

For illustration purposes we consider that the diag-

noses for the running example are ambiguous. Before we discuss methods for dealing with this ambiguity, we address the major research challenge of model generation.

6.3 Model Generation

The abstract vulnerability model can either be constructed manually or semi-automatically. The challenge with modeling is that an APT campaign generally exploits unknown vulnerabilities. Therefore, our approach to address this issue is to construct the model which captures expected behavior (*known goods*) of the system. Starting from generic parameterized vulnerability models and security objectives, the abstract vulnerability model can be extended with information related to known vulnerabilities (*known bads*).

Generating the model can be done either manually or semi-automatically. We will explore venues to generate this model manually, which requires significant knowledge about potential security vulnerabilities, while being error prone and not detailed enough. Amongst company specific requirements, we envisage the abstract vulnerability model to capture the most common attacks that target software systems, as described in the Common Attack Pattern Enumeration and Classification (CAPEC¹). The comprehensive list of known attacks has been designed to better understand the perspective of an attacker exploiting the vulnerabilities and, from this knowledge, devise appropriate defenses.

As modeling is challenging, we propose to explore semi-automatic approaches to construct models. The semi-automatic method is suitable to addressing the modeling because in security, similarly to diagnosis, there is (1) component models and (2) structure. While it is difficult to automate the building of component models (this may even require natural language parsing of databases such as CAPEC), it is feasible to capture diagnosis-oriented information from structure (physical networking or network communication).

Yet another approach to semi-automatically generate the model is to learn it from executions of the system (e.g., during regression testing, just before deployment). This approach to system modeling is inspired by the work in automatic software debugging work [29], where modeling of program behavior is done in terms of abstraction of program traces – known as spectra [30], abstracting from modeling specific components and data dependencies

The outlined approaches to construct the abstract vulnerability model entail different costs and diagnostic accuracies. As expected, manually building the model is the most expensive one. Note that building the model is a time-consuming and error-prone task. The two semi-automatic ways also entail different costs: one exploits the available, static information and the other requires the system to be executed to compute a meaningful set of executions. We will investigate the trade-offs between modeling approaches

and their diagnostic accuracy in the context of transparent computing.

7 Conclusions

Identifying the root-cause and perform damage impact assessment of advanced persistent threats can be framed as a diagnostic problem. In this paper, we discuss an approach that leverages machine learning and model-based diagnosis techniques to reason about potential attacks.

Our approach classifies audit trails into high-level activities, and then reasons about those activities and their threat potential in real-time and forensically. By using the outcome of this reasoning to explain complex evidence of malicious behavior, the system administrators is provided with the proper tools to promptly react to, stop, and mitigate attacks.

References

- [1] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. High accuracy attack provenance via binary-based execution partition. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, San Diego, CA, February 2013.
- [2] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. LogGC: Garbage collecting audit log. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pages 1005–1016, Berlin, Germany, 2013.
- [3] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, July 2012.
- [4] Charu C Aggarwal. Outlier ensembles: Position paper. *ACM SIGKDD Explorations Newsletter*, 14(2):49–58, 2013.
- [5] Jing Gao and Pang-Ning Tan. Converting outlier scores from outlier detection algorithms into probability estimates. In *Proceedings of the Sixth International Conference on Data Mining*, pages 212–221. IEEE, December 2006.
- [6] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pages 13–24, April 2011.
- [7] Erich Schubert, Remigius Wojdanowski, Arthur Zimek, and Hans-Peter Kriegel. On evaluation of outlier rankings and outlier scores. In *Proceedings of the Twelfth SIAM International Conference on Data Mining*, pages 1047–1058, April 2012.

¹<http://capec.mitre.org/>

- [8] Hoda Eldardiry, Kumar Sricharan, Juan Liu, John Hanley, Bob Price, Oliver Brdiczka, and Eugene Bart. Multi-source fusion for anomaly detection: using across-domain and across-time peer-group consistency checks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 5(2):39–58, 2014.
- [9] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4(Nov):933–969, 2003.
- [10] Ke Deng, Simeng Han, Kate J Li, and Jun S Liu. Bayesian aggregation of order-based rank data. *Journal of the American Statistical Association*, 109(507):1023–1039, 2014.
- [11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B*, 39(1):1–38, 1977.
- [12] Johan de Kleer, Olivier Raiman, and Mark Shirley. One step lookahead is pretty good. In *Readings in Model-Based Diagnosis*, pages 138–142. Morgan Kaufmann Publishers, San Francisco, CA, 1992.
- [13] Alexander Feldman, Tolga Kurtoglu, Sriram Narasimhan, Scott Poll, David Garcia, Johan de Kleer, Lukas Kuhn, and Arjan van Gemund. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*, pages 1–28, 2010.
- [14] Johan de Kleer and Brian Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.
- [15] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proceeding of the 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, May 2002.
- [16] Seyit Ahmet Camtepe and Bülent Yener. Modeling and detection of complex attacks. In *Proceeding of the Third International Conference on Security and Privacy in Communications Networks*, pages 234–243, September 2007.
- [17] Rui Abreu and Arjan JC van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *Proceedings of the Eighth Symposium on Abstraction, Reformulation and Approximation*, pages 2–9, July 2009.
- [18] Alexander Feldman, Gregory Provan, and Arjan van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, 38:371–413, 2010.
- [19] Nuno Cardoso and Rui Abreu. A distributed approach to diagnosis candidate generation. In *Progress in Artificial Intelligence*, pages 175–186. Springer, 2013.
- [20] Alexander Feldman, Jurryt Pietersma, and Arjan van Gemund. All roads lead to fault diagnosis: Model-based reasoning with LYDIA. In *Proceedings of the Eighteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC’06)*, Namur, Belgium, October 2006.
- [21] Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen, and Demosthenis C Teneketzis. Failure diagnosis using discrete-event models. *Control Systems Technology, IEEE Transactions on*, 4(2):105–124, 1996.
- [22] Alban Grastien, Marie-Odile Cordier, and Christine Largouët. Incremental diagnosis of discrete-event systems. In *DX*, 2005.
- [23] Alban Grastien, Patrik Haslum, and Sylvie Thiébaux. Conflict-based diagnosis of discrete event systems: theory and practice. 2012.
- [24] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Inc., New York, NY, USA, 2nd edition, 2009.
- [25] Janice Glasgow, Glenn Macewen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.
- [26] Kenneth Forbus and Johan de Kleer. *Building Problem Solvers*. MIT Press, 1993.
- [27] Johan de Kleer. Diagnosing multiple persistent and intermittent faults. In *Proceeding of the 2009 International Joint Conference on Artificial Intelligence*, pages 733–738, July 2009.
- [28] Rui Abreu, Peter Zoetewij, and Arjan J. C. Van Gemund. A new bayesian approach to multiple intermittent fault diagnosis. In *Proceeding of the 2009 International Joint Conference on Artificial Intelligence*, pages 653–658, July 2009.
- [29] Rui Abreu, Peter Zoetewij, and Arjan JC Van Gemund. Spectrum-based multiple fault localization. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering*, pages 88–99, November 2009.
- [30] Mary Jean Harrold, Gregg Rothermel, Kent Sayre, Rui Wu, and Liu Yi. An empirical investigation of the relationship between spectra differences and regression faults. *Software Testing Verification and Reliability*, 10(3):171–194, 2000.

A Structural Model Decomposition Framework for Hybrid Systems Diagnosis

Matthew Daigle¹ and Anibal Bregon² and Indranil Roychoudhury³

¹NASA Ames Research Center, Moffett Field, CA 94035, USA

e-mail: matthew.j.daigle@nasa.gov

²Department of Computer Science, University of Valladolid, Valladolid, 47011, Spain

e-mail: anibal@infor.uva.es

³Stinger Ghaffarian Technologies Inc., NASA Ames Research Center, Moffett Field, CA 94035, USA

e-mail: indranil.roychoudhury@nasa.gov

Abstract

Nowadays, a large number of practical systems in aerospace and industrial environments are best represented as hybrid systems that consist of discrete modes of behavior, each defined by a set of continuous dynamics. These hybrid dynamics make the on-line fault diagnosis task very challenging. In this work, we present a new modeling and diagnosis framework for hybrid systems. Models are composed from sets of user-defined components using a compositional modeling approach. Submodels for residual generation are then generated for a given mode, and reconfigured efficiently when the mode changes. Efficient reconfiguration is established by exploiting causality information within the hybrid system models. The submodels can then be used for fault diagnosis based on residual generation and analysis. We demonstrate the efficient causality reassignment, submodel reconfiguration, and residual generation for fault diagnosis using an electrical circuit case study.

1 Introduction

Robust and efficient fault diagnosis plays an important role in ensuring the safe, correct, and efficient operation of complex engineering systems. Many engineering systems are modeled as *hybrid systems* that have both continuous and discrete-event dynamics, and for such systems, the complexity of fault diagnosis methodologies increases significantly. In this paper, we develop a new modeling framework and structural model decomposition approach that enable efficient online fault diagnosis of hybrid systems.

During the last few years, different proposals have been made for hybrid systems diagnosis, focusing on either hybrid modeling, such as hybrid automata [1–3], hybrid state estimation [4], or a combination of on-line state tracking and residual evaluation [5]. However, in all these cases, the proposed solutions involve modeling and pre-enumeration of the set of *all* possible system-level discrete modes, which grows exponentially with the number of switching components. Both steps are computationally very expensive or unfeasible for hybrid systems with complex interacting subsystems.

One solution to the mode pre-enumeration problem is to build hybrid system models in a *compositional* way, where discrete modes are defined at a local level (e.g., at the component level), in which the system-level mode is defined implicitly by the local component-level modes. Since this

allows the modeler to focus on the discrete behavior only at the component level, the pre-enumeration of all the system-level modes can be avoided [6, 7]. Additionally, building models in a compositional way facilitates reusability and maintenance, and allows the validation of the components individually before they are composed to create the global hybrid system model.

In a system model, the effects of mode changes in individual components may force other components to reconfigure their computational structures, or *causality*, during the simulation process, which requires developing efficient online causality reassignment procedures. As an example of this kind of approach, Hybrid Bond Graphs (HBGs) [8] have been used by different authors [9, 10], and efficient causality reassignment has been developed previously for such models [11]. However, the main limitation of HBGs is that the set of possible components is restricted (e.g., resistors, capacitors, 0-junctions, etc.), with each component having to conform to a certain set of mathematical constraints, and modelers do not have the liberty to define and use their own components. Another example is that of [7], which uses a more general modeling framework, and tackles the causality reassignment problem from a graph-theoretic perspective.

In this work, we propose a compositional modeling approach for hybrid systems, where models are made up of sets of user-defined components. Here, a component is constructed by defining a set of discrete modes, with a different set of mathematical constraints describing the continuous dynamics in each mode. Then, we borrow ideas for efficient causality reassignment in HBGs [11], and propose algorithms for efficient causality assignment in our component-based models, extending and generalizing those from HBGs. We then apply structural model-decomposition [12] to compute minimal submodels for the initial mode of the system. These submodels are used for fault diagnosis based on residual generation and analysis. Based on efficient causality reassignment, submodels can be reconfigured upon mode changes efficiently. Using an electrical circuit as a case study, we demonstrate efficient causality reassignment and submodel reconfiguration and show that these submodels can correctly compute system outputs for residual generation in the presence of known mode changes.

The paper is organized as follows. Section 2 presents the modeling approach and introduces the case study. Section 3 presents the overall approach for hybrid systems fault diagnosis based on structural model decomposition. Section 4 develops the causality analysis and assignment algorithms. Section 5 presents the structural model decomposition ap-

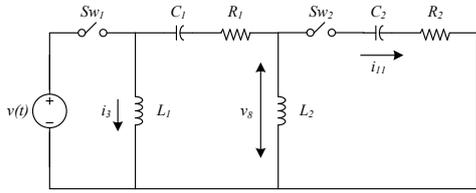


Figure 1: Electrical circuit running example.

proach. Section 6 describes efficient causality reassignment. Section 7 demonstrates the approach for the electrical case study. Section 8 reviews the related work and current approaches for hybrid systems fault diagnosis. Finally, Section 9 concludes the paper.

2 Compositional Hybrid Systems Modeling

We define hybrid system dynamics in a general compositional way, where the system is made up of a set of components. Each component is defined by a set of discrete modes, with a different set of constraints describing the continuous dynamics of the component in each mode. Here, system-level modes are defined implicitly through the composition of the component-level modes. Because the number of system-level modes is exponential in the number of switching components, we want to avoid generating and reasoning over the system-level hybrid model, instead working directly with the component models.

To illustrate our proposal, throughout the paper we will use a circuit example, shown in Fig. 1. The components of the circuit are a voltage source, V , two capacitors, C_1 and C_2 , two inductors, L_1 and L_2 , two resistors, R_1 and R_2 , and two switches, Sw_1 and Sw_2 , as well as components for series and parallel connections. Sensors measure the current or voltage in different locations (i_3 , v_8 , and i_{11} , as indicated in Fig. 1). Because each switch has two modes (on and off), there are four total modes in the system.

In the following, we present the main details of our hybrid system modeling framework, which may be viewed as an extension of our modeling approach described in [12], extended with the notion of components, and with hybrid system dynamics.

2.1 System Modeling

At the basic level, the continuous dynamics of a component in each mode are modeled using a set of *variables* and a set of *constraints*. A constraint is defined as follows:

Definition 1 (Constraint). A constraint c is a tuple (ε_c, V_c) , where ε_c is an equation involving variables V_c .

A component is defined by a set of constraints over a set of variables. The constraints are partitioned into different sets, one for each component mode. A component is then defined as follows:

Definition 2 (Component). A component δ with n discrete modes is a tuple $\delta = (V_\delta, \mathcal{C}_\delta)$, where V_δ is a set of variables and \mathcal{C}_δ is a set of constraints sets, where \mathcal{C}_δ is defined as $\mathcal{C}_\delta = \{C_\delta^1, C_\delta^2, \dots, C_\delta^n\}$, with a constraint set, C_δ^m , defined for each mode $m = \{1, \dots, n\}$.

The components of the circuit are defined in Table 1 (first three columns).

Example 1. Consider the component R_1 (δ_6). It has only a single mode with a single constraint $v_5 = i_5 * R_1$ over variables $\{v_5, i_5, R_1\}$.

Example 2. Consider the component Sw_2 (δ_{10}). It has two modes: *on* and *off*. In the *off* mode, it has three constraints setting each of its currents (i_9, i_{10}, i_{11}) to 0. In the *on* mode, it has also three constraints, setting the three currents equal to each other and establishing that the voltages sum up (it acts like a series connection when in the on mode).

We can define a system model by composing components:

Definition 3 (Model). A model $\mathcal{M} = \{\delta_1, \delta_2, \dots, \delta_d\}$ is a finite set of d components for $d \in \mathbb{N}$.

Example 3. The model of the electrical system is made up of the components detailed in Table 1, i.e., $\mathcal{M} = \{\delta_1, \delta_2, \dots, \delta_{15}\}$. For each component, the variables and constraints are defined for each component mode (third column).

Note that the set of variables for a model does not change with the mode, hence we need only a variable set in a component and not a set of variable sets as with constraints. The set of variables for a model, $V_{\mathcal{M}}$, is simply the union of all the component variable sets, i.e., for d components, $V_{\mathcal{M}} = V_{\delta_1} \cup V_{\delta_2} \cup \dots \cup V_{\delta_d}$. The interconnection structure of the model is captured using shared variables between components, i.e., we say that two components are connected if they share a variable, i.e., components δ_i and δ_j are connected if $V_{\delta_i} \cap V_{\delta_j} \neq \emptyset$. $V_{\mathcal{M}}$ consists of five disjoint sets, namely, the set of state variables, $X_{\mathcal{M}}$; the set of parameters, $\Theta_{\mathcal{M}}$; the set of inputs (variables not computed by any constraint), $U_{\mathcal{M}}$; the set of outputs (variables not used to compute any other variables), $Y_{\mathcal{M}}$; and the set of auxiliary variables, $A_{\mathcal{M}}$. Parameters, $\Theta_{\mathcal{M}}$, include explicit model parameters that are used in the model constraints (e.g., fault parameters). Auxiliary variables, $A_{\mathcal{M}}$, are additional variables that are algebraically related to the state, parameter, and input variables, and are used to simplify the structure of the equations.

Example 4. In the circuit model, we have $X_{\mathcal{M}} = \{i_3, v_6, i_8, v_{11}\}$, $\Theta_{\mathcal{M}} = \{L_1, R_1, C_1, L_2, R_2, C_2\}$, $U_{\mathcal{M}} = \{u_v\}$, and $Y_{\mathcal{M}} = \{i_3^*, i_{11}^*, v_8^*\}$. Remaining variables belong to $A_{\mathcal{M}}$. Here, the * superscript is used to denote a measured value of a physical variable, e.g., $i_3 \in X_{\mathcal{M}}$ is the current and $i_3^* \in Y_{\mathcal{M}}$ is the measured current. Since i_3 is used to compute other variables, like i_2 , it cannot belong to $Y_{\mathcal{M}}$ and a separation of the variables is required. Connected components are known by shared variables, e.g., R_1 and Series Connection₁ are connected because they share i_5 and v_5 .

The model constraints, $C_{\mathcal{M}}$, are a union of the component constraints over all modes, i.e., $C_{\mathcal{M}} = C_{\delta_1} \cup C_{\delta_2} \cup \dots \cup C_{\delta_d}$, where $C_{\delta_i} = C_{\delta_i}^1 \cup C_{\delta_i}^2 \cup \dots \cup C_{\delta_i}^n$ for n modes. Constraints are exclusive to components, that is, a constraint $c \in C_{\mathcal{M}}$ belongs to exactly one C_δ for $\delta \in \mathcal{M}$.

To refer to a particular mode of a model we use the concept of a *mode vector*. A mode vector \mathbf{m} specifies the current mode of each of the components of a model. So, the constraints for a mode \mathbf{m} are denoted as $C_{\mathcal{M}}^{\mathbf{m}}$.

Example 5. Consider a model with five components, then if $\mathbf{m} = [1, 1, 3, 2, 1]$, it indicates that components δ_1 , δ_2 , and δ_5 use constraints of their mode 1, component δ_3 use constraints of its mode 3, and component δ_4 use constraints of its mode 2.

Table 1: Components of the electrical circuit.

Component	Mode	Constraints	$\mathcal{A}^{[1\ 2]}$	$\mathcal{A}_{i_3}^{[1\ 2]}$	$\mathcal{A}_{v_8^*}^{[1\ 2]}$	$\mathcal{A}_{i_{11}^*}^{[1\ 2]}$	$\mathcal{A}^{[2\ 1]}$	$\mathcal{A}_{i_3^*}^{[2\ 1]}$	$\mathcal{A}_{v_8^*}^{[2\ 1]}$	$\mathcal{A}_{i_{11}^*}^{[2\ 1]}$
δ_1 : V	1	$v_1 = u_v$	v_1	v_1			v_1	v_1	v_1	
δ_2 : Sw ₁	1	$i_1 = 0$ $i_2 = 0$	i_1 i_2	i_2	i_2	i_2				
	2	$i_1 = i_2$ $v_1 = v_2$					i_1 v_2	v_2	v_2	v_2
δ_3 : Parallel Connection ₁	1	$v_2 = v_3$ $v_2 = v_4$ $i_2 = i_3 + i_4$	v_3 v_2 i_4	v_3 v_2 i_4		i_4 i_4	v_3 v_4 i_2	v_3		v_4
		$i_3 = v_3 / L_1$ $i_3 = \int_{t_0}^t i_3$	i_3 i_3	i_3 i_3			i_3 i_3	i_3		
δ_5 : Series Connection ₁	1	$i_4 = i_5$ $i_4 = i_6$ $i_4 = i_7$ $v_4 = v_5 + v_6 + v_7$	i_5 i_6 i_7 v_4	i_5 i_6 i_7 v_4			i_5 i_6 i_4 v_7			i_5 i_6 i_4 v_7
		$v_5 = i_5 * R_1$	v_5	v_5			v_5			v_5
		$v_6 = i_6 / C_1$ $v_6 = \int_{t_0}^t v_6$	v_6 v_6	v_6 v_6			v_6			v_6
		$v_7 = v_8$ $v_7 = v_9$ $i_7 = i_8 + i_9$	v_7 v_7 i_9	v_7 v_7 i_9			v_7 v_7 i_9	v_9 i_7		
δ_9 : L ₂	1	$i_8 = v_8 / L_2$ $i_8 = \int_{t_0}^t i_8$	i_8 i_8	i_8 i_8		i_8 i_8	i_8 i_8			i_8 i_8
		$i_9 = 0$ $i_{10} = 0$ $i_{11} = 0$					i_9 i_{10} i_{11}			i_9 i_{11}
δ_{10} : Sw ₂	2	$i_9 = i_{10}$ $i_9 = i_{11}$ $v_9 = v_{10} + v_{11}$	i_{10} i_{11} v_9		i_{10}		i_{11}			
	1	$v_{10} = i_{10} * R_2$	v_{10}	v_{10}			v_{10}			
δ_{12} : C ₂	1	$v_{11} = i_{11} / C_1$ $v_{11} = \int_{t_0}^t v_{11}$	v_{11} v_{11}		v_{11}		v_{11}			
δ_{13} : Current Sensor ₁₁	1	$i_{11}^* = i_{11}$	i_{11}^*		i_{11}	i_{11}^*	i_{11}^*			i_{11}^*
δ_{14} : Voltage Sensor ₈	1	$v_8^* = v_8$	v_8^*	v_8	v_8^*	v_8	v_8^*			v_8^*
δ_{15} : Current Sensor ₃	1	$i_3^* = i_3$	i_3^*	i_3	i_3	i_3	i_3^*	i_3^*		

For shorthand, we will refer to the modes only of the components with multiple modes. So, for the circuit, we will refer only to components δ_2 and δ_{10} , and we will have four possible mode vectors, $[1\ 1]$, $[1\ 2]$, $[2\ 1]$, and $[2\ 2]$.

The switching behavior of each component can be defined using a finite state machine or a similar type of control specification. The state transitions may be attributed to controlled or autonomous events. However, for the purposes of this paper, we view the switching behavior as a black box where the mode change event is given, and refer the reader to many of the approaches already proposed in the literature for modeling the switching behavior [1, 8].

2.2 Causality

Given a constraint c , which belongs to a specific mode of a specific component, the notion of a *causal assignment* is used to specify a possible computational direction, or *causality*, for the constraint c . This is done by defining which $v \in V_c$ is the dependent variable in equation ε_c .

Definition 4 (Causal Assignment). A *causal assignment* α_c to a constraint $c = (\varepsilon_c, V_c)$ is a tuple $\alpha_c = (c, v_c^{out})$, where $v_c^{out} \in V_c$ is assigned as the dependent variable in ε_c . We use V_c^{in} to denote the independent variables in the constraint, where $V_c^{in} = V_c - \{v_c^{out}\}$.

In general, the set of possible causal assignments for a constraint c is as big as V_c , because each variable in V_c can

act as v_c^{out} . However, in some cases some causal assignments may not be possible, e.g., if we have noninvertible nonlinear constraints. Also, if we assume integral causality, then state variables must always be computed via integration, and so the derivative causality is not allowed. Further, when placed in the context of a model, additional causalities may not be applicable, because the causal assignments of other constraints may limit the potential causal assignments. To denote this concept, we use $\hat{\mathcal{A}}_c$ to refer to the set of permissible causal assignments of a constraint c .

For a given mode, we have the set of (specific) causal assignments over the entire model in its mode, denoted using \mathcal{A}^m . So, some $\alpha \in \mathcal{A}^m$ would refer to the causal assignment of some constraint in some component of the model in its correct mode. The consistency of the causal assignments \mathcal{A}^m is defined as follows,

Definition 5 (Consistent Causal Assignments). Given a mode m , we say that a set of causal assignments \mathcal{A}^m , for a model \mathcal{M} is *consistent* if (i) for all $v \in U_{\mathcal{M}} \cup \Theta_{\mathcal{M}}$, \mathcal{A}^m does not contain any α such that $\alpha = (c, v)$, i.e., input or parameter variables cannot be the dependent variables in the causal assignment; (ii) for all $v \in Y_{\mathcal{M}}$, \mathcal{A}^m does not contain any $\alpha = (c, v_c^{out})$ where $v \in V_c^{in}$, i.e., an output variable can only be used as the dependent variable; and (iii) for all $v \in V_{\mathcal{M}} - U_{\mathcal{M}} - \Theta_{\mathcal{M}}$, \mathcal{A}^m contains exactly one $\alpha = (c, v)$, i.e., every variable that is not an input or

parameter is computed by only one (causal) constraint.

With causality information, we can efficiently derive a set of submodels for residual generation [12].

3 Hybrid Systems Diagnosis Approach

We propose a hybrid systems diagnosis approach based on structural model decomposition. In this approach, we generate submodels for the purpose of computing residuals. Residuals can then be used for diagnosis.

For hybrid systems, however, the problem is that these submodels may change as the result of a mode change. That is, we may obtain two different submodels when decomposing the model in two different modes. There are two approaches to this problem. One is to find a set of submodels that work for all modes, and can be easily reconfigured by executing only local mode changes within the submodels [10]. This approach requires the least online effort, with some offline effort in finding these submodels, which exist only in limited cases. The other approach is to generate submodels for the current mode, and when a mode change occurs, reconfigure the submodels to be consistent with the new system mode. This is the approach we develop in this paper.

In order to execute this type of approach, however, we must be able to efficiently reconfigure submodels online. In order to do this, we take advantage of causality in two ways. First, we perform an offline model analysis to determine which causalities of the hybrid system model are not permissible, i.e., they will never be used in any mode of the system (determine $\mathbb{A}_{\mathcal{M}}$ for a model). Second, we use an efficient causality reassignment algorithm, so that the causality of a hybrid systems model is updated incrementally when a mode changes (given \mathcal{A} for the previous mode, compute it for the new mode). Since causal changes usually only propagate in a local area in the model, causality does not need to be reassigned at the global model level. Together, these algorithms reduce the number of potential causalities to search within the model decomposition algorithm and allow efficient submodel reconfiguration.

4 Causality Assignment

In order to compute minimal submodels for residual generation, we need a model \mathcal{M} with a valid causal assignment \mathcal{A}^m . As described in Section 2, causality assignment can only be defined for a given mode. However, there are some causal assignments that are independent of the system mode, i.e., they are valid for all system modes. We capture this through the notion of permissible causal assignments, introduced as $\mathbb{A}_{\mathcal{M}}$ in Section 2.

Given a model with a number of modes, some constraints will always have the same causal assignment in all modes, and we say these constraints are in *fixed causality*.

Definition 6 (Fixed Causality). A constraint c_δ is in *fixed causality* if (i) component δ has only a single mode, i.e., $|\mathcal{C}_\delta| = 1$, and (ii) for c_δ in the single $C \in \mathcal{C}_\delta$, it always has the same causal assignment in all system modes.

If a constraint is in fixed causality, then $|\mathbb{A}_c| = 1$, i.e., there is only one permissible causal assignment. For example, if we make the integral causality assumption, then constraints computing state variables will always be in the integral causality, and thus they are in fixed causality.

Additionally, when the constraint is viewed in the context of the model, the concept of fixed causality can be propagated

from one constraint to the related constraints (those sharing a variable with the fixed causality constraint). This will help to reduce the number of permissible causal assignments. For example, if we again assume integral causality, then any constraint involving a state variable cannot be in a causal assignment where the state variable is the dependent/output variable, because the integration constraint is the one that must compute it. For such a constraint, $1 < |\mathbb{A}_c| < |V_c|$.

Given a system model and a set of outputs, Algorithm 1 searches over the model constraints to reduce the set of permissible causal constraints based on system-level information.¹ First, it determines which constraints are mode-variant, i.e., they can appear/disappear from the model depending on the mode (so belong to components with multiple modes), and which are mode-invariant, i.e., they are present in all system modes (so belong to components with a single mode). It is only the mode-invariant constraints for which causal assignments can be removed. We then construct a queue of variables from which to propagate. This queue contains the inputs and parameters (which must always be independent/input variables in constraints), and the outputs (which must always be dependent/output variables in constraints). We create a variable set V that refers to the variables that are resolved, i.e., either they are inputs/parameters or there is a constraint with a single causal assignment that will compute the variable. So, V is initially set to include $U_{\mathcal{M}}$ and $\Theta_{\mathcal{M}}$. Further, for any mode-invariant constraints that only has a single causal assignment, the output variable is added to V , and all variables of the constraint added to the queue.

The main idea is to analyze the causality restrictions imposed by variables in the queue, which will be propagated throughout the model. While the queue is nonempty, we pop a variable v off the queue. We then count the number of constraints involving v that have no set causal assignment yet, including constraints that are both mode-variant and mode-invariant. We then go through all mode-invariant constraints involving v , and remove causal assignments that will never be possible. There are three conditions in which this holds: a causal assignment is not possible in any system mode if (i) the output variable is already computed by another constraint, or is an input/parameter (i.e., in V), (ii) any of the input variables are in the model outputs (i.e., in Y), or (iii) v is not yet computed by any constraint (i.e., not in V), there is only one noncausal constraint involving v remaining, and v is not the output in this causality (in this case, v needs to be computed by some constraint and there is only one option left, so this constraint must only be in the causality computing v). These causal assignments are removed. If only one is left, then we add the output for that causal assignment to V , and add the constraint's variables to the queue. The algorithm stops when causalities can no longer be removed, i.e., there are not enough restrictions imposed by the current permissible causalities to reduce $\mathbb{A}_{\mathcal{M}}$ further.

Example 6. For the circuit, we assume integral causality, so all constraints with the state variables are limited to causal assignments in which the states are computed via integration. Further, the constraint with u_V is also fixed so that u_V is the independent variable. For any specified outputs, $\mathbb{A}_{\mathcal{M}}$ is also

¹For structural model decomposition, some output variables may become input variables and so the causal assignments permitting that must be retained. Therefore, the algorithm only reduces the permissible set of causal assignments for a given set of outputs $Y \subseteq Y_{\mathcal{M}}$.

Algorithm 1 $\mathbb{A}_M \leftarrow \text{ReduceCausality}(\mathcal{M}, \mathbb{A}_M, Y)$

```

1:  $C_{\text{invariant}} \leftarrow \emptyset$ 
2:  $C_{\text{variant}} \leftarrow \emptyset$ 
3: for all  $\delta \in \mathcal{M}$  do
4:   if  $|C_\delta| = 1$  then
5:      $C_{\text{invariant}} \leftarrow C_{\text{invariant}} \cup C_\delta^1$ 
6:   else
7:      $C_{\text{variant}} \leftarrow C_{\text{variant}} \cup \left( \bigcup_{C \in C_\delta} C \right)$ 
8:  $Q \leftarrow U_M \cup \Theta_M \cup Y$ 
9:  $V \leftarrow U_M \cup \Theta_M$ 
10: for all  $c \in C_{\text{invariant}}$  do
11:   if  $|\mathbb{A}_c| = 1$  then
12:      $(c, v) \leftarrow \mathbb{A}_c(1)$ 
13:      $Q \leftarrow Q \cup V_c$ 
14:      $V \leftarrow V \cup v$ 
15: while  $|Q| > 0$  do
16:    $v \leftarrow \text{pop}(Q)$ 
17:    $n_{\text{noncausal}} \leftarrow 0$ 
18:   for all  $c \in C_{\text{invariant}}(v)$  do
19:     if  $|\mathbb{A}_c| > 1$  or  $(|\mathbb{A}_c| = 1 \text{ and } v_{\mathbb{A}_c(1)} \notin V)$  then
20:        $n_{\text{noncausal}} \leftarrow n_{\text{noncausal}} + 1$ 
21:   for all  $c \in C_{\text{variant}}(v)$  do
22:      $n_{\text{noncausal}} \leftarrow n_{\text{noncausal}} + 1$ 
23:   for all  $c \in C_{\text{invariant}}(v)$  do
24:     if  $|\mathbb{A}_c| > 1$  or  $(|\mathbb{A}_c| = 1 \text{ and } v_{\mathbb{A}_c(1)} \notin V)$  then
25:       for all  $(c', v') \in \mathbb{A}_c$  do
26:         if  $v' \in V$  then
27:            $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$ 
28:         if  $(V_c - \{v\}) \cap Y' \neq \emptyset$  then
29:            $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$ 
30:         if  $n_{\text{noncausal}} = 1$  and  $v' \notin V$  and  $v' \neq v$  then
31:            $\mathbb{A}_c \leftarrow \mathbb{A}_c - (c', v')$ 
32:         if  $|\mathbb{A}_c| = 1$  then
33:            $(c', v') \leftarrow \mathbb{A}_c(1)$ 
34:            $Q \leftarrow Q \cup (V_{c'} - V)$ 
35:            $V \leftarrow V \cup \{v'\}$ 

```

reduced so that they can appear only as dependent variables.

With \mathbb{A}_M defined, we can perform causality assignment for a given mode, \mathbf{m} . Because \mathbb{A}_M was reduced as much as possible, causality assignment (and, later, reassignment) will be more efficient than otherwise. Algorithm 2 describes the causality assignment process for a model given a mode. Here, the model is assumed to not have an initial causal assignment. Causal assignment works by propagating causal restrictions throughout the model. The process starts at inputs, which must always be independent variables in constraints; outputs, which must always be the dependent variables in constraints; and variables for involved in fixed causality constraints. From these variables, we should be able to propagate throughout the model and compute a valid causal assignment for the model in the given mode. For the purposes of this paper, we assume integral causality and that the model possesses no algebraic loops.² In this case, there is only one valid causal assignment (this is a familiar concept within bond graphs) [13].

Specifically, the algorithm works as follows. Similar to Algorithm 1, we keep a queue of variables to propagate causality restrictions, Q , and a set of variables that are computed in the current causality, V . Initially, V is set to U and Θ , because these variables are not to be computed by any constraint. Q is set to U , Θ , and Y , since the causality of

²If algebraic loops exist, the algorithm will terminate before all constraints have been assigned a causality. Extending the algorithm to handle algebraic loops is similar to that for bond graphs; a constraint without a causality assignment is assigned one arbitrarily, and then effects of this assignment are propagated until nothing more is forced. This process repeats until all constraints have been assigned causality.

Algorithm 2 $\mathcal{A} \leftarrow \text{AssignCausality}(\mathcal{M}, \mathbf{m}, \mathbb{A})$

```

1:  $\mathcal{A} \leftarrow \emptyset$ 
2:  $V \leftarrow U_M \cup \Theta_M$ 
3:  $Q \leftarrow U_M \cup \Theta_M \cup Y_M$ 
4: for all  $c \in C_{M^{\mathbf{m}}}$  do
5:   if  $|\mathbb{A}_c| = 1$  then
6:      $(c, v) \leftarrow \mathbb{A}_c(1)$ 
7:      $Q \leftarrow Q \cup v$ 
8:   while  $|Q| > 0$  do
9:      $v \leftarrow \text{pop}(Q)$ 
10:    for all  $c \in C_{M^{\mathbf{m}}}(v)$  do
11:      if  $c \notin \{c : (c, v) \in \mathcal{A}\}$  then
12:         $\alpha^* \leftarrow \emptyset$ 
13:        for all  $\alpha \in \mathbb{A}_c$  do
14:          if  $V_c - \{v_{\alpha^*}\} \cup V \neq \emptyset$  then
15:             $\alpha^* \leftarrow \alpha$ 
16:          else if  $\alpha_v \in Y$  then
17:             $\alpha^* \leftarrow \alpha$ 
18:          else if  $v_{\alpha^*} = v$  and  $|C_{M^{\mathbf{m}}}(v)| - |\{c' : (c', v') \in \mathcal{A} \wedge v \in v_{c'}\}| = 1$  then
19:             $\alpha^* \leftarrow \alpha$ 
20:          if  $\alpha^* \neq \emptyset$  then
21:             $\mathcal{A} \leftarrow \mathcal{A} \cup \{\alpha^*\}$ 
22:             $Q \leftarrow Q \cup (V_c - V)$ 
23:             $V \leftarrow V \cup \{v_{\alpha^*}\}$ 

```

constraints is restricted to U and Θ variables being independent variables and Y variables being dependent variables. We add also to Q any variables involved in constraints that have only one permissible causal assignment, because this will also restrict other causal assignments. The set of causal assignments is maintained in \mathcal{A} .

The algorithm goes through the queue, inspecting variables. For a given variable, we obtain all constraints it is involved in, and for each one that does not yet have a causal assignment (in \mathcal{A}), we go through all permissible causal assignments, and determine if the causality is forced into one particular causal assignment, α^* . If so, we assign that causality and propagate by adding the involved variables to the queue. A causal assignment $\alpha = (c, v)$ is forced in one of three cases: (i) v is in Y , (ii) all variables other than v of the constraint are already in V , and (iii) v is not yet in V , and all but one of the constraints involving v have an assigned causality, in which case no constraint is computing v and there is only one remaining constraint that must compute v .

Example 7. Consider the mode $\mathbf{m} = [1 \ 2]$. Here, $\mathcal{A}^{[1 \ 2]}$ is given in column 4 of Table 1, denoted by the v_c^{out} in the causal assignment. In this mode, the first switch is off, so i_1 and i_2 act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column.

Example 8. Consider the mode $\mathbf{m} = [2 \ 1]$. Here, $\mathcal{A}^{[2 \ 1]}$ is given in column 8 of Table 1. In this mode, the second switch is off, so i_9 , i_{10} , and i_{11} act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column. Note that some causal assignments are in the same as in $\mathbf{m} = [1 \ 2]$, while others are different. In changing from one mode to another, an efficient causality reassignment should be able to determine which constraints need to change causality, and do the work for only that portion of the model.³ Causal assignments that do not change from mode to mode are in fixed causality and found by Algorithm 1.

³Note that this particular circuit was carefully chosen so that causality does propagate across much of the circuit, in order to demonstrate the causality reassignment algorithm.

5 Structural Model Decomposition

For a given causal model in a given mode, we have the equivalent of a continuous systems model for the purpose of structural model decomposition, and we can compute minimal submodels using the `GenerateSubmodel` algorithm described in our previous work [12]. The algorithm finds a submodel, which computes a set of local outputs given a set of local inputs, by searching over the causal model. It starts at the local inputs, and propagates backwards through the causal constraints, finding which constraints and variables must be included in the submodel. When possible, causal constraints are inverted in order to take advantage of local inputs. Additional information and the pseudocode are provided in [12].

In the context of residual generation, we set the local output set to a single measured value, and the local inputs to all other measured values and the (known) system inputs. That is, we exploit the analytical redundancy provided by the sensors in order to find minimal submodels to compute estimated values of sensor outputs. In this framework, we consider one submodel per sensor, each producing estimated values for that sensor.

Assuming that the set of sensors does not change from mode to mode, then for a hybrid system we have one submodel for each sensor.⁴ However, since the set of constraints changes from mode to mode, the result of the `GenerateSubmodel` algorithm will also change. When a mode changes, we first reassign causality to the model for the new mode. Then, we generate new updated submodels for that mode using `GenerateSubmodel`. In order to reduce the work performed by this algorithm when a mode changes, we use an efficient causality reassignment algorithm. That, coupled with the reduced set $\mathbb{A}_{\mathcal{M}}$, significantly reduces the work of the algorithm compared to a naive approach, where the submodels are completely regenerated for a new mode. Additionally, when the system transitions to a new mode, the causal assignments for the previous mode can be stored, so that when the system changes to a mode that has already been visited, it just takes the causal assignments that were stored previously. Similarly, submodels generated in previously visited modes can be saved and reused when the mode reappears.

Example 9. The causal assignments for the submodels in the different modes are shown in Table 1. For example, consider the submodel for i_{11}^* in $\mathbf{m} = [2 \ 1]$. Here, i_{11} is zero, since Sw_2 is off, and therefore we have just two constraints needed to compute i_{11}^* . In mode $\mathbf{m} = [1 \ 2]$, i_{11}^* can be computed using 16 constraints, where v_8^* is used as a local input to the submodel.

Note that a submodel for an output may have different states in two different modes (e.g., in moving from $\mathbf{m} = [2 \ 1]$ to $\mathbf{m} = [1 \ 2]$, the i_{11}^* submodel adds state v_6). In order to continue tracking, new states must be initialized. For the purposes of this paper, we assume that in any one system mode, all states are included in at least one submodel.⁵ Therefore,

⁴By assuming that the sensor set does not change, we mean only that sensors are not added/removed to/from the physical system upon a mode change. They are still allowed to be connected/disconnected, but still appear in the system model even when disconnected. For example, if a disconnected sensor outputs 0, then that needs to still be in the model.

⁵If this is not the case, then a state is not observable in some

Algorithm 3 $\mathcal{A}^{\mathbf{m}'}$ ←
ReassignCausality($\mathcal{M}, \mathbf{m}, \mathcal{A}^{\mathbf{m}}, \mathbb{A}$)

```

1:  $\mathcal{A}^{\mathbf{m}'} \leftarrow \emptyset$ 
2: for all  $(c, v) \in \mathcal{A}^{\mathbf{m}}$  do
3:   if  $c \in C_{\mathcal{M}^{\mathbf{m}}}$  then
4:      $\mathcal{A}^{\mathbf{m}'} \leftarrow \mathcal{A}^{\mathbf{m}'} \cup \mathcal{A}^{\mathbf{m}}$ 
5:  $V \leftarrow \emptyset$ 
6:  $Q \leftarrow \emptyset$ 
7: for all  $\delta \in \mathcal{M}$  where  $m_\delta \neq m'_\delta$  do
8:    $Q \leftarrow Q \cup V_\delta$ 
9:   while  $|Q| > 0$  do
10:     $v \leftarrow \text{pop}(Q)$ 
11:    for all  $c \in C_{\mathcal{M}^{\mathbf{m}}}(v)$  do
12:      if  $c \notin \{c : (c, v) \in \mathcal{A}^{\mathbf{m}'}\}$  then
13:         $\alpha^* \leftarrow \emptyset$ 
14:        for all  $\alpha \in \mathbb{A}_c$  do
15:          if  $V_c - \{v_{\alpha^*}\} \cup V \neq \emptyset$  then
16:             $\alpha^* \leftarrow \alpha$ 
17:          else if  $\alpha_v \in Y$  then
18:             $\alpha^* \leftarrow \alpha$ 
19:          else if  $v_{\alpha^*} = v$  and  $|C_{\mathcal{M}^{\mathbf{m}}}(v)| - |\{c' : (c', v') \in \mathcal{A} \wedge v \in v_{c'}\}| = 1$  then
20:             $\alpha^* \leftarrow \alpha$ 
21:          if  $\alpha^* \neq \emptyset$  then
22:            if  $\exists \alpha \in \mathcal{A}^{\mathbf{m}'}$  where  $v_\alpha = v_{\alpha^*}$  then
23:               $\mathcal{A}^{\mathbf{m}'} \leftarrow \mathcal{A}^{\mathbf{m}'} - \{\alpha^*\}$ 
24:               $Q \leftarrow Q \cup (V_{\{\alpha^*\}} - V)$ 
25:             $\mathcal{A}^{\mathbf{m}'} \leftarrow \mathcal{A}^{\mathbf{m}'} \cup \{\alpha^*\}$ 
26:             $Q \leftarrow Q \cup (V_c - V)$ 
27:             $V \leftarrow V \cup \{v_{\alpha^*}\}$ 
28:          else if  $(V_c - V) = v$  then
29:             $V \leftarrow V \cup \{v\}$ 
30:             $Q \leftarrow Q \cup \{v\}$ 

```

a submodel that gets a state added in a new mode can initialize using the estimated value from another submodel in the previous mode.

6 Online Causality Reassignment

As we mentioned before, from the initial mode in the system with a valid set of causal assignments, we compute minimal submodels. However, when the system transitions to a different mode, any submodel containing constraints of a switching component will no longer be consistent, and must be recomputed. In order to do this, we need to know the causal assignments for the new mode. We can reassign causality in an efficient incremental process to avoid having to reassign causality to the whole model, as causal changes typically propagate only to a small local area in the model [11].

Algorithm 3 presents the causality reassignment procedure. The main ideas are based on the hybrid sequential causality assignment procedure (HSCAP) developed for hybrid bond graphs in [11]. In our more general modeling framework, we find that similar ideas apply. Essentially, we start with a causal model in a given mode. We then switch to a different mode, so for the switching components we have a new set of constraints in the model. We need to find causal assignments for these constraints. It is likely that some of the necessary causal assignments will conflict with causal assignments from the old mode, therefore, we have to resolve the conflict and propagate the change. The change will propagate only as far as it needs to in order to obtain a valid causal assignment for the model in the new mode. Here, propagation stops along a computational path when a new causal assignment does not conflict with a previous assignment.

mode. Estimation techniques to handle that situation are outside the scope of this paper.

The algorithm works similarly to Algorithm 2. It maintains a queue of variables to inspect and a set V of variables that are known to be computed in the causality for the new mode (so includes only variables from new assignments or confirmed assignments made in the new mode). In this case, we initialize the queue only to variables involved in the constraints of the switching components. If no components are switching, the queue will be empty and no work will be done. The main idea is that the required causal changes from the variables placed in the queue will, on average, be limited to a very small area. The causal assignments for the new mode are initialized to those for the previous mode, for any constraints that still exist in the new mode. Some of these may conflict with the new mode and will be removed and replaced with different assignments.

As in all the other causality algorithms, we go through the queue and propagate the restrictions we find on causality. We pop a variable off the queue, and look at all involved constraints. If the constraint is not causal, then we need to assign causality. We do the same analysis as before to find if a causality is forced, but checking things only with respect to V that includes only variables with confirmed causal assignments computing it in the new mode. If we find a constraint that is forced into a particular causal assignment for the new mode, we make the assignment. If it conflicts with one already in the set of causal assignments (copied from the old mode), then we remove the old assignment and add the new one, adding the involved variables to the queue so that changes are propagated.

7 Demonstration of Approach

For the circuit example, we consider two modes: one where Sw_1 is on and Sw_2 is off (i.e., $\mathbf{m} = [2 \ 1]$), and one where Sw_1 is off and Sw_2 is on (i.e., $\mathbf{m} = [1 \ 2]$). We consider a scenario in which to demonstrate the approach where the system starts in $\mathbf{m} = [2 \ 1]$, switches to $\mathbf{m} = [1 \ 2]$ at $t = 10$ s, and switches back to $\mathbf{m} = [2 \ 1]$ at $t = 20$ s. Additionally, at $t = 15$ s, a fault is injected, specifically, an increase in R_1 .

Fig. 2 shows the measured and submodel-estimated values for the sensors. Up through the first mode change, the outputs are correctly tracked by the submodels. At the first mode change at 10 s, the submodels reconfigure and track correctly up to 15 s, when the fault is injected, and a discrepancy is observed in the i_3^* submodel. Specifically, the current increases above what is expected. The other submodels in this mode are independent of the fault, and so continue to track correctly. When the second mode change occurs, i_{11}^* can still be tracked correctly, since its estimation remains independent of the fault. However, we now see a discrepancy in v_8^* , as the measurement increases above what is expected. This transient occurs because we switch from a mode in which the submodel is independent of the fault to one where it is dependent on the fault. Fault isolation can be performed by using the information that in $\mathbf{m} = [1 \ 2]$, an increase in R_1 would produce an increase in the residual for i_3^* , and in $\mathbf{m} = [2 \ 1]$, it would produce an increase also in the v_8^* residual.

8 Related Work

Modeling and diagnosis for hybrid systems have been an important focus of study for researchers from both the FDI and DX communities during the last 15 years. In the FDI community, several hybrid system diagnosis approaches have been

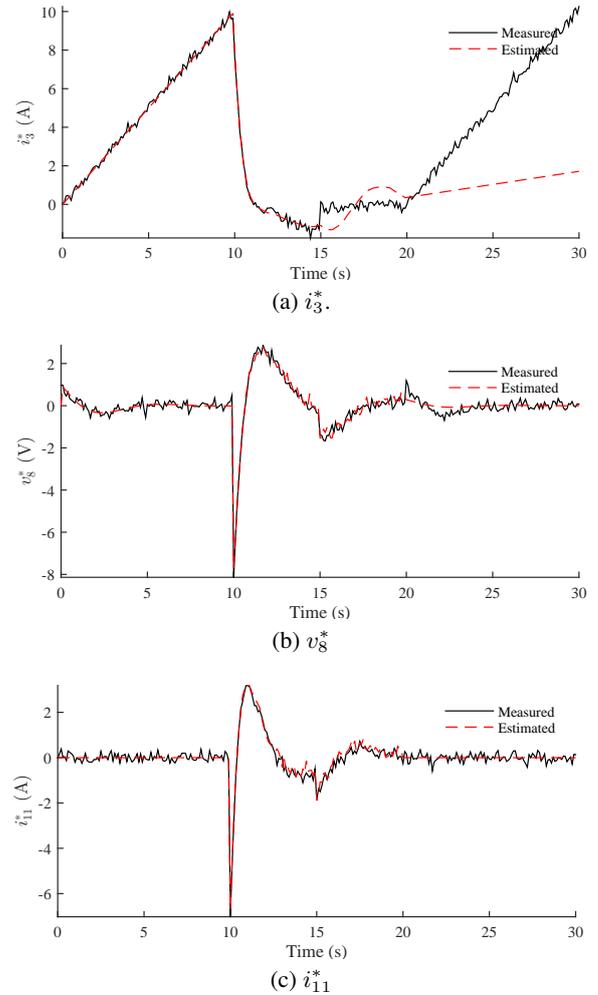


Figure 2: Measured and estimated values with an increase in R_1 at $t = 15$ s.

developed. In [14], parameterized ARRr are used. However, the approach is not suitable for systems with high nonlinearities or a large set of modes. A different approach [15], but uses purely discrete models.

In the DX community, some approaches have used different kind of automata to model the complete set of modes and transitions between them. In those cases, the main research topic has been hybrid system state estimation, which has been done using probabilistic (e.g., some kind of filter [16] or hybrid automata [4]) or set-theoretic approaches [5].

Another solution has been to use an automaton to track the system mode, and then use a different technique to diagnose the continuous behavior (for example, using a set of ARRr for each mode [3], or parameterized ARRr for the complete set of modes [17]). Nevertheless, one of the main difficulties regarding state estimation using these techniques is the need to pre-enumerate the set of possible system-level modes and mode transitions, which is difficult for complex systems. We avoid this problem by using a compositional approach.

Regarding hybrid systems modeling, there are several proposals. For HBGs [8, 18], there are two main approaches: those that use switching elements with fixed causality [18–20], and those who use ideal switching elements that change causality [8]. The advantages of the latter are that the modeling of hybrid systems is done through a special kind of

hybrid component (which avoid the mode pre-enumeration in the system), and also changes are handled in a very efficient way [11]. Finally, in [10] the HBGs are used to compute minimal submodels (Hybrid Possible Conflicts, HPCs) similar to the minimal submodels presented in this paper. HPCs can track hybrid systems behavior, efficiently changing on-line for each mode the PC simulation model, by using block diagrams as in [11], and performing diagnosis without pre-enumerating the set of modes in the system. However, HPCs rely on HBG modeling and do not provide a generalized framework for hybrid systems.

9 Conclusions

In this work, we have developed a compositional modeling framework for hybrid systems. Using computational causality, we developed efficient causality assignment algorithms. Given this causal information, submodels computed using structural model decomposition can be computed and reconfigured efficiently. The approach was demonstrated with a circuit system. In future work, we will further develop the hybrid systems diagnosis approach for the single and multiple fault cases, and we will approach the diagnosis task in a distributed manner. The assumption of one submodel per sensor can also be dropped, using the extended framework developed in [21, 22].

Acknowledgments

This work has been funded by the Spanish MINECO DPI2013-45414-R grant and the NASA SMART-NAS project in the Airspace Operations and Safety Program of the Aeronautics Mission Directorate.

References

- [1] T. A. Henzinger. *The theory of hybrid automata*. Springer, 2000.
- [2] T. Rienmüller, M. Bayouh, M.W. Hofbaur, and L. Travé-Massuyès. Hybrid Estimation through Synergic Mode-Set Focusing. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1480–1485, Barcelona, Spain, 2009.
- [3] M. Bayouh, L. Travé-Massuyès, and X. Olive. Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. In *18th European Conf. on Artificial Intel.*, pages 219–223, 2008.
- [4] M.W. Hofbaur and B.C. Williams. Hybrid estimation of complex systems. *IEEE Trans. on Sys., Man, and Cyber, Part B: Cyber.*, 34(5):2178–2191, 2004.
- [5] E. Benazera and L. Travé-Massuyès. Set-theoretic estimation of hybrid system configurations. *Trans. Sys. Man Cyber. Part B*, 39:1277–1291, October 2009.
- [6] S. Narasimhan and L. Brownston. HyDE: A General Framework for Stochastic and Hybrid Model-based Diagnosis. In *Proc. of the 18th Int. WS. on Principles of Diagnosis*, pages 186–193, May 2007.
- [7] L. Trave-Massuyes and R. Pons. Causal ordering for multiple mode systems. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning*, pages 203–214, 1997.
- [8] P.J. Mosterman and G. Biswas. A comprehensive methodology for building hybrid models of physical systems. *Artificial Intel.*, 121(1-2):171 – 209, 2000.
- [9] S. Narasimhan and G. Biswas. Model-Based Diagnosis of Hybrid Systems. *IEEE Trans. Syst. Man. Cy. Part A*, 37(3):348–361, May 2007.
- [10] A. Bregon, C. Alonso, G. Biswas, B. Pulido, and N. Moya. Hybrid systems fault diagnosis with possible conflicts. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, pages 195–202, Murnau, Germany, October 2011.
- [11] I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Efficient simulation of hybrid systems: A hybrid bond graph approach. *SIMULATION: Transactions of the Society for Modeling and Simulation International*, 87(6):467–498, June 2011.
- [12] I. Roychoudhury, M. Daigle, A. Bregon, and B. Pulido. A structural model decomposition framework for systems health management. In *Proceedings of the 2013 IEEE Aerospace Conference*, March 2013.
- [13] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg. *Systems Dynamics: Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons, Inc., NY, 2000.
- [14] V. Cocquempot, T. El Mezyani, and M. Staroswiecki. Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference*, volume 2, pages 1204–1212, July 2004.
- [15] J. Lunze. Diagnosis of quantised systems by means of timed discrete-event representations. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control, HSCC '00*, pages 258–271, London, UK, 2000. Springer-Verlag.
- [16] X. Koutsoukos, J. Kurien, and F. Zhao. Estimation of distributed hybrid systems using particle filtering methods. In *In Hybrid Systems: Computation and Control (HSCC 2003)*. Springer Verlag Lecture Notes on Computer Science, pages 298–313. Springer, 2003.
- [17] M. Bayouh, L. Travé-Massuyès, and X. Olive. Diagnosis of a Class of Non Linear Hybrid Systems by On-line Instantiation of Parameterized Analytical Redundancy Relations. In *20th International Workshop on Principles of Diagnosis*, pages 283–289, 2009.
- [18] W. Borutzky. Representing discontinuities by means of sinks of fixed causality. In F.E. Cellier and J.J. Granda, editors, *Proc. of the Int. Conf. on Bond Graph Modeling*, pages 65–72, 1995.
- [19] M. Delgado and H. Sira-Ramirez. Modeling and simulation of switch regulated dc-to-dc power converters of the boost type. In *IEEE Int. Conf. on Devices, Circuits and Systems*, pages 84–88, December 1995.
- [20] P.J. Gawthrop. Hybrid Bond Graphs Using Switched I and C Components. CSC report 97005, Centre for Sys. and Control, Faculty of Eng., Glasgow, U.K., 1997.
- [21] A. Bregon, M. Daigle, and I. Roychoudhury. An integrated framework for distributed diagnosis of process and sensor faults. In *2015 IEEE Aerospace Conf.*, 2015.
- [22] M. Daigle, I. Roychoudhury, and A. Bregon. Diagnosability-based sensor placement through structural model decomposition. In *Second Euro. Conf. of the PHM Society 2014*, pages 33–46, 2014.

Device Health Estimation by Combining Contextual Control Information with Sensor Data

Tomonori Honda and **Linxia Liao** and **Hoda Eldardiry** and **Bhaskar Saha** and **Rui Abreu**

Palo Alto Research Center, Palo Alto, California, USA

e-mail: {tomo.honda, linxia.liao, hoda.eldardiry, bhaskar.saha, rui.maranhao}@parc.com

Radu Pavel and **Jonathan C. Iverson**

TechSolve, Inc., Cincinnati, Ohio, USA

e-mail: {pavel, iverson}@TechSolve.org

Abstract

The goal of this work is to bridge the gap between business decision making and real-time factory data. Beyond real-time data collection, we aim to provide analysis capability to obtain insights from the data and converting the learnings into actionable recommendations. We focus on analyzing device health conditions and propose a data fusion method that combines sensor data with limited diagnostic signals with the device's operating context. We propose a segmentation algorithm that provides a temporal representation of the device's operation context, which is combined with sensor data to facilitate device health estimation. Sensor data is decomposed into features by time-domain and frequency-domain analysis. Principal component analysis (PCA) is used to project the high-dimensional feature space into a low-dimensional space followed by a linear discriminant analysis (LDA) to search the optimal separation among different device health conditions. Our industrial experimental results show that by combining device operating context with sensor data, our proposed segmentation and PCA-LDA approach can accurately identify various device imbalance conditions even for limited sensor data which could not be used to diagnose imbalance on its own.

1 Introduction

The growing Internet of Things is predicted to connect 30 billion devices by 2020 [1]. This will bring in tremendous amounts of data and drive the innovations needed to realize the vision of Industry 4.0—cyber-physical systems monitoring physical processes, and communicating and cooperating with each other and with humans in real time. One of the key challenges to be addressed is how to analyze large amounts of data to provide useful and actionable information for businesses intelligence and decision making. In particular, to prevent unexpected downtime and its significant impact on overall equipment effectiveness (OEE) and total cost of ownership (TCO) in many industries. Continuous monitoring of equipment and early detection of incipient faults can support optimal maintenance strategies, prevent downtime, increase productivity, and reduce costs.

A significant number of anomaly detection and diagnosis methods have been proposed for machine fault detection and machine health condition estimation. Chandola et al. [2]

discusses various categories of anomaly detection technologies and their assumptions as well as their computational complexity. Several approaches such as statistical methods [3], neural network methods [4] and reliability methods [5], have been applied to detect anomalies for various types of equipment. The philosophies and techniques of monitoring and predicting machine health with the goal of improving reliability and reducing unscheduled downtime of rotary machines are presented by Lee et al. [6].

Many of these methods focus on analyzing, combining, and modeling sensor data (e.g. vibration, current, acoustics signal) to detect machine faults. One issue that remains mostly unaddressed in these methods is that they rarely consider the varying operating context of the machine. In many cases, false alarms are generated due to a change in machine operation (e.g. rotational speed) rather than a change in machine condition. A major challenge in addressing this issue is that most machine controllers are built with proprietary communication protocols, which leads to a barrier in obtaining control parameters to understand the context under which the machine is operating. Recently, the MTConnect open protocol [7] was developed to connect various legacy machines independent of the controller providers. MTConnect provides an unprecedented opportunity to monitor machine operating context in real-time. In this paper, we leverage MTConnect to diagnose machine health condition by combining sensor data with operating context information. Additionally, we investigate whether it is possible to diagnose machine health condition using less sensor data when it is combined with context information.

Prior work [8] has demonstrated that vibration data could be used for diagnosing machine imbalance fault conditions. Our study focuses on extending prior work by exploring various types of sensor and control data for diagnosing the imbalance of the machine tools.

Our contribution includes the following extensions:

- Combining control and sensor signals to improve accuracy.
- Utilizing a different set of sensor data such as temperature, power, flow, and lubricant/coolant pH.

Our hypothesis is that these advancements to prior work will aid in improving the diagnosis capability as well as reducing the cost of machine diagnostics by utilizing cheaper sensors.

2 Experimental Data

The data under study has been collected from experiments utilizing a machine tool monitoring system implemented on a horizontal machining center manufactured by Milltronic with *Fanuc Oi-MC* control. We have two main sources of data: (i) data from additional sensors installed on the machine, and (ii) data from the machine tool controller. This data has been collected using National Instrument equipment and software (LabVIEW).

The external sensors used for data collection include:

- power sensor that measures power using Hall effect,
- accelerometers that capture machine tool motion in 6 degrees of freedom,
- thermocouples that measure temperatures at 10 locations on the machine tool,
- pH sensor for detecting the pH level of the metalworking fluid, and
- flow rate sensor to measure metalworking fluid pump flow.

The second category consists of data collected from the controller. This data includes drive loads, absolute and relative positions, servo delays, and feed rate. The complete list of the components of the control data is listed in Pavel et al.[8].

Data has been collected in two sessions, one in 2009 and the other in 2010. Although the basic control signals are similar, they are offset by constant values (see Figure 1). Since the positional offset could cause a difference in the motion dynamics, we have treated them as separate data sets for this study.

3 Technical Approach

For each extension to prior work listed in Section 1, we have performed two main steps for creating appropriate diagnostics:

- Feature Extraction & Synthesis
- Model Selection

3.1 Feature Extraction & Synthesis

There are various approaches for condensing time series information into data mining features. Prior work has utilized transfer functions to map control signals to vibrational sensor data [8]. The diagnosis step is then reduced to comparing the features of transfer function-predicted vibration data and the sensor-derived vibration data. This approach makes sense when the control signal directly impacts the output variables of the machine. For motion control of machine tools, the estimated transfer function should be similar to the transfer function of the implemented control (like PI or PID). Typical vibration data features would include average, standard deviation, and maximum FFT values [9].

However, we would like to diagnose the state of machine using not only accelerometers, but also other sensors, such as temperature sensors. Since temperatures at various locations are not part of active control loops, there may not exist well defined transfer functions that can map control signals to temperature sensor data very accurately. In such cases where conventional features extracted from temperature signals are not correlated with the fault (imbalance) to a

sufficient degree. Additionally, if the associated sensors are too expensive to install, then data fusion may be applied.

There are three data fusion approaches typically used in machinery diagnostics [10; 11]—data-level fusion, feature-level fusion, and decision-level fusion. Data-level fusion involves combining sensor data before feature extraction, such that features contain information gathered from multiple sensors. Feature-level fusion involves generating features from each sensor separately, then fusing this set of features generated from all of the sensors coherently for diagnostics. Finally, decision-level fusion creates diagnostics from each sensor separately, then aggregates these diagnostics into a single diagnostic output.

The choice of the three types of data fusion methods is often application specific. In our application, we found that temperature sensor data cannot resolve imbalance conditions by itself and control signal data is too coarse-grained to aid in classifying imbalance conditions using the standard data-fusion techniques. Note that we did not focus on spindle acceleration data, which could diagnose imbalance on its own (see Subsection 4.1) since that would require retrofitting existing machine tools with new expensive sensors and data acquisition hardware. Ideally we would like to use the readily accessible control signals and data from inexpensive temperature sensors to diagnose imbalance. To achieve this goal, we proposed a different type of data fusion approach. We used the control signal to provide the contextual information for temperature sensor data. The control signal is used for the segmentation of sensor data, but does not directly map into feature vectors (see Subsection 4.2).

3.2 Model Selection

Since the data sets are statistically small and dimensionality of the data is increased by feature synthesis, the models to be used for imbalance classification need to be carefully chosen to avoid over-fitting. The high-dimensional data needs to be projected to a much smaller sub-space to prevent over-fitting¹. To accomplish this, the main techniques used in this study are Principal Component Analysis (PCA) [12] and Linear Discriminant Analysis (LDA) [13]. These techniques are based on linear coordinate transformation, which makes them more likely to under-fit and less likely to over-fit [14].

4 Results

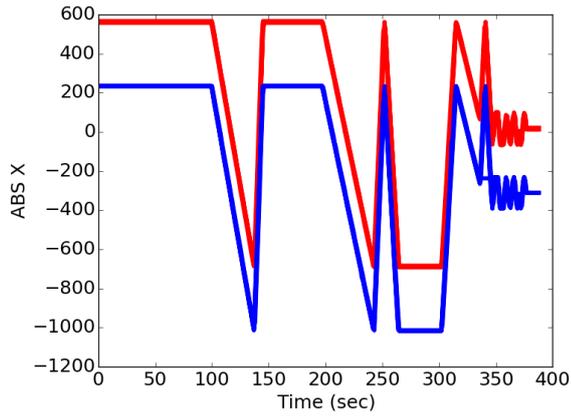
We have explored three types of imbalance diagnostics to investigate the hypothesis posed in Section 1:

- Sensor based Diagnostics
- Control based Temporal Segmentation followed by Sensor based Diagnostics

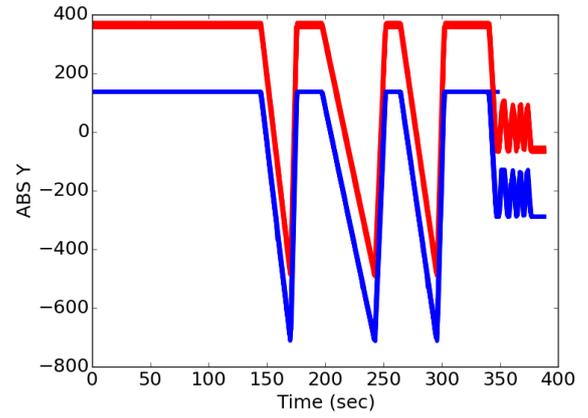
4.1 Sensor based Diagnostics

In this case, each sensor signal was analyzed separately to determine if any of the sensor signals contains enough diagnostic information to detect imbalance on its own. By plotting the time series data we find that spindle acceleration sensors (which captures vibration) show higher oscillation

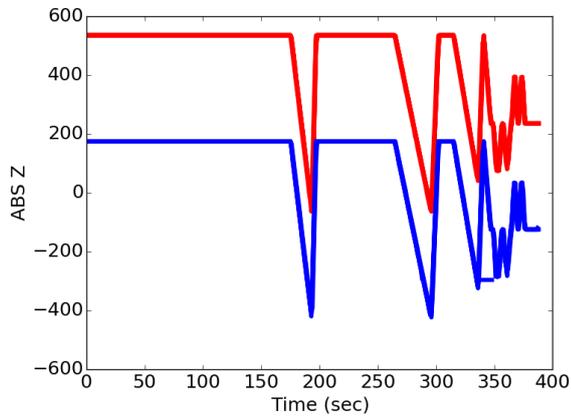
¹Note that complexity of model is positively correlated with likelihood of over-fitting. Thus, creating a classifier that takes high-dimensional input will have higher degree of freedom (i.e. higher complexity) compare to low-dimensional inputs, which results in higher likelihood of over-fitting.



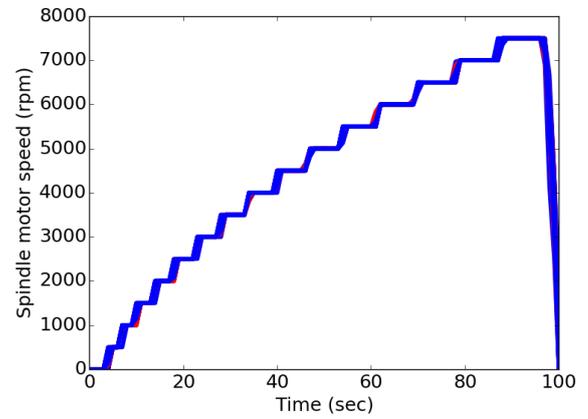
(a) Absolute X position



(b) Absolute Y position



(c) Absolute Z position



(d) Spindle Motor Speed

Figure 1: Primary Control Signals

amplitudes (see Figure 2) with increasing imbalance. Since imbalance actually impacts moment of inertia of the spindle, this change in acceleration is expected.

We also considered measuring imbalance through temperature. From the energy flow perspective, additional acceleration caused by imbalance should result in higher energy consumption from the power source and higher energy dissipation to thermal inertias due to friction, which should result in temperature increase in parts of the machine tool. However, the time series data, from each of the temperature sensors, did not show distinguishing features similar to the acceleration sensors. An example of temperature sensor time series data is shown in Figure 3.

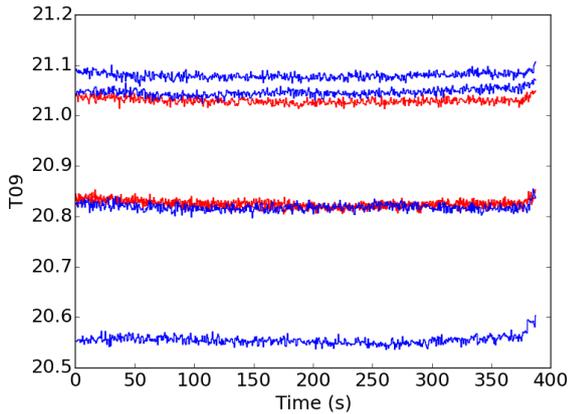


Figure 3: Sample Temperature Sensor Data (Fluid Temperature): blue and red traces indicate nominal and faulty conditions respectively

For this sensor data analysis, the features extracted are (i) average, (ii) standard deviation, (iii) maximum amplitude of FFT, and (iv) frequency for maximum amplitude of FFT. These four features are inspected visually to determine if imbalance could be classified by a simple linear classifier. The spindle acceleration (X, Y, and Z) feature (maximum amplitude of FFT) showed easily visible characteristics that can distinguish between degrees of imbalance. See Figure 4 for an example of visual classification based on X-axis acceleration data. Other sensor signals like power, pH, flow, and temperature did not exhibit such classification capability.

4.2 Control-based Segmentation followed by Sensor-based Diagnostics

The second diagnostic approach that we explored combines both sensor and control data in a coherent manner. The first step in this approach is to utilize the control signal to provide temporal segmentation, i.e., assuming quasi-steady state, the goal is to find the time intervals in which the following conditions are satisfied: (i) all experiments display same values for the primary control signal (actual spindle speed), and (ii) all the control signals are constant over the same period. Note that, to investigate the dynamic response, rather than quasi steady state response, the control signals should be consistent across the experiments so that responses are compared under the same set of control inputs. Figure 5 (a) shows the result of this temporal segmentation scheme. For each of the control signals, we have

computed the standard deviation at the each time step and identified the periods with standard deviation below a set threshold to find the consistent time intervals (shown as colored segments along the time axis in Figure 5 (b)). Then we find the intersection of the sets of consistent time intervals over all the control signals to determine the aggregate time intervals over which the control signals are statistically consistent (shown as black segments along the time axis in Figure 5 (c)).

These temporal segments are then mapped to sensor data to facilitate diagnostics. For each of 16 temporal segments, we computed features including (i) average, (ii) standard deviation, (iii) maximum FFT value, and (iv) FFT frequency at maximum amplitude. This step produces a 64 dimensional feature space to diagnose machine imbalance. As mentioned before, to avoid the overfitting we focus on linear transformation based approaches. We implemented Principal Component Analysis (PCA) to reduce the dimensionality from 64 to 4 (postulating that there should be 4 unique dimensions given the 4 uncorrelated features that we have selected). The PCA step is followed by Linear Discriminant Analysis to find the optimal coordinate transformation that provides maximum separation between classes. Result of this PCA-LDA analysis is shown in Figure 6 for Fluid Temperature sensor data. Another temperature sensor located at Spindle Motor also exhibits similar diagnostic capability after application of control based temporal segmentation. This demonstrates that control data can be used to provide context to sensor data in a way that helps diagnose machine imbalance. Thus, temperature sensor which had inferior diagnostic performance without context data, could classify imbalance perfectly when it is combined with additional context from control signal.

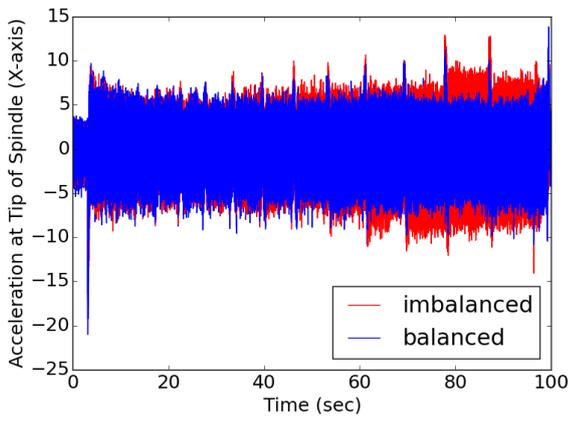
5 Conclusion and Discussion

This work explores various types of sensor and control data for diagnosing the imbalance of the machine tools. Our proposed approaches utilize sensor data that has not been used before for this purpose. This includes temperature, power, flow, and lubricant/coolant pH. In addition, our proposed techniques combine control and sensor signals to improve accuracy. Namely, by combining context information gained from the control signal, temperature sensor was able to classify machine imbalance conditions with much higher accuracy than using itself alone.

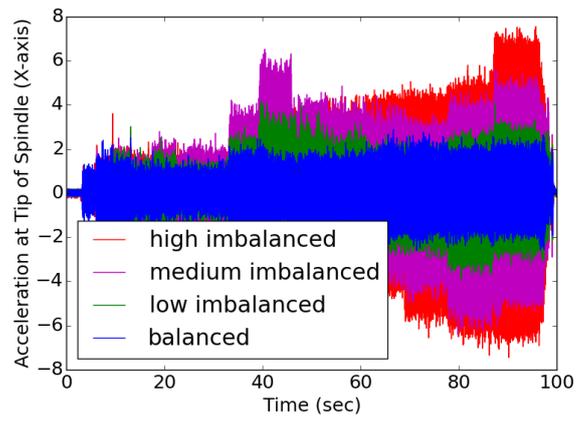
For future work, we will explore diagnostics based on control signal alone. Given that relying on sensor data typically requires adding sensors to existing machine tools, it would be ideal if we could diagnose imbalance of the machine from control signals that are usually recorded (i.e. no additional hardware required). The expectation is that if a machine tool uses feedback controls, then the control signal should be impacted by any change in the operational characteristics (in this case the imbalance of the machine tools).

References

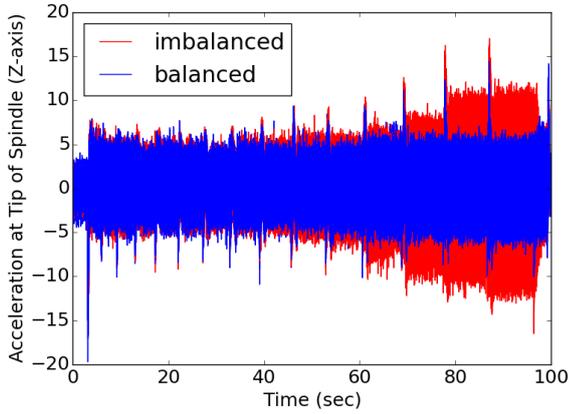
- [1] Carrie MacGillivray, Vernon Turner, and Denise Lund. Worldwide internet of things (iot) 2013–2020 forecast: Billions of things, trillions of dollars. *IDC. Doc.*, 243661(3), 2013.
- [2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.



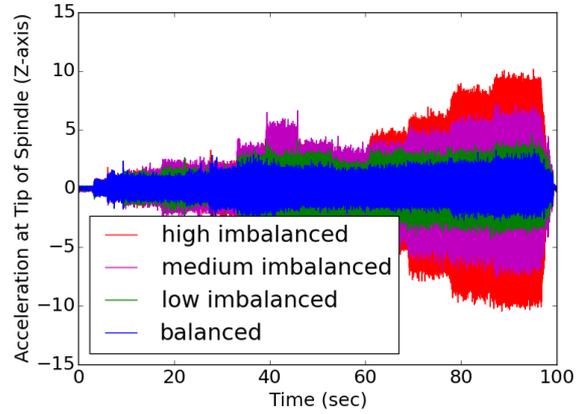
(a) Spindle X Acceleration: 2009 Data



(b) Spindle X Acceleration: 2010 Data

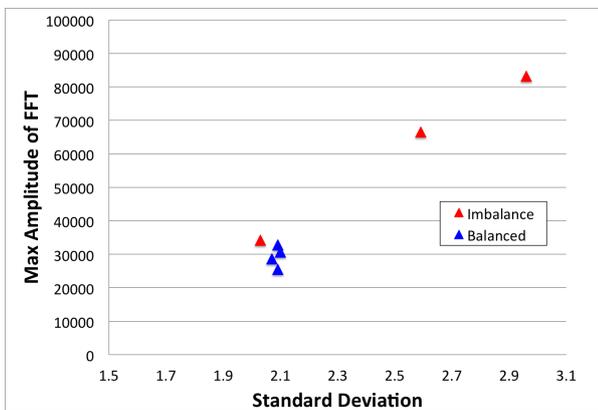


(c) Spindle Z Acceleration: 2009 Data

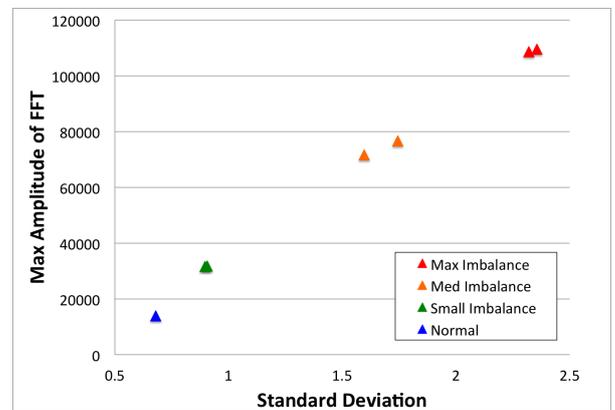


(d) Spindle Z Acceleration: 2010 Data

Figure 2: Spindle Acceleration Data for different imbalance level



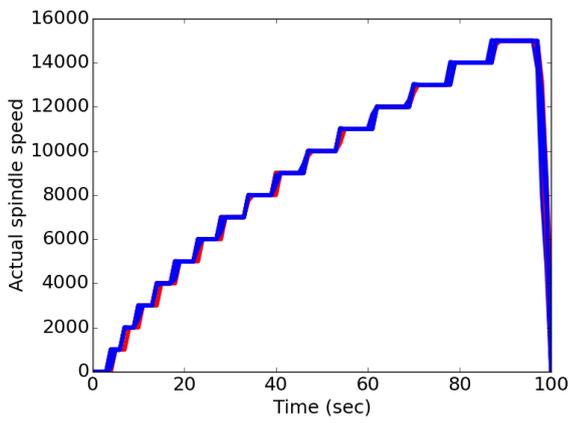
(a) Spindle X Acceleration for 2009 Data



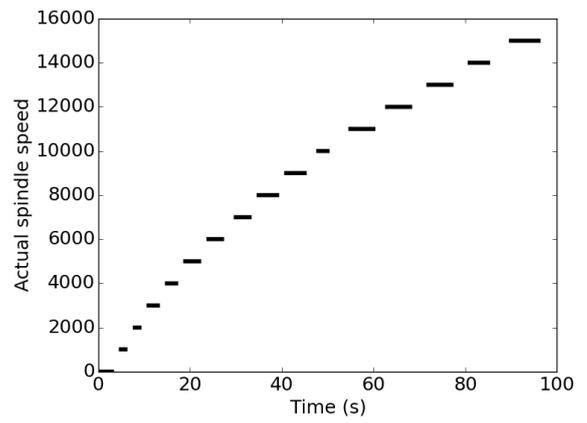
(b) Spindle X Acceleration for 2010 Data

Figure 4: Visual Classification using Spindle X Acceleration Sensor

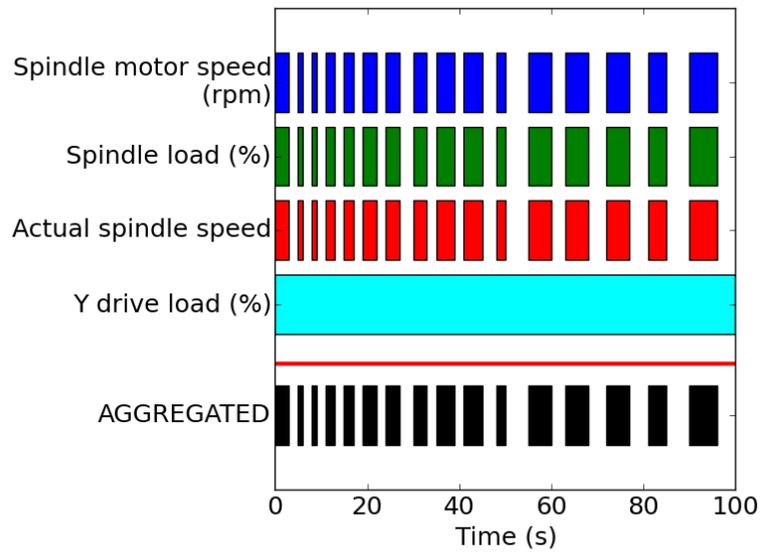
- [3] Markos Markou and Sameer Singh. Novelty detection: a review-part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.
- [4] Markou Markos and Sameer Singh. Novelty detection: a review-part 2: neural network based approaches. *Signal Processing*, 83(12):2499–2521, 2003.
- [5] Haitao Guo, Simon Watson, Peter Tavner, and Jiangping Xiang. Reliability analysis for wind turbines with incomplete failure data collected from after the date of initial installation. *Reliability Engineering & System Safety*, 94(6):1057–1063, 2009.
- [6] Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems-reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(1):314–334, 2014.



(a) Raw Spindle Speed Control

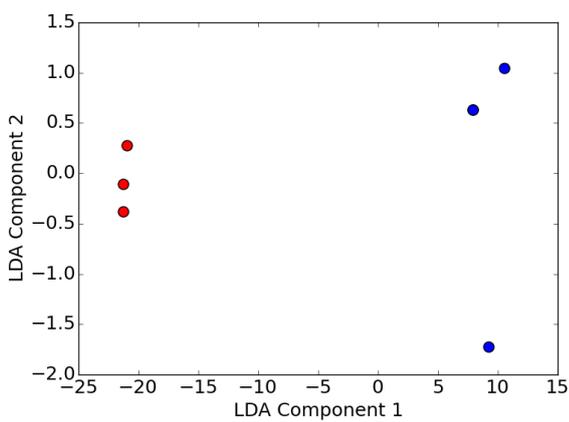


(b) Spindle Speed Control with Consistent Time Segment

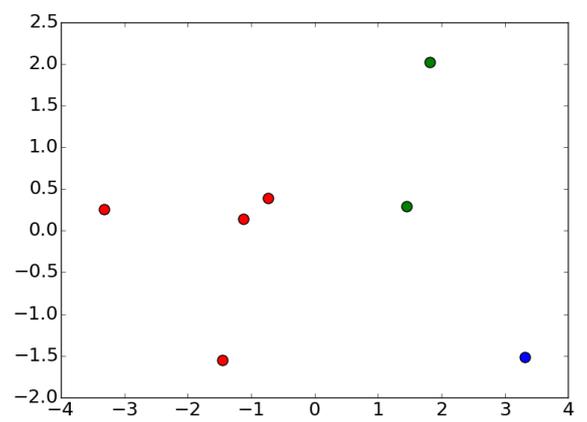


(c) Aggregating Control Signals

Figure 5: Time Series Segmentation



(a) Group 1



(b) Group 2

Figure 6: PCA-LDA Result using Fluid Temperature

[7] MTConnect Standard. Part 1-overview and protocol, version 1.01. *MTConnect Institute*, 2009.

[8] Radu Pavel, John Snyder, Nick Frankle, Gary Key, and Loran Miller. Machine tool health monitoring using

prognostic health monitoring software. In *MFPT 2010 Conference*, Huntsville, AL, April 2010.

- [9] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.
- [10] Qing Charlie Liu and Hsu-Pin Ben Wang. A case study on multisensor data fusion for imbalance diagnosis of rotating machinery. *AI EDAM*, 15(03):203–210, 2001.
- [11] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510, 2006.
- [12] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1):37–52, 1987.
- [13] Gary J Koehler and S Selcuk Erenguc. Minimizing misclassifications in linear discriminant analysis*. *Decision sciences*, 21(1):63–85, 1990.
- [14] Bo Yang, Songcan Chen, and Xindong Wu. A structurally motivated framework for discriminant analysis. *Pattern Analysis and Applications*, 14(4):349–367, 2011.

On the Learning of Timing Behavior for Anomaly Detection in Cyber-Physical Production Systems

Alexander Maier¹ and Oliver Niggemann^{1,2} and Jens Eickmeyer¹

¹Fraunhofer Application Center Industrial Automation IOSB-INA

e-mail: {alexander.maier, jens.eickmeyer}@iosb-ina.fraunhofer.de

² inIT - Institute Industrial IT

e-mail: oliver.niggemann@hs-owl.de

Abstract

Model-based anomaly detection approaches by now have established themselves in the field of engineering sciences. Algorithms from the field of artificial intelligence and machine learning are used to identify a model automatically based on observations. Many algorithms have been developed to manage different tasks such as monitoring and diagnosis. However, the usage of the factor of time in modeling formalisms has not yet been duly investigated, though many systems are dependent on time.

In this paper, we evaluate the requirements of the factor of time on the modeling formalisms and the suitability for automatic identification. Based on these features, which classify the timing modeling formalisms, we classify the formalisms concerning their suitability for automatic identification and the use of the identified models for the diagnosis in Cyber-Physical Production Systems (CPPS). We argue the reasons for choosing timed automata for this task and propose a new timing learning method, which differs from existing approaches and we proof the enhanced calculation runtime. The presentation of a use case in a real plant set up completes this paper.

1 Introduction

Many learning algorithms have been developed for the identification of behavior models of CPPS, e.g. [1], [2], [3]. However, most of the learning algorithms do not include timing information, not least because the modeling formalisms do not consider timing information.

Indeed, technical systems mostly depend on time, e.g. the filling of a bottle or the moving of a part on a conveyor belt. Therefore, many applications (such as the anomaly detection) require a model with timing information. Some faults only can be detected using timing information (especially degradation faults, e.g. a worn conveyor belt runs slower).

In this paper, we use the term "Cyber-Physical Systems (CPS)" for "systems that associate (real) objects and processes with information processing (virtual) objects and processes through open, partly global, anytime interconnected information networks". Further, a CPPS is a CPS in the context of an industrial production environment.

In this paper, we give a taxonomy of modeling formalisms. These formalisms are evaluated according to

specific features. The taxonomy is then used to evaluate whether the models can be identified automatically and used for anomaly detection.

Based on this evaluation, we present a timing learning method, which is used to learn the timing behavior as timed automaton. In contrast to other approaches, we use the underlying timing distribution function to differentiate between transitions with equal events which belong to different processes.

By calculating the computation runtime, we prove that our approach runs faster than other existing methods for timed automaton learning.

The presented learning method is used in an exemplary plant setup to demonstrate the suitability for anomaly detection in CPPS.

The paper is organized as follows: In Section 2 we evaluate some timing learning features and give a taxonomy of how these features are met by three categories of timing modeling formalisms, namely (i) Dynamic system models, (ii) Operational formalisms and (iii) Descriptive formalisms. In Section 3, we argue why we use timed automata as formalism, point out some challenges in timed automaton learning and present our timing learning approach. Further, we prove formally the enhancement of the calculation runtime of our approach. Section 4 completes the contribution with the presentation of a use case in a real plant. Finally in Section 5, we conclude this paper with a short discussion and give an outlook to future work.

2 Classification of Timing Learning Features and Algorithms

The modeling of time for computation purpose is a widely researched area (e.g. in [4], [5] and [6]). Many formalisms have been created to model different aspects of timing behavior. In this paper, some aspects are analyzed which have to be considered when choosing an appropriate timing modeling formalism. Based on this analysis, some modeling formalisms are evaluated according to their capabilities to model the timing behavior. One of those formalisms is chosen that is well suited for the anomaly detection in CPPS.

To keep the application domain in mind, a special focus is on modeling and identification of the timing behavior of CPPS. Additionally, the suitability of the modeling formalisms according to automatic learnability from observations only and the suitability for anomaly detection is evaluated.

2.1 Evaluation of Timing Modeling Features

Before choosing an appropriate timing modeling formalism some key issues have to be considered, which are listed below. Some of that and additional features are given in [5] where the authors provide a comprehensive analysis on timing modeling features and corresponding modeling formalisms is given.

Discrete or dense time domain

The separation of formalisms concerning the usage of discrete and dense time domains is a first natural categorization. Discrete time models comprise a set of isolated points, whereas dense time means that in a dense set, ordered by " $<$ ", for every 2 points t_1 and t_2 with $t_1 < t_2$ there is always a third point t_3 in between, such that $t_1 < t_3 < t_2$.

Explicit or implicit modeling of time

Another major distinctive feature is the possibility of implicit and explicit modeling of time. Model formalisms with explicit time allow the modeling of concrete time values for some specific event, e.g. "if the sensor is activated, start the conveyor belt within two seconds". Implicit modeling of time only gives information about the time duration as a whole.

One clock or many clocks

Furthermore, time model formalisms can be differentiated according to their number of used clocks. When dealing with independent modules within a system, the question arises whether to use one or many clocks. The usage of many clocks leads to the need of clock synchronization in the simulation step, whereas the usage of one clock only requires a transformation from an n-clock model to a 1-clock model.

Concurrency and composition

Most real systems are too complex to model them in one overall model. The behavior has to be divided into several subsystems, so that the overall model is a composition of its sub-models. For finite state machines, the number of states reduces enormously if the system is decomposed into subsystems. This is also referred to as modularization.

The decomposition is a less mature process. Difficulties can arise in the synchronization step. Mostly, the separated models of subsystems have equal or identical properties. Furthermore, the time bases can be different between the modules, discrete or continuous, or the time base is implicit for one module and explicit for another.

Single-mode and multiple-modes

The distinction between models, which can only cope with single-modes and models that additionally can deal with multiple-modes, goes a step deeper than concurrency and decomposition. A system may, at some point in time, abruptly change its behavior. In technical systems, this happens for reasons such as shifting a gear or stopping a conveyor belt. All state based models (e.g. statecharts, Petri nets or finite state machines) are able to describe multiple-mode systems, where equation based formalisms (e.g. ordinary differential equation) can only describe the behavior of single-mode systems.

Linear- and branching-time models

A difference can also be made between linear and branching time models [7]. Linear-time formalisms are interpreted over linear sequences of states. Each description refers to

(a set of) linear behaviors, where the future behavior from a given state at a given time is always identical. Branching-time formalisms are interpreted over trees of states. That means, in contrast to linear-time models, the future behavior of a given state at a given time can follow different behavior according to the tree.

A linear behavior can be regarded as a special case of a tree. Conversely, a tree can be treated as a set of linear behaviors that share common prefixes (i.e., that are prefix-closed); this notion is captured formally by the notion of fusion closure [8]. Thus, linear and branching models can be put on a common ground and compared.

2.2 Taxonomy of Timing Modeling Formalisms

Mainly, the timing modeling formalisms can be subdivided into three categories: (i) Dynamic system models, (ii) Operational formalisms and (iii) Descriptive formalisms:

Dynamic system models

In various engineering disciplines (like mechanical or electrical) and especially in control engineering, the so-called *state-space representation* is a common way to model the timing behavior of technical systems [9].

Three key elements are essential for the state-based representation: The vector \mathbf{x} with the state variables, the vector \mathbf{u} with the input variables and the vector \mathbf{y} with the output variables. All these values explicitly depend on the time at which they are evaluated (usually represented as $\mathbf{x}(t)$, $\mathbf{u}(t)$, and $\mathbf{y}(t)$), however, the timing information is not explicitly described in the form as "*the filling of the bottle takes five second*" i.e. it uses implicit timing.

The main advantage of dynamical system models is that very detailed physical models can be created using established mathematical methods. But this also can turn into a disadvantage. For many purposes, the models are too detailed, i.e. they are unsuitable for high-level description, since some expert knowledge is required to read and understand the models. As proposed in [10], dynamical systems can be used for the diagnosis of distributed systems.

Various methods exist to identify dynamic system models. These methods are grouped under the term model identification (sometimes the term "system identification" is also used), although, the model is not identified completely, but a structure model is presumed and the identification methods only determine the parameters. So, still some expert knowledge is necessary and manual work has to be done. In [6], Isermann describes some methods, e.g. by means of parameter estimation. The states itself are not identified.

Dynamic system models also can be used for fault detection (e.g. [11]). The model-based fault detection uses the inputs \mathbf{u} and the outputs \mathbf{y} to generate residuals \mathbf{r} , the parameter estimates $\hat{\Phi}$ or state estimates $\hat{\mathbf{x}}$, that are called features. A comparison of these features with the nominal values (normal behavior) detects changes of features, which lead to analytical symptoms s . The symptoms are then used to determine the faults.

Despite their suitability for the modeling of timing behavior, dynamic system models can hardly be learned automatically based on observations only, since the structure of the model has to be given and mostly only the parameters are identified.

Operational Formalisms

Operational formalisms further can be subdivided into (i) synchronous state machines and (ii) asynchronous abstract machines:

Synchronous state machines:

A large variety of synchronous state machines exists: finite state machine, statecharts, timed automaton, hybrid automaton, Büchi automaton, Muller automaton, and others (see [12]). Here, we confine our self to finite state machines and timed automata, the timing extension of finite state machines.

The main strength and the reason for the wide usage of finite state machines is their accessibility for humans and their simplicity. Often, processes or timing behavior are described by a sequence of events. In fact, technical systems are often programmed in state machines, e.g. using the standardized programming language from IEC 61131. Therefore, modeling the timing behavior of such technical systems, in the sense of finite state machines or timed automata, is consequential.

Some algorithms already exist to identify timed automata from observations (e.g. in [13], [14], [15], [16], [1]). Most automata identification algorithms are based on the state merging method. The basic procedure is illustrated in Figure 1. It works as follows:

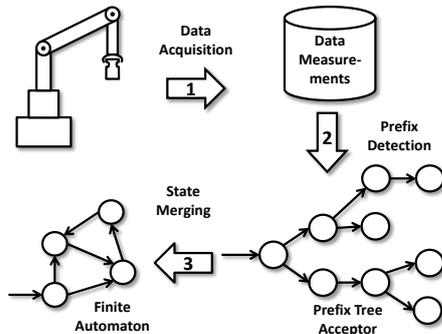


Figure 1: The principle of offline automaton learning algorithms using the state merging approach.

First, in step (1), the data is acquired from the system and stored into a database. In step (2), the observations are used to create a prefix tree acceptor (PTA) in a dense form, whereas equal prefixes are stored only once. Then, in step (3), in an iterative manner all pairs of states are checked for compatibility. If a compatible pair of states is found, the states are merged. In [13], additionally a transition splitting operation is introduced, which is executed when the resulting subtrees are different enough. The result is a finite automaton the generalizes the observed behavior in an appropriate way.

Finite state machines can also be used for fault detection and diagnosis (e.g. in [17], [18], [19]). Depending on the used formalism, different errors can be detected: wrong event sequence, improper event, timing deviation and error in continuous signals.

Asynchronous abstract machines:

Beside the finite state machines, which work synchronously, there exist formalisms that work asynchronously, called the asynchronous abstract machines. The most popular formalism in this group is Petri nets.

Petri nets are named according to Carl Adam Petri, who initially developed this modeling formalism [20]. A variety of Petri nets exists [21]. The most common type is place/transition-nets. It basically consists of states and transitions. Places store tokens and hand them over to the transitions. If all incoming places hold at least one token, a transition is enabled. An enabled transition will be fired. After firing the transition, tokens from incoming transitions are moved to outgoing transitions.

Petri nets also have been extended to handle timing information. Merlin and Farber proposed the first Timed Petri net in [22]. Each transition is extended with the minimum and maximum firing time, where the minimum firing time can be 0 and the maximum can be ∞ . A comprehensive survey on several timed extensions to Petri nets can be found in [23] and [24].

Furthermore, several approaches exist to identify Petri nets from sampled data. However, some requirements are put on the language to be identified or some assumptions are made, e.g. in [25], Petri nets are identified from knowledge of their language, where it is assumed that the set of transitions and the number of places is known. Only the net structure and the initial marking are identified.

Petri nets in general are suited for fault detection (e.g. in [26] or [2]). The different types of Petri nets (mainly condition/event-systems, place/transition-nets and high-level Petri nets) have different time and space complexity.

Descriptive Formalisms

As the name suggests, descriptive formalisms describe the model using a natural language, mostly based on mathematical logic [27]. Such formalisms are especially suited if some conditions have to be described.

Example 1. *If it is raining or if it was raining in the last two hours, then the street is wet.*

Similar rules can also be created for the prediction of output signals (actuators) based on the inputs (sensors) in a CPPS.

As already shown in Example 1, the conditions can also contain time information.

There exist different types of descriptive formalisms, e.g. first order logics, temporal logics, explicit-time logics or algebraic formalisms. Further details can be found in the literature, e.g. [27].

Some algorithms exist to identify descriptive models. For the prediction of the behavior of CPPS, a timed decision tree can be learned for instance. Examples for such learning algorithms are ID3 [28], the C4.5 algorithm as extension of the ID3 algorithm [29] or a generic algorithm for building a decision tree by Console [3].

Note that the rule can not always be interpreted backwards. Using Example 1, a reason for the wet street could be that somebody has washed his car on the street. Therefore, descriptive formalisms have a limited suitability for anomaly detection. The usage of descriptive formalisms for anomaly detection puts additional requirements on the rules, they have to be more concrete. Using the given example, it can be modified as follows:

Example 2. *The street is wet if and only if it is raining or it was raining in the last two hours.*

This rule allows a backward interpretation, if the reason for the wet street is unknown. However, the meaning of the

rule has now changed. Additionally, these kind of rules is hardly identifiable from observations only.

Comparison of Modeling Formalisms

Table 1 shows how the mentioned timing modeling features are met by the corresponding modeling formalisms.

It can be seen that operational and descriptive formalisms allow a similar level of timing modeling, while dynamic system models differ in nearly all features. In contrast to the other formalisms, dynamic system models use a dense time domain, only allow implicit modeling of, time, use one clock only and can model linear time models.

Please note the different possibilities to handle concurrent behavior. Petri nets are the first choice for this task. Using tokens, concurrent behavior can be modeled in one model. Timed automata and hidden Markov models (HMM) are able to decompose the behavior in several subsystems.

3 Automaton Learning

The decision of which formalism to use is based on several factors. These can differ based on the individual use case. Here, we consider the models to be used for learning and diagnosis of CPPS.

Despite there exist several algorithms for the identification of timed behavior, it can be seen in Table 2 that the usage of timed automata is a good choice.

- **Understandability:** In contrast to many other automatically identified models, the identified finite state machines can be better understood by third persons. They can be verified by experts.
- **Wide usage:** Finite state machines are widely used, e.g. for modeling or programming.
- **Learnability:** Finite state machines are suitable for automatic learning. The goal is to use as few expert knowledge as possible.
- **Diagnosability:** Finite state machines are suitable for fault detection. This applies for both, manually created and automatically identified finite state machines.
- **Suitability for verification:** The identified finite state machines can be used for automatic verification.
- **Modification:** The identified finite state machine can be manually modified and adapted after learning. This can also be done automatically.

3.1 Challenges in Automaton Learning

Some algorithms have already been introduced for the identification of timed automata, see Section 2.2. However, there are still some challenges in learning timed automata. This applies in particular to the time factor.

- **Identification of states and events:** The timing behavior includes not only the time stamps for some observations, but also some states and transitions with timed events in between. Many learning algorithms (especially for learning of Markov chains) assume the states and transitions as given and only learn the transition probabilities. Here, the structure (states and events) is not given but has to be identified from observations.
- **Timing representation method:** Additionally, an appropriate timing representation method has to be chosen, which is able to correctly describe the technical processes. At the beginning of Section 3.2 we review

some state of the art timing representation methods and propose our solution.

- **Relative or absolute time base:** The time base is also a very important issue. The base can be either absolute e.g. referred to the beginning of a production cycle or relative to the last event.
- **Number of clocks:** Technical systems may be programmed using a certain number of clocks. These have to be identified or the behavior has to be expressed using only one clock.

Timed automata allow both, one and many clocks. However, in [13] Verwer showed that 1-clock timed automata and n-clock timed automata are language-equivalent, but in contrast to n-clock timed automata, 1-clock timed automata can be identified efficiently.

- **Event splitting:** When do events with different timing belong to the same event, or do they describe different events? As can be seen in Figure 2, the events can be split based on the timing, which is based on the container size: The robot needs more time to move the big container compared to the small one, this is captured in the given probability distribution function over time. More formally: The event's timing distribution function can comprise several modes that have to be identified.

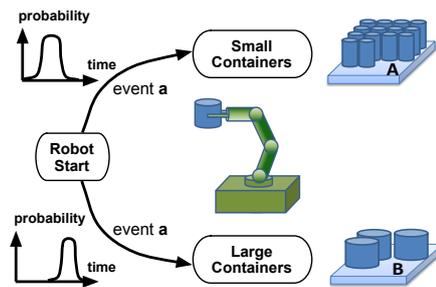


Figure 2: The timing behavior changes based on the container size.

- **Event splitting or timing preprocessing:** Continuing from the previous point, additionally the question arises that whether the modes are identified during the learning process itself or whether a preprocessing can be used to identify multiple modes and use this information in the learning process, avoiding the additional splitting operation.

3.2 Timed Automaton Learning Algorithm

Several algorithms have been introduced to learn an automaton based on observations of the normal behavior only. While most automaton identification algorithms do not consider time (e.g. MDI [30] and Alergia [31]), recently only few algorithms have been introduced that identify a Timed Automaton. RTI+ [13] and BUTLA/HyBUTLA [16] learn in an offline manner, i.e. first the data is acquired and stored and then the automaton is learned. However, for the case that observations cannot be stored, an online learning algorithm is desirable, which includes each observed event online, without a preprocessing. OTALA [1] is an extension of BUTLA and learns a timed automaton in an online manner.

Table 1: Taxonomy of the timing modeling features and how they are satisfied by the corresponding modeling formalisms.

	operational Formalisms			descriptive Formalisms	Dynamic system models
	Timed Automata	HMM	Petri nets	e.g. Rule-based system	e.g. state space representation
Discrete or dense time domain	discrete	discrete	discrete	discrete	dense
Explicit or implicit modeling of time	explicit	explicit	explicit	explicit	implicit
One clock or many clocks	one/many	one/many	one/many	one/many	one
Concurrency and composition	++	++	+++	+	+
Single-mode and multiple-modes	single/multiple	single/multiple	single/multiple	single	single
Linear- and branching-time models	linear/branching	linear/branching	linear/branching	linear/branching	linear

Table 2: Satisfiability of the mentioned properties by different timing modeling formalisms.

	Timed Automata	HMM	Petri nets	Rule-based system	State space representation
Understandability	+++	++	++	+++	+
Wide usage	+++	+++	++	++	++
Learnability	+++	++	+	+++	+
Diagnosability	+++	++	++	++	++
Suitability for verification	+++	+++	++	++	+
Modification	+++	++	++	+++	+

A crucial issue for the modeling formalism of timed systems is the representation of the timing information. Usually, timed automata use a single clock only and therefore a relative time base is required, where a relative time stamp represents the passed time from entering until leaving a state. The timing information is annotated in the transition next to an event. The usual way is to use intervals recording the minimum and maximum observed time values for a specific event [13], [14], [15], [1].

RTI+, the first algorithm for the identification of timed automata [13], included a transition splitting operation in addition to the merging operation. The timing in the transitions is represented with histograms using bins and uniform distribution [13]. During the state merging procedure, it is also checked, whether a transition can be split. A transition is split when the resulting subtrees are different enough. However, the splitting operation is associated with a high calculation time, since depending on the bin size, all possible splits have to be calculated. The disadvantage of this approach is that the bin size has to be set manually by experts. Further, it does not take the underlying distribution into account.

In contrast to other existing algorithms for the identification of timed automata, our proposed identification algorithm BUTLA [16] uses probability density functions over time (PDFs) to express the timing behavior. Unlike other approaches, we base our decision on the timing information itself, not on the subtree resemblance.

The identification algorithm BUTLA follows the methodology from Figure 1. Additionally, instead of the splitting operation, a preprocessing step is introduced, which identifies the timing behavior and captures different behavior pat-

terns as shown in Figure 2.

Timing preprocessing

The timing of events is analyzed in a preprocessing step. The relative time values of each event are collected in a histogram. It is decided whether the timing behavior is subdivided into multiple modes based on this histogram and the resulting probability density distribution over time. In case of multiple modes, an event is separated according to the number of modes in the PDF such that each event consists of only one mode. For instance, an event e_i with 2 modes is separated into $e_{i,1}$, $e_{i,2}$, as can be seen in Figure 3.

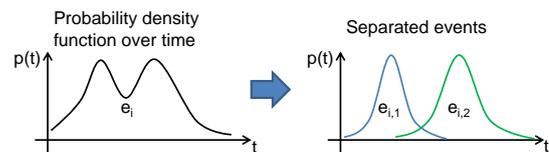


Figure 3: An event with a multi-mode timing behavior is separated into its modes.

For the detection of multiple modes in events, three methods have been evaluated:

- *Kernel density estimation*: This version is straight forward by estimating the density of the distribution function and subdividing at local minimums. It is optimized for efficient computation time. Nevertheless it delivers useful results.
- *Expectation Maximization (EM) - algorithm*: This method is well-known from the state of the art. It per-

forms well, but the number of mixed distribution functions has to be known or determined subsequently by trying all values and take the best fitting.

- *Variational Bayesian inference*: This version has the weakest performance but delivers the best results. The number of overlapping distribution function is calculated in an iterative manner.

Due to the high computation effort of the EM-algorithm and Variational Bayesian inference, we chose to use the kernel density estimation for the timing preprocessing in BUTLA. The determination of the timing modes using the kernel density estimation works as follows:

First, for each event e , all timing values t_1, t_2, \dots, t_k are collected and stored in a list $\{e, \{t_1, t_2, \dots, t_k\}\}$, $k \in \mathbf{N}$ is the number of collected timing values for one event. Then, the PDFs are calculated using the kernel density estimation method for each event. Density estimation methods use a set of observations to find the subjacent density function. Given a vector \mathbf{t} with the time values of the observations, the underlying density distribution for a time value t can be estimated as

$$f(t) = \frac{1}{N} \sum_{i=1}^N k(\mathbf{t}_i; t) \quad (1)$$

where $N \in \mathbf{N}$ is the number of time values in the vector of observations and $k(\mathbf{t}_i; t)$ is a non negative kernel function

$$\int_{-\infty}^{\infty} k(\mathbf{t}; t) dt = 1. \quad (2)$$

As underlying probability distribution, we use the Gaussian distribution, which is defined as:

$$G(\mu, \sigma^2, t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \quad (3)$$

where σ^2 is the bandwidth (smoothing factor), μ the mean value and t is the time value, for which the probability is calculated.

The choice of the bandwidth is important for the correctness of the results and it is the subject of research in different publications (e.g. [32]). In the case of identifying the normal behavior of production plants, it is useful not to use a fixed value for smoothing factor but to keep it variable. Here, the variable smoothing factor is 5% of the current value. This results in the greater variance for greater time values and smaller variance for smaller time values. Therefore, the density is estimated as:

$$f(t) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\sqrt{2\pi \cdot 0.05t_i}} e^{-\frac{(x-t)^2}{2 \cdot 0.05t_i}}. \quad (4)$$

In the next step the local minimums in the calculated PDF are localized. One mode is assumed to be between the local minimums.

Finally, referring to the original data (discrete time values) and based on the assumption of normally distributed data, the needed statistic parameters (mean μ and standard deviation σ) are calculated. This is done for each mode: between the minimum value, all local minimums and the maximum value.

Using this preprocessing of the timing information, the time-consuming splitting operation during the state merging

procedure is not necessary, since the transitions are already split according to the identified timing modes.

3.3 Analysis of the Timing Preprocessing

Figure 2 illustrates that a state can be a starting point for different processes: When the robot is started, it depends on the size of the containers that which of the sub-trees is taken for the further process, based on the time that is needed to move the container. Different possibilities exist to identify the different timing behavior of the sub-trees.

The algorithm RTI+ uses a splitting operation, which calculates a p-value for all possible splits and its sub-trees. If the lowest p-value of one split is less than 0.05, the transition is split.

Figure 4 illustrates the problem of the splitting operation. The main drawback of using the splitting operation is that it requires additional computation time. First, all possible splits have to be evaluated. Based on the number of observations, these can be a huge amount. And after finding the best splitting point based on the smallest p-value, the transition has to be split. Here, for all postfixes of the corresponding transition, it has to be decided that which path to follow. Since all these paths are mixed in the previous states, the information that which path follows which states, based on the original data, has to be stored somehow. This leads to a huge memory consumption. To avoid this high memory consumption, RTI+ renews the prefix tree acceptor beginning with the corresponding state after each splitting operation. However, this is still time and space consuming.

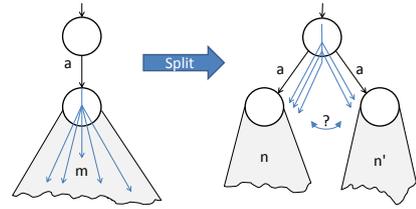


Figure 4: The problem of the splitting operation.

Proposition 1. *The time complexity of calculating and performing a splitting operation is $\mathcal{O}(m^2 \cdot n^2)$, where m is the number of input samples and n is the number of states in the PTA.*

Proof. For each transition (in worst case there are $n - 1$ transitions in the PTA, if it is a linked list of states with only one input sample or all input samples follow the path), the p-value has to be calculated (which has to be done for each input sample using the certain transition). Therefore, the complexity for calculating the p-values is $\mathcal{O}(m \cdot n)$.

One splitting operation itself also needs time in $\mathcal{O}(m \cdot n)$ for the creation of the PTA with m input samples, where each can have n states.

In the worst case, if each transition has to be split, the complexity is in $\mathcal{O}(m^2 \cdot n^2)$. \square

BUTLA firstly uses a preprocessing of timing values to avoid this splitting operation. This version is based on the assumption that events with the same changing signals but different timing behavior describe different behavior.

In the preprocessing step, events with multiple timing modes are identified. These modes are used for the creation

of the prefix tree. Events with the same symbol but arising from different timing modes are handled as different events and lead to different states in the prefix tree. In the identification phase, these events are also handled as different. Using this preprocessing step, the splitting process can be omitted. This leads to a computation speed increase.

Proposition 2. *The time complexity of calculating the timing modes in a preprocessing step is in $\mathcal{O}(n)$, where n is the number of observed events.*

Proof. Since this is during the preprocessing step and the PTA does not exist so far, the worst case is not dependent on the PTA structure, but only on the number of incoming events and the number of symbols.

First, the time stamps for each symbol in the alphabet $a \in \Sigma$ have to be collected. This takes time $\mathcal{O}(n)$.

Then for each $a \in \Sigma$, the probability density distribution over time has to be calculated. For this, Equation 4 is computed. Note that all events are not considered for a single symbol $a \in \Sigma$, but only those that belong to this symbol a . All computations together need time $\mathcal{O}(n)$. Additionally the local minimums have to be identified, which is also done in $\mathcal{O}(n)$.

All these steps are performed subsequently and therefore the overall time complexity for the preprocessing step is $\mathcal{O}(n)$. \square

Using the preprocessing step, the computation time can be reduced compared to the splitting version. While the splitting version runs in polynomial time, we could reduce this additional timing computation to linear time using the preprocessing step.

4 Learning Automata Results

As mentioned before, the goal of the identified automata is the usage for anomaly detection. An exemplary plant at the institute has been used for experimental results. Figure 5 shows a part of the Lemgo Model Factory and the identified models of two modules.

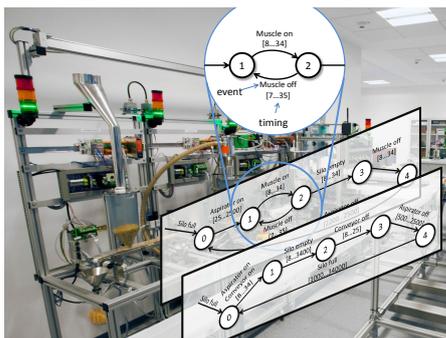


Figure 5: Example plant with identified models for two modules.

During the anomaly detection phase, the running plant's timing behavior is compared to the prognosis of the automaton. A timing anomaly is signaled whenever a measured timing is outside the timing interval in the learned timed automaton. Here, the interval is defined as $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$, $k \in \mathbf{R}_+$ where μ is the mean value of the corresponding

original observations' timings and σ is the standard deviation.

In a first experiment, the Lemgo Model Factory (see Figure 5) is used. A frequently occurring error for example is the wear of a conveyor belt which leads to a decrease in the system's throughput. 12 production cycles are used to identify a normal behavior model. The PTA comprises 6221 states. BUTLA reduces this to 13 states—this corresponds to a compression rate of 99.79%.

To verify the model learning algorithm with a high amount of data, in a second experiment, data is generated artificially using the modified Reber grammar (extended with timing information). 1000 samples are generated to learn the model, then 2000 test samples are created where 1000 comprise timing errors. From the initial 5377 states in the PTA, a model with 6 states is learned.

Table 3 shows the error rates for the anomaly detection applied to both data sets using different factors k in the timing intervals.

Table 3: Experimental results using real and artificial data.

	$k=1$	$k=2$	$k=3$	$k=4$
false negative rate (%) - LMF	2	5.3	12.8	30
false positive rate (%) - LMF	12	4.2	2	0
false negative rate (%) - Reber	0	1.3	7.5	21
false positive rate (%) - Reber	9	3.1	1.1	0

The experimental results in Table 3 show that the false positive rate could be reduced by enlarging the time bounds. But at the same time, the false negative rate rose. The application of the enlargement of the time requires a trade off between false positive and false negative rate. This has to be done separately for each application.

5 Conclusion

In this paper we analyzed the possibilities of learning the timing behavior for anomaly detection in CPPS. First, we gave a taxonomy of timing modeling formalisms. Based on this taxonomy we analyzed whether the models can be identified automatically and whether they are suitable for anomaly detection.

Timed automata are often the first choice for the modeling of timed behavior of CPPS, especially for the modeling of sequential timed behavior.

Due to the intuitive interpretation, timed automata are well-suited to model the timing behavior. In our proposed learning method, we used probability density distribution functions over time for the timing representation. In a preprocessing step multiple modes in single transitions are identified, this enables the omission of the time consuming splitting operation.

We proved the runtime enhancement formally and gave some experimental results which prove the practicability of timed automata for automatic identification and for anomaly detection.

References

- [1] A. Maier. Online passive learning of timed automata for cyber-physical production systems. In *The 12th IEEE International Conference on Industrial Informatics (INDIN 2014)*. Porto Alegre, Brazil, Jul 2014.

- [2] M.M. Mansour, M. Wahab, and W.M. Soliman. Petri nets for fault diagnosis of large power generation station. *Ain Shams Engineering Journal*, 4(4):831 – 842, 2013.
- [3] L. Console, C. Picardi, and D.T. Dupré. Temporal decision trees: Model-based diagnosis of dynamic systems on-board. *CoRR*, abs/1106.5268, 2011.
- [4] A. Maier. *Identification of Timed Behavior Models for Diagnosis in Production Systems*. PhD thesis, University of Paderborn, 2015.
- [5] C. A. Furia, D. Mandrioli, A. Morzenti, and M. Rossi. Modeling time in computing: A taxonomy and a comparative survey. *ACM Comput. Surv.*, 42(2):6:1–6:59, March 2010.
- [6] R. Isermann and M. Münchhof. *Identification of Dynamic Systems: An Introduction with Applications*. Advanced textbooks in control and signal processing. Springer, 2010.
- [7] M. Y. Vardi. Branching vs. linear time: Final showdown. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2001, pages 1–22, London, UK, 2001. Springer-Verlag.
- [8] R. Alur and T. A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *In Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 177–186. IEEE Computer Society Press, 1992.
- [9] H. Khalil. *Nonlinear Systems*. Prentice Hall, January 2002.
- [10] S. Indra. Decentralized Diagnosis with Isolation on Request for Spacecraft. In Astorga Zaragoza, editor, *Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 283–288, August 2012.
- [11] R. Isermann. Model-based fault detection and diagnosis - status and applications. In *16th IFAC Symposium on Automatic Control in Aerospace*, St. Petersburg, Russia, 2004.
- [12] W. Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B)*, pages 133–191. MIT Press, Cambridge, MA, USA, 1990.
- [13] S. Verwer. *Efficient Identification of Timed Automata: Theory and Practice*. PhD thesis, Delft University of Technology, 2010.
- [14] M. Roth, J. Lesage, and L. Litz. Black-box identification of discrete event systems with optimal partitioning of concurrent subsystems. In *American Control Conference (ACC), 2010*, pages 2601–2606, June 2010.
- [15] M. Roth, S. Schneider, J.-J. Lesage, and L. Litz. Fault detection and isolation in manufacturing systems with an identified discrete event model. *Int. J. Systems Science*, 43(10):1826–1841, 2012.
- [16] O. Niggemann, B. Stein, A. Vodenčarević, A. Maier, and H. Kleine Büning. Learning behavior models for hybrid timed systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.
- [17] S. Tripakis. Fault diagnosis for timed automata. In Werner Damm and Ernst-Rüdiger Olderog, editors, *FTRTFT*, volume 2469 of *Lecture Notes in Computer Science*, pages 205–224. Springer, 2002.
- [18] P. Supavatanakul, C. Falkenberg, and J. J. Lunze. Identification of timed discrete-event models for diagnosis, 2003.
- [19] Z. Simeu-Abazi, M. Di Mascolo, and M. Knotek. Diagnosis of discrete event systems using timed automata. In *International Conference on cost effective automation in Networked Product Development and Manufacturing*, Monterrey, Mexico, 2007.
- [20] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962.
- [21] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
- [22] P. M. Merlin and D. J. Farber. Recoverability of communication protocols—implications of a theoretical study. *Communications, IEEE Transactions on*, 24(9):1036–1043, Sep 1976.
- [23] A. Cerone. *A Net-based Approach for Specifying Real-time Systems*. Serie TD. Ed. ETS, 1993.
- [24] A. Cerone and A. Maggiolo-Schettini. Time-based expressivity of time petri nets for system specification. *Theoretical Computer Science*, 216(1 - 2):1 – 53, 1999.
- [25] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of Petri Nets from Knowledge of Their Language. *Discrete Event Dynamic Systems*, 17(4):447–474, 2007.
- [26] P. Nazemzadeh, A. Dideban, and M. Zareiee. Fault modeling in discrete event systems using petri nets. *ACM Trans. Embed. Comput. Syst.*, 12(1):12:1–12:19, January 2013.
- [27] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.
- [28] J. R. Quinlan. Induction of decision trees. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.
- [29] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [30] Franck Thollard, Pierre Dupont, and Colin de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. of the 17th International Conf. on Machine Learning*, pages 975–982. Morgan Kaufmann, 2000.
- [31] Rafael C. Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In *GRAMMATICAL INFERENCE AND APPLICATIONS*, pages 139–152. Springer-Verlag, 1994.
- [32] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Annals of Statistics*, 38(5):2916–2957, 2010.

The Case for a Hybrid Approach to Diagnosis: A Railway Switch

Ion Matei and Anurag Ganguli and Tomonori Honda and Johan de Kleer

Palo Alto Research Center, Palo Alto, California, USA

e-mail: {imatei,aganguli,thonda,dekleer}@parc.com

Abstract

Behavioral models are at the core of Fault-Detection and Isolation (FDI) and Model-Based Diagnosis (MBD) methods. In some practical applications, however, building and validating such models may not always be possible, or only partially validated models can be obtained. In this paper we present a diagnosis solution when only a partially validated model is available. The solution uses a fault-augmented physics-based model to extract meaningful behavioral features corresponding to the normal and abnormal behavior. These features together with experimental training data are used to build a data-driven statistical model used for classifying the behavior of the system based on observations. We apply this approach for a railway switch diagnosis problem.

1 Introduction

Consider the case of developing diagnostic software for a complex system (for this paper our example is a railway switch). The task is to determine from operational data whether the switch is operating correctly or in one of a fixed number of fault modes. We are given the following very limiting (but all too common) conditions: (a) very limited resources to complete the project (a few man months); (b) limited number of sensors; (c) unavailability of the model of the system; (d) unavailability of the system itself (would require an instrumented private rail system); (e) unavailability of the parameters of the system components; (f) limited nominal data; (g) extremely limited fault data (supplied as time series); (h) highly non-linear multi-physics system having multiple operating modes. Broadly speaking there are three approaches to this type of problem: Model-Based Diagnosis (MBD), Fault Detection and Isolation (FDI) and Machine Learning (ML). None of these approaches is adequate of this task. MBD and FDI require models and parameters which are unavailable. ML approaches will require a large amount of training data, and most approaches would require extensive feature engineering. In this paper we will demonstrate a hybrid approach to this task which was ultimately fully satisfactory for the train company. Many real world diagnostic tasks have similar limitations and we believe our approach is one that yields good diagnostic algorithms for many cases.

At a high level our approach is as follows. First we build by hand an approximate model in Modelica (our switch

model ultimately has 56 continuous time state and more than 2000 time-varying variables). We require this model to contain the key mechanisms which comprise a switch mechanism. Under the limiting conditions, building an accurate model of the system proved to be impractical and therefore we used simplified models for the system's components. For example, we model the controller as a PID controller while the actual mechanism surely has a more complex one. The Modelica model is fault augmented [Minhas *et al.*, 2014] including parameters which represent the fault amounts for wear, etc. Second, we develop ML classifiers to detect and diagnose faults by running the Modelica model repeatedly with various fault amounts. We mix noise in the simulation to avoid over-fitting. For the ML classifier to work requires developing a set of features for the signal. Each time series is segmented at defined conditions and a set of features is designed (e.g., mean in segment, max in segment). Multiple ML techniques can develop a classifier, the best we found are based on random-forest. Third, we throw away the model — it was only important to develop the features and the classifier. We now use the classifiers developed for the synthetic data on the real data. We were able to *detect* faults with a high level of accuracy, but were only partially successful in identifying the correct fault mode (or nominal) for the operating system. Independently, we showed that given enough data for the various fault modes, *using the same set of features*, a ML classifier can be designed that also achieves a high diagnostic accuracy. The latter effort is not the subject of the paper. Overall, the customer was very satisfied with the results of the project. Throughout the rest of the paper we describe in detail the procedure described above.

1.1 FDI and MBD

In model-based approaches (FDI and MBD), the diagnosis engine is provided with a model of the system, values of the parameters of the model and values of some of its inputs and outputs. Its main goal is to determine from only this information whether the system is malfunctioning, which components might be faulty and what additional information need to be gathered (if any) to identify the faulty components with relative certainty. The distinguishing features of the MBD [de Kleer *et al.*, 1992] approach are an emphasis on general diagnostic reasoning engines that perform a variety of diagnostic tasks via on-line reasoning, and inference of a system's global behavior from the automatic combination of physical components. Hence, MBD models are compositional - the model of a combination of two systems

is directly constructed from the models of the constituent systems. FDI methods can work with both physics-based and empirical models. The physics-based models are usually flattened, that is, the components and sub-components structure is lost into an overall behavioral model. Often, the faults are seen as separate inputs that need to be computed by the diagnosis engine. The disadvantage of this approach is that the physical semantics of the faults is ignored. In addition, treating the faults as exogenous inputs ignores the fact that the abnormal behavior may in fact depend on the variables of the systems. However, many FDI techniques were shown to be effective in diagnosing dynamical systems [Gertler, 1998; Isermann, 1997; 2005; Patton *et al.*, 2000].

The above discussion emphasizes the need for a model when using either an FDI or MBD approach. As we will see later in the paper, there are cases when such a model is very difficult to obtain and (more importantly) validate, or only a partial model is available. Naturally, both FDI and MBD approaches would not fare well in such a scenario. When no model is available, data-driven methods can be used to learn the behavior of the system and use this knowledge to predict the system behavior. Such methods require experimental data corresponding to the normal and abnormal behavior for classification purposes; data that is used to extract features representative for the system's behavior. The set of features together with observations of the system (output measurements) are used to learn a data-driven statistical model that is further used to classify the current observed behavior. Namely, when new data is available it is fed into the data-driven model, which in turn will provide a "best guess" to which class of behavior (normal or abnormal) the data corresponds to. It is well recognized that in data-driven approaches, the effectiveness of the classification is highly dependent on the quality of the features used for learning.

In this paper, we begin to bridge the gap between pure model-based and data-driven methods with a more hybrid approach. We propose the use of a partially validated model to help us determine a set of features that are representative for the normal and abnormal behavior. In this approach we build a physics based model of the system, emphasizing its components and sub-components. Due to the lack of sufficient technical specifications and measurement data, only partial validation is achieved. By this we mean that only a sub-set of the variables of interest match their counterpart in the experimental data. The rest of the variables, although not completely matching the real data, they do exhibit similar characteristics compared to the real data, e.g., same number of maxima, minima, or common regions of increasing/decreasing values, etc. In other words they are qualitatively equivalent. The physics-based model is further extended to include behaviors under different fault operating modes. In particular, physics-based models for the faults are included in the nominal model. The fault-augmented model is then used to generate synthetic simulated normal and abnormal (including multiple faults) behavior and extract representative features that are used in a data-driven approach. Note that although ideally we would like to execute the feature extraction step automatically, in this paper it is performed manually as the automatic feature extraction is a challenging problem in its own. The diagnosis procedure described above is pictorially presented in Figure 1.

The rest of the paper is organized as follows: in Section

2 we motivate and describe the railway switch diagnosis problem. Sections 3 and 4 present the physics-based model, its fault-augmented version and the partial validation of the system. Section 5 describes the diagnosis solution under a partially validated physics-based model while Section 6 puts our solution in the context of exiting work on railway switch diagnostics.

2 Problem Description

Railway signaling equipment (including switches) generates approximately 60% of the failure statistics related to traffic disruptions due to signalling problems. As a consequence more and more attention is paid to railway safety and optimal railway maintenance. As a result of the rapid technological advances in microelectronics and communication technologies in the past decades, it has become possible to add sensing and communication capabilities to railway equipment such as switches, to detect equipment failure and therefore to enhance the quality of the railway service. Although these sensing capabilities allow for easy detection of faults in the electrical components of the equipment, a significant number of faults related to the mechanical components affect parameters whose monitoring would be difficult either due to cost or impracticality of sensor placement.

The rail switch assembly considered in this paper is shown Figure 2. The component responsible for moving the switch blades is the point machine. The point machine has two sub-components: a servo-motor (generates rotational motion) and a gear-cam mechanism (amplifies the torque generated by the motor and transforms the rotational motion into a translational motion).

The adjuster transfers the motion from the point machine to the load (switch blades) through a drive rod. In particular, by adjusting two bolts, the adjuster controls the time when the switch blades start moving having as reference the time when the drive rod commence moving. The switch blades are supported by a set of rolling bearings to minimize motion friction. The manufacturer of the point machine endowed the equipment with a series of sensors that can measure the motor's angular velocity and torque, and the cam's angle and stroke (linear position). These sensors log data in real time which is then sent to a central station for analysis. These sensors were installed by design on the point machine to monitor its safety. Although the operator of the railway switch is also interested in the diagnosis of the point machine, other possible faults are of interest as well. The faults considered in this paper are as follows: *loose lock-pin* fault (at the connection between the drive rod and the point machine), *adjuster bolts misalignment* (the bolts move away from their nominal position), *missing bearings* and the presence of an *obstacle* preventing the completion of the switch blades motion. Adding new sensors measuring forces applied to the switch blades or the position of the switch blades may facilitate immediate detection of such faults. However, due to the sheer number and possible configurations of switches in the railway transportation network, this is not a scalable solution. Therefore, the challenge is to diagnose the aforementioned faults using *only the available measurements*.

3 System Modeling

This section presents the fault augmented physics-based model of railway switch assembly, together with some

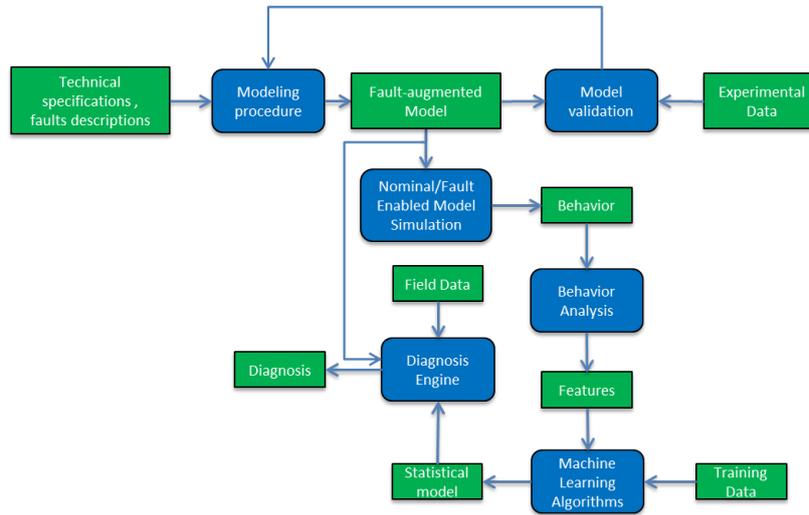


Figure 1: Diagnosis procedure with partially validated model

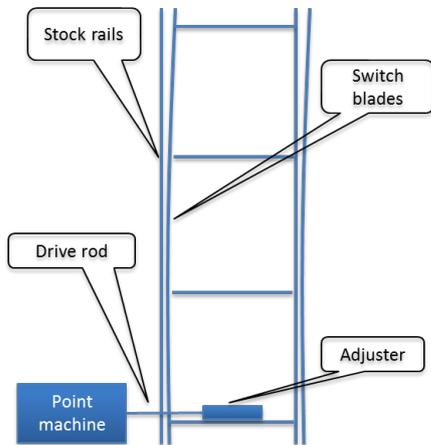


Figure 2: Diagnosis procedure with partially validated model

model validation results. Such models provide deeper insight on the behavior of the physical system. Simulated behavior helps with learning of normal and abnormal behavior patterns. The abnormal patterns are especially useful when not enough experimental data describing the abnormal behavior is available. The modeling process consists of decomposing the system into its main components, build physical models and combining them into an overall model of the system. We used the Modelica language to construct the model, which is a non-proprietary, object-oriented, equation based language to model complex physical systems [Tiller, 2001]. Models for the three main components of the railway switch, the point machine, the adjuster and the switch blades, are presented in what follows.

3.1 Point machine

The point machine is the component of the railway switch system that is responsible for moving the switch blades and locking them in the final position until a new motion action is initiated. It is composed of two sub-components: servomotor and gear-cam mechanism. The electrical motor transforms electrical energy into mechanical energy and gener-

ates a rotational motion. The gear-cam mechanism scales down the angular velocity of the motor and amplifies the torque generated by the motor. In addition, it transforms the rotational motion into a translational motion.

Servomotor

No technical details were provided on this component, such as type of motor or type of controller. Values for technical parameters (e.g., armature resistance, motor shaft inertia) were not available either. This information was not available to the switch operator either. Therefore, as a result of a literature review on the type of motors used in railway switches, a DC-permanent motor was chosen to be the most likely candidate. The dynamical model for this component is given by

$$\begin{aligned} L_a \frac{di(t)}{dt} &= -R_a i(t) - K_e \omega(t) + v(t), \\ J \frac{d\omega(t)}{dt} &= K_t i(t) - B\omega(t) - \tau(t), \end{aligned}$$

where $v(t)$ acts as input signal, $\omega(t)$ is the angular velocity at the motor flange that acts as output, $\tau(t)$ is the torque load of the motor and $i(t)$ is the current through the armature. Generic motor parameters from the literature were also chosen [Zattoni, 2006]. One question that may arise is if an empirical model can be estimated. Unfortunately since only the output $\omega(t)$ is available, an empirical model based on system identification cannot be estimated, since no voltage measurements are available. No information on the type of controller was available to us either. As a consequence, we used a PID controller for the feedback loop. Based on the observed profile of the motor output we determined that the controlled variable is the angular velocity $\omega(t)$. Indeed, Figure 3 shows the motor's angular velocity¹ that is maintained at a constant value by the controller. To compute the parameters of the PID controller we estimated metrics corresponding to the transient component of the output (angular velocity), such as rise time and overshoot; metrics that are formulated in .

¹The angular velocity profile shown in the graph is similar but not exactly the observed one, due to proprietary information restrictions.

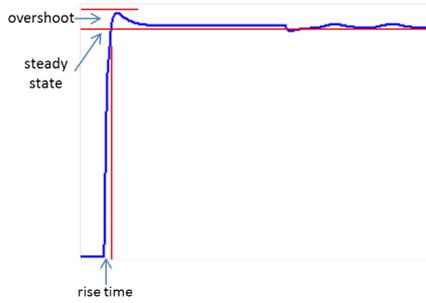


Figure 3: Motor angular velocity

The Gear-Cam mechanism

As mentioned earlier, the gear-cam mechanism amplifies the torque generated by the motor and transforms the rotational motion into a translational motion. The technical details provided to us confirmed only the presence of the cam, but not of the gear. We inferred the presence of the latter, by comparing the angular velocity of the motor with the cam's angular velocity, estimated from the measured cam's angle. This allowed us to estimate the ratio between the two velocities, and therefore estimate the gear ratio. The cam diagram is shown in Figure 4, where a wheel rotates as a result of the torque transmitted through the gear and acts on a lever that pushes the drive rod. Using the geometry of the cam,

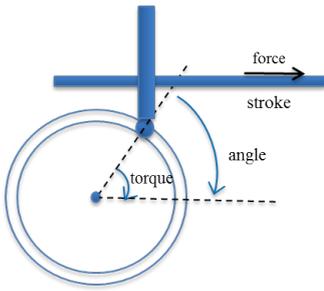


Figure 4: Cam schematics

the relation between the rotation motion and the linear motion (that is, the relation between the angle and the stroke) is given by

$$\text{stroke} = R \times \sin(\text{angle}),$$

where R denotes the radius of the cam. In addition, the map between the applied torque and the generated force is

$$\text{force} = \frac{1}{R} \times \text{torque} \times \cos(\text{angle}).$$

As both the cam angle and the stroke were included in the available measurements, we used a least square method to estimate the radius of the cam.

3.2 Adjuster

The adjuster links the drive rod connected to the point machine to the switch blades, and hence it is responsible for transferring the translational motion. There is a delay between the time instants the drive rod and the switch blades start moving. This delay is controlled by setting the positions of two bolts on the drive rod. Tighter bolt setting means a smaller delay, while looser bolt setting produce a larger delay. The high level diagram of the adjuster is depicted in Figure 5. The most challenging part in construct-

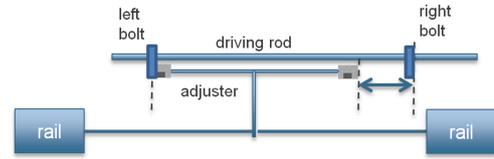


Figure 5: Adjuster diagram

ing the adjuster was modeling the non-sticking contact between the drive rod and the adjuster extremes. Stiff contact two bodies is usually modeled using a spring-damper component with very large values for the elasticity and damping constants. However, under this approach once contact takes place, it is permanent. To solve this challenge, we built a custom component that models the non-sticking contact.

3.3 Switch blades

The adjuster is connected to two switch blades that are moved from left to right or right to left, depending on the traffic needs. We look at a switch blade as a flexible body and used an approximation method to modeling beams, namely the lumped parameter approximation. This method assumes that beam deflection is small and in the linear regime. The lumped parameter approach approximates a flexible body as a set of rigid bodies coupled with springs and dampers. It can be implemented by a chain of alternating bodies and joints. The springs and dampers act on the bodies or the joints. The spring stiffness and damping coefficients are functions of the material properties and the geometry of the flexible elements. Parameters such a rail length, mass and mass moment of inertia were provided to us through technical documentation. To model the effect of the rail moving on rolling bearings, we included a friction component that accounts for energy loss due to friction. Although the component can model different friction models, the default models is Coulomb friction.

3.4 Fault augmentation

In this section we describe the modeling artifacts that were used to include in the behavior of the system the four fault operating modes: loose lock-pin, misaligned adjuster bolts, obstacle and missing bearings.

Loose lock-pin

The lock-pin referred in this fault mode connects the point machine with the drive rod that transfers the motion to the switch blades. More precisely, it locks the drive rod to the point machine. When this lock-pin becomes loose due to wear, it introduces a slackness in the way the motion is transferred to the switch blades. The lock-pin fault affects stability the connection point between the drive rod and the point machine. In time, if not fixed, this can lead to a complete failure of the pin, and therefore the point-machine cannot longer act upon the blades. A custom-built component whose main characteristic is that it implements a non-sticking pushing and pulling between two rods was built to model the effects of this fault. The impact between the two rods is assumed to be elastic, that is, we use a spring-damper assembly with large values for their parameters to model the contact. There are two types of contact: contact of the rods with the boundaries of the locking mechanism and contact between the rods. Both these types of contact must exhibit non-sticking pushing and pulling properties.

Misaligned adjuster bolts

In this fault mode the bolts of the adjuster deviate from their nominal position. As a result, the instant at which the drive rod meets the adjuster (and therefore the instant at which the the switch rail starts moving) happens either earlier or later. For example in a left-to-right motion, if the left bolt moves to the right, the contact happens earlier. The reason is that since the distance between the two bolts decreases, the left bolt reaches the adjuster faster. As a result, when the drive rod reaches its final position, there may be a gap between the right switch blade and the right stock rail. In contrast, if the left bolt moves to the left the contact happens later. The model of the adjuster includes parameters that can set the positions of the bolts, and therefore the effects of this fault mode can be modeled without difficulty.

Obstacle

In this fault mode, an obstacle prevents the switch blades reach their final nominal position, and therefore a gap between the switch blades and the stock rail appears. The effect on the motor torque is a sudden increase in value, as the motor tries to overcome the obstacle. To model this fault we included a component that implements a *hard stop* for the position of the switch blades. This component has two parameters for setting the left and right limits within motion of the switch blades is allowed. By changing the values of these parameters, the presence of an obstacle can be simulated.

Missing bearings

To minimize friction, the rails are supported by a set of rolling bearings. When they become stuck or lost, the energy losses due to friction increase. As mentioned in the section describing the switch blades modeling, a component was included to account for friction. This component has a parameter that sets the value for the friction coefficient. By increasing the value of this parameter, the effect of the missing bearings fault can be simulated.

4 Model Validation

Motor angular velocity, cam angle and stroke, together with the motor torque were used in the validation process. To these measurements, we added the rail position that was estimated from a set of movies depicting the rail motion, to which image processing techniques were applied. We achieved partial validation of the model. The simulated motor angular velocity, cam angle and stroke closely match the measured data. The simulated motor torque however matches in a qualitative sense its measured counterpart. The main reason is the fact that we had to make assumptions on the type controller motor and controller, without no way to validate these assumptions. In addition, the available measurements did not allow for the estimating the parameters in the assumed models, as this problem is ill posed. Figure 6 depicts the simulated torque, emphasizing the five operating zone. In Zone 1, the motor rotates the cam and the drive rod moves freely. No contact with the switch blades takes place in this zone, and the (small) energy loss is due to friction in the mechanical components. Zone 2 corresponds to the case where the drive rod pushes the two switch blades. The elasticity in the switch blades can be noticed in the torque profile in this zone. In Zone 3, the switch blades accelerate (as they drop off the rolling bearings) and again the drive rod moves freely (note the drop in torque). Zone 4 depicts the case

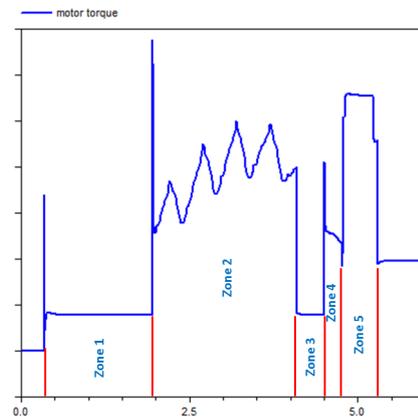


Figure 6: Motor torque with its five operating zones

where the drive rod catches up again with switch blades and pushes them to their final position. Finally, in Zone 5 the switch blades are pushed against the stock rails for a short period of time, hence the increase in torque. In support of the validation of these five operating zone, a set of movies depicting the motion of the switch blades were used. With respect to the fault operating modes, we managed to generate similar effects in the simulated data, as the ones observed in the measured data. Figure 7 shows the effect of the misaligned bolts fault, and in particular the case where the left bolt moves to the left. The effect is a delay applied on the time instant the drive rod reaches the switch blades. In addition, Zone 5 is also affected since due to the decreased distance, the switch blades are no longer pushed against the stock rails. In the case of an obstacle, the switch blades (and

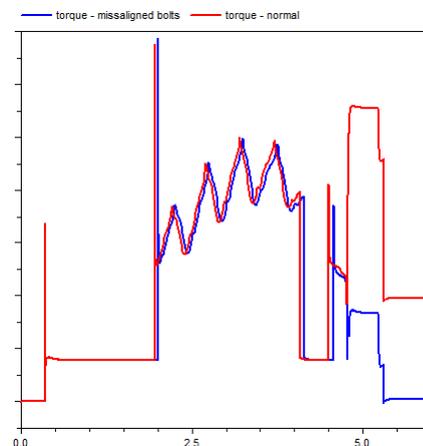


Figure 7: Motor torque in the normal and misaligned bolts fault modes

hence the drive rod) push against an obstacle that does not allow the completion of the motion. Therefore, the electric motor develops the maximum allowable torque as seen in Figure 8. In the case of the missing bearing fault mode, the motion friction of the switch blades increases, and hence the torque generated by the motor must accommodate this increase. We obtained this effect in simulation as shown in Figure 9. Finally, Figure 10 shows the effects of the lock-pin fault. The slackness introduced by the looseness of the pin induces a delay in the rail motion which also affects the behavior in Zone 5. In terms of the changes in the five op-

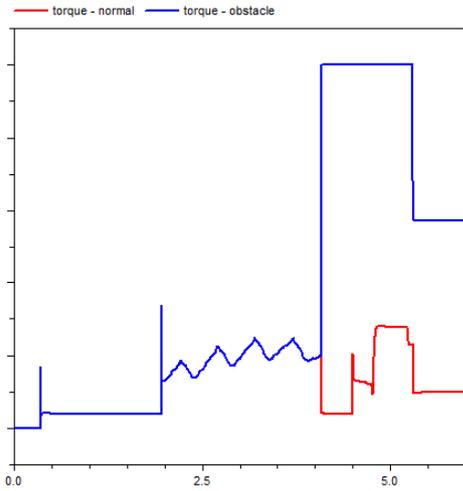


Figure 8: Motor torque in the normal and obstacle fault modes

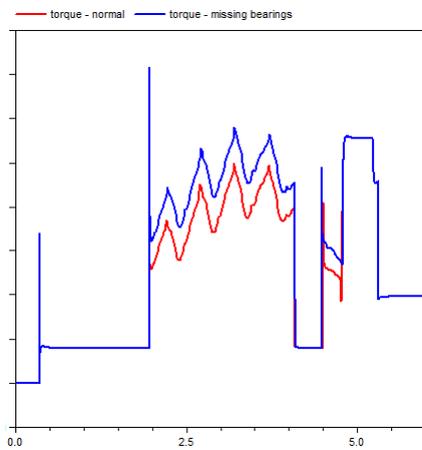


Figure 9: Motor torque in the normal and missing bearings fault modes

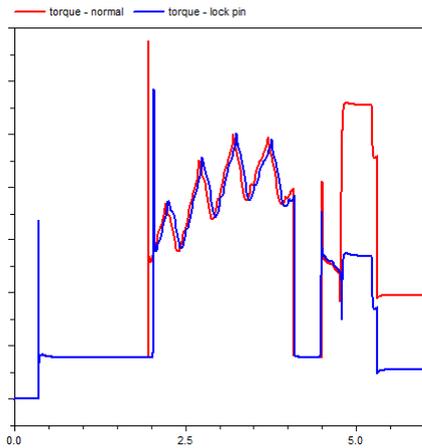


Figure 10: Motor torque in the normal and lock-pin fault modes

erating zones, the simulated behavior showed similar characteristics as in the case of the real data. The understanding of these behaviors come as a result of building the model, augmenting the model with fault modes, and analyzing their

effects in simulation. The choice of features described in the next section was supported by this understanding.

5 Fault Detection and Diagnosis

In the case of a railway switch, our measurements include the motor torque and motor angular velocity. As the switch moves from one extreme position to the other, these quantities are measured at a fixed sampling rate. Thus, we obtain a time series for each of the measurements. Let $\{\tau(t_1), \dots, \tau(t_N)\}$ denote torque measured at time instants $\{t_1, \dots, t_N\}$. Likewise, let $\{\omega(t_1), \dots, \omega(t_N)\}$ denote the angular velocity. For simplicity's sake, we denote the two time series of measurements by X . The diagnosis objective is to determine the underlying condition of the system from these time series. In other words, the objective is to determine a classifier $f : X \rightarrow \{N, F_1, F_2, F_3, F_4, F_5\}$, where N refers to the class label corresponding to the normal condition and F_1, F_2, F_3 and F_4 denote the class labels loose bolt, tight bolt, loose lock-pin, missing bearings, and obstacle respectively.

We adopt a machine learning approach to constructing the above mentioned classifier. The two main steps in building a machine learning classifier are feature selection and classifier type selection. These two steps are discussed next.

5.1 Feature selection

As seen in Figure 6, the motor torque profile shows five distinct operating zones. Moreover, we notice from Figures 7, 8, 9 and 10 that a given fault's impact on the torque profile seems limited to only some of the five zones. With this observation, our feature selection strategy is as follows.

1. Identify the approximate time instants that define the boundaries of the five zones. For example, Zone 1 is defined to be between times 0.8 seconds and 2 seconds, zone 2 is defined to be between times 2 seconds and 4.1 seconds, and so on.
2. Within each zone, compute a set of measures. An example of a measure is the total energy dissipated within the zone. This is computed as instantaneous power integrated over the duration of the zone. The instantaneous power is the product of instantaneous torque and angular velocity. Other examples of features include maximum and minimum torque values within the zone. The disclosure of the full set of measures used is not possible at this time for proprietary reasons. The features are normalized to have zero mean and unit standard deviation.

Note that it might be possible to combine one or more zones into one for feature selection.

5.2 Classifier selection

To map the features to the classes, $\{N, F_1, F_2, F_3, F_4, F_5\}$, we use machine learning. Examples of types of classifiers commonly used include k -nearest neighbors, support vector machines, neural networks and decision trees. We chose Random Forest, an ensemble classifier, because of its robustness to overfitting. For a more detailed discussion on the advantages of Random Forest, we refer the reader to [Breiman, 2001]. In addition, we also developed a binary classifier for fault detection based on Alternating Decision Tree (AD Tree). The advantage of AD Tree is that the results are human interpretable.

5.3 Results

For each fault type, we introduce varying magnitudes of fault and simulate the switch model described earlier. The fault magnitude is parameterized by a factor k which is varied over a pre specified range. A value of k equal to zero corresponds to normal case. Higher values of k correspond to the faulty cases. In addition, we also add representative noise to the measurements. Figure 11 shows some example torque profiles generated by the simulation.

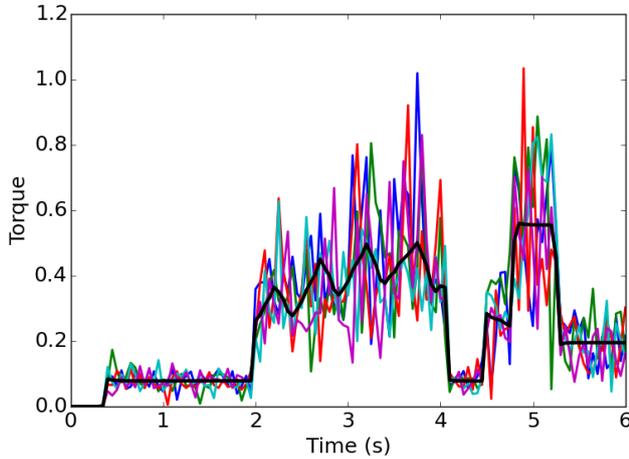


Figure 11: Simulated torque measurements with added noise.

The data generated is recorded and used to train and test the machine learning classifier. We use leave-one-out cross-validation for training and testing the classifiers. In this approach, one data sample is used for testing whereas all the rest of the data is used for training. This is repeated until each data sample has been tested once. Table 1 shows the confusion matrix for the simulated data described earlier. The (i, j) th entry of the confusion matrix refers to the percentage of cases where the true class was i but was classified as j by the classifier. A matrix with 100 along all the diagonal entries would correspond to a perfect classifier. In the results shown in Table 1, we observe some misclassification between classes N and F_4 . Recall that N is the normal class and F_4 is the missing bearing class. On further investigation, we determined that the misclassification occurs between the normal data and data corresponding to *low magnitudes* of the missing bearing fault.

Table 1: Fault diagnosis confusion matrix on simulated data

	N	F_1	F_2	F_3	F_4	F_5
N	97.2	0	0	2	0.8	0
F_1	0	100	0	0	0	0
F_2	0	0	99	1	0	0
F_3	9	0	4	87	0	0
F_4	11	0	0	0	89	0
F_5	0	0	0	0	0	100

The binary classification or fault detection result using AD Tree is shown in Table 2. As in the multi-class classification case, the false positives (normal classified as abnormal), and false negatives (abnormal classified as normal) are

primarily due to confusion between missing bearings and normal. Figure 12 shows part of the fault detection AD Tree. A pink oval represents a feature node. Depending on the value of the feature, one of two branches is followed until a leaf node is reached. Each edge that is traversed results in a score shown within the blue rectangles. For every root to leaf traversal, the total score is the sum of the scores accumulated on each edge. For a given data sample, multiple root to leaf paths may be traversed. In that case, the final score is the sum of the scores accumulated over all the paths. If the final score is negative, the decision is normal; otherwise the decision is abnormal.

Table 2: Fault detection confusion matrix on simulated data

	Normal	Abnormal
Normal	94.6	5.4
Abnormal	9.6	90.4

Next, we test the classifiers on real data. A key preprocessing step is to compute a linear transformation that transforms the mean and standard deviation of the features of the nominal (normal) real data to make them equal to the mean and standard deviation of the features of the nominal simulated data. The same transformation is then applied on the real faulty data before testing with the ML classifier. We emphasize here that to compute the transformation we only require examples of real data showing normal behavior. We do not use any real fault data for training the ML classifier. Table 3 shows the fault detection results on real data. As can be seen, we achieve a high accuracy of greater than 80 percent. We also tested the multi-class random forest classifier to diagnose the various faults. We were able to diagnose correctly all missing bearing faults but were unable to correctly diagnose the other faults.

Table 3: Fault detection confusion matrix on real data

	Normal	Abnormal
Normal	85.5	14.5
Abnormal	20	80

6 Related Work

A malfunctioning railway switch assembly can have a high impact on the railway transportation safety, and therefore the problem of diagnosing such systems has been addressed in other works. [Zattoni, 2006] proposes a detection system based on off-line processing of the armature current and voltage. The system implements an algorithm that realizes a finite impulse response system designed on the basis of an H_2 -norm criterion, and allows for detection of incremental faults (e.g., loss of lubrication, increasing obstructions, etc.). The approach hinges on the availability of a validated model of the point machine, which was not the case in our setup. [Zhou *et al.*, 2001; 2002] propose a remote monitoring system for railway point machines. The system includes a variety of sensors for acquiring trackside data related to parameters such as, distance, driving force, voltage, electrical noise, or temperature. The monitoring system logs data for offline analysis that offers detailed information on the condition of the system in the form of event

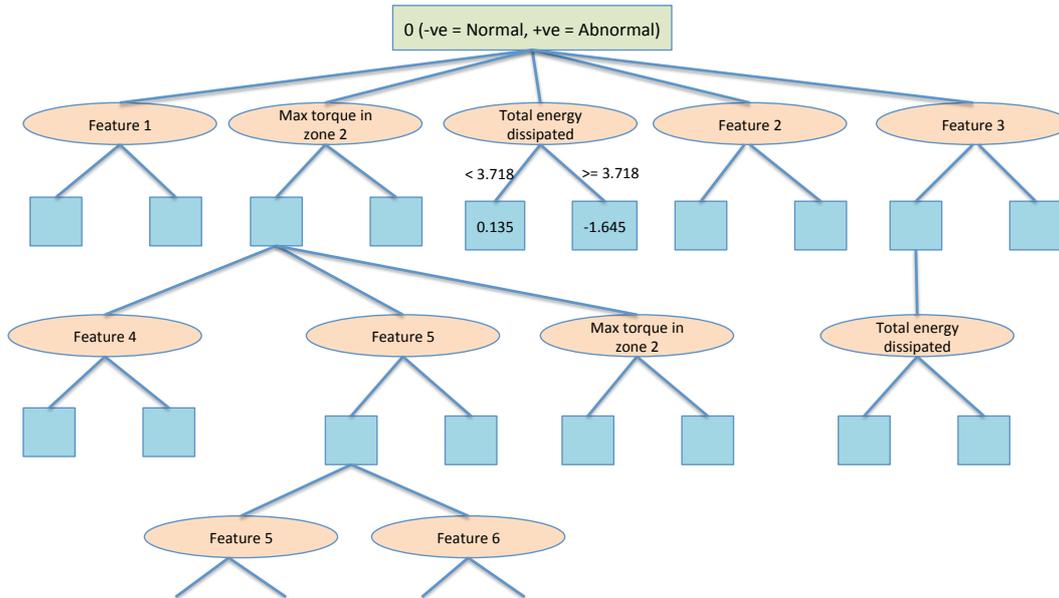


Figure 12: Part of the fault detection AD Tree

analysis and data trends. Hence unlike in our setup, the focus is on detection rather than isolation. In addition, due to scalability constraints, our solution is based on the embedded sensors, no other sensor being added. In [Asada *et al.*, 2013] classification based fault detection and diagnosis algorithm is developed using measurements such as drive force, electrical current and voltage. In particular, a classifier based on support vector machines is used. Our work also uses classification for diagnosis, but considers a wider variety of classifiers such as Multiclass Random Forest or Logitboosted Random Forest that were proved to be more robust [Opitz and Maclin, 1999]. The classification step in [Asada *et al.*, 2013] depends on a set of features extracted by applying the discrete wavelet transform on the active power. This step is oblivious on the operating modes of the point machine, which we showed to be relevant in our case. Hence, the diagnosis approach in [Asada *et al.*, 2013] is purely data driven. Since we had no access to current and voltage measurements this avenue for feature construction was not available to us. Depending of the type of electrical motors, the current and the voltage could be computed from the angular velocity and torque, respectively. However, knowledge of motor parameters is needed. [Asada *et al.*, 2013] consider two type of faults: underdriving and overdriving of the drive rod. Overdriving refers to the case where the switch blades are pushed against the stock rails due to misalignment, and a higher force then normal appears between the stock rails and the switch blades. Overdriving map to misaligned bolts, missing bearings and obstacles in our setup. All these fault modes exhibit higher forces than normal. Underdriving maps to a particular instance of the misaligned bolts fault (left bolt moves to the left for example). Therefore, our solution differentiate between more possible causes of higher forces since we take advantage of the particular signature these forces have in each fault corresponding to overdriving. Another pure data-driven approach for railway point machine monitoring was proposed in [Oyebande and Renfrew, 2002], where a net energy analysis technique was used to discriminate between

normal and abnormal behavior. This approach relies on a set of sensors measurements such as motors, voltage, current or switch blade positions, not all of them being available in our case. In addition, the computation of the net energy requires parameters of the electrical motor (armature resistance and motor shaft inertia) that again are not available in our setup. In addition, unlike our diagnosis objective, the focus in on detecting abnormalities within the point machine.

7 Conclusions

The three main general approaches to developing diagnostic software (FDI, MBR, and ML) all have severe limitations in many real-world applications. We believe we will see many more hybrid approaches to diagnosis that include the best of these three approaches to build accurate diagnosers. The railway switch is a critical and complex piece of equipment requiring extremely high diagnostic accuracy (the main reason this project was initiated), and the approach outlined in this paper was ultimately successful. Ultimately deployment of this approach will depend on expanding the set of faults detecting and on installation of more sensor rich switches in railroad infrastructures.

References

- [Asada *et al.*, 2013] T. Asada, C. Roberts, and T. Koseki. An algorithm for improved performance of railway condition monitoring equipment: Alternating-current point machine case study. *Transportation Research Part C: Emerging Technologies*, 30(0):81 – 92, 2013.
- [Breiman, 2001] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [de Kleer *et al.*, 1992] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing diagnoses and systems. 56(2-3):197–222, 1992.
- [Gertler, 1998] J. Gertler. *Fault-Detection and Diagnosis in Engineering Systems*. New York: Marcel Dekker, 1998.

- [Isermann, 1997] R. Isermann. Supervision, fault-detection and fault-diagnosis methods - An introduction. *Control Engineering Practice*, 5(5):639 – 652, 1997.
- [Isermann, 2005] Rolf Isermann. Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, 29(1):71 – 85, 2005.
- [Minhas *et al.*, 2014] R. Minhas, J. de Kleer, I. Matei, B. Saha, B. Janssen, D.G. Bobrow, and T Kortuglu. Using fault augmented Modelica model for diagnostics. In *Proceedings of the 10th International Modelica Conference*, Dec 2014.
- [Opitz and Maclin, 1999] David Opitz and Richard Maclin. Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
- [Oyebande and Renfrew, 2002] B.O. Oyebande and A.C. Renfrew. Condition monitoring of railway electric point machines. *Electric Power Applications, IEE Proceedings* -, 149(6):465–473, Nov 2002.
- [Patton *et al.*, 2000] Ron J. Patton, Paul M. Frank, and Robert N. Clark. *Issues of Fault Diagnosis for Dynamic Systems*. Springer-Verlag London, 2000.
- [Tiller, 2001] Michael Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.
- [Zattoni, 2006] Elena Zattoni. Detection of incipient failures by using an ℓ_1 -norm criterion: Application to railway switching points. *Control Engineering Practice*, 14(8):885 – 895, 2006.
- [Zhou *et al.*, 2001] F. Zhou, M. Duta, M. Henry, S. Baker, and C. Burton. Condition monitoring and validation of railway point machines. In *Intelligent and Self-Validating Instruments – Sensors and Actuators (Ref. No. 2001/179)*, IEE Seminar on, pages 6/1–6/7, Dec 2001.
- [Zhou *et al.*, 2002] F.B. Zhou, M.D. Duta, M.P. Henry, S. Baker, and C. Burton. Remote condition monitoring for railway point machine. In *Railroad Conference, 2002 ASME/IEEE Joint*, pages 103–108, April 2002.

Design of PD Observer-Based Fault Estimator Using a Descriptor Approach

Dušan Krokavec, Anna Filasová, Pavol Liščinský, Vladimír Serbák

Department of Cybernetics and Artificial Intelligence,

Technical University of Košice, Faculty of Electrical Engineering and Informatics,
Košice, Slovakia

e-mail: {dusan.krokavec, anna.filasova, pavel.liscinsky, vladimir.serbak}@tuke.sk

Abstract

A generalized principle of PD faults observer design for continuous-time linear MIMO systems is presented in the paper. The problem addressed is formulated as a descriptor system approach to PD fault observers design, implying the asymptotic convergence both the state observer error as fault estimate error. Presented in the sense of the second Lyapunov method, an associated structure of linear matrix inequalities is outlined to possess the observer asymptotic dynamic properties. The proposed design conditions are verified by simulations in the numerical illustrative example.

1 Introduction

As is well known, observer design is a hot research field owing to its particular importance in observer-based control, residual fault detection and fault estimation [1], where, especially from the stand point of the active fault tolerant control (FTC) structures, the problem of simultaneous state and fault estimation is very eligible. In that sense various effective methods have been developed to take into account the faults effect on control structure reconfiguration and fault detection [16], [22]. The fault detection filters, usually relying on the use of particular type of state observers, are mostly used to produce fault residuals in FTC. Because it is generally not possible in residuals to decouple totally fault effects from the perturbation influence, different approaches are used to tackle in part this conflict and to create residuals that are as a rule zero in the fault free case, maximally sensitive to faults, as well as robust to disturbances [2], [8]. Since faults are detected usually by setting a threshold on the generated residual signal, determination of an actual threshold is often formulated in adaptive frames [3]. Generalized method to solve the problem of actuator faults detection and isolation in over-actuated systems is given in [14], [15].

To estimate actuator faults for the linear time invariant systems without external disturbance the principles based on adaptive observers are frequently used, which make estimation of actuator faults by integrating the system output errors [25]. In particular, proportional-derivative (PD) observers introduce a design freedom giving an opportunity for generating state and fault estimates with good sensitivity properties and improving the observer design performance [6], [18], [19]. Since derivatives of the system outputs can be exploited in the fault estimator design to achieve faster

fault estimation, a proportional multi-integral derivative estimators are proposed in [7], [24].

Although the state observers for linear and nonlinear systems received considerable attention, the descriptor design principles have not been studied extensively for non-singular systems. Modifying the descriptor observer design principle [13], the first result giving sufficient design conditions, but for linear time-delay systems, can be found in [5]. Reflecting the same problems concerning the observers for descriptor systems, linear matrix inequality (LMI) methods were presented e.g. in [9] but a hint of this method can be found in [23], [25]. The extension for a class of nonlinear systems which can be described by Takagi-Sugeno models is presented in [12].

Adapting the approach to the observer-based fault estimation for descriptor systems as well as its potential extension, the main issue of this paper is to apply the descriptor principle in PD fault observer design. Preferring LMI formulation, the stability condition proofs use standard arguments in the sense of Lyapunov principle for the design conditions requiring to solve only LMIs without additional constraints. This presents a method designing the PD observation derivative and proportional gain matrices such that the design is non-singular and ensures that the estimation error dynamics has asymptotical convergence. From viewpoint of application, although the descriptor principle is used, it is not necessary to transform the system parameter into a descriptor form or to use matrix inversions in design task formulation. Despite a partly conservative form, the design conditions can be transformed to LMIs with minimal number of symmetric LMI variables.

The paper is organized as follows. Placed after Introduction, Sec. 2 gives a basic description of the PD fault observer and Sec. 3 presents design problem formulation in the descriptor form for a standard Luenberger observer. A new LMI structure, describing the PD fault observer design conditions, is theoretically explained in Sec 4. An example is provided to demonstrate the proposed approach in Sec. 5 and Sec. 6 draws some conclusions.

Used notations are conventional so that x^T , X^T denote transpose of the vector x and matrix X , respectively, $X = X^T > 0$ means that X is a symmetric positive definite matrix, $\|X\|_\infty$ designs the H_∞ norm of the matrix X , the symbol I_n represents the n -th order unit matrix, $\rho(X)$ and $rank(X)$ indicate the eigenvalue spectrum and rank of a square matrix X , \mathbb{R} denotes the set of real numbers and \mathbb{R}^n , $\mathbb{R}^{n \times r}$ refer to the set of all n -dimensional real vectors and $n \times r$ real matrices, respectively.

2 The Problem Statement

The systems under consideration are linear continuous-time dynamic systems represented in state-space form as

$$\dot{\mathbf{q}}(t) = \mathbf{A}\mathbf{q}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{F}\mathbf{f}(t), \quad (1)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{q}(t), \quad (2)$$

where $\mathbf{q}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^r$, $\mathbf{y}(t) \in \mathbb{R}^m$ are the vectors of the state, input and output variables, $\mathbf{f}(t) \in \mathbb{R}^p$ is the fault vector, $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times r}$, $\mathbf{C} \in \mathbb{R}^{m \times n}$ and $\mathbf{F} \in \mathbb{R}^{n \times p}$ are real finite values matrices, $m, r, p < n$ and

$$\text{rank} \begin{bmatrix} \mathbf{A} & \mathbf{F} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} = n + p. \quad (3)$$

It is considered that the fault $\mathbf{f}(t)$ may occur at an uncertain time, the size of the fault is unknown but bounded and that the pair (\mathbf{A}, \mathbf{C}) is observable.

Focusing on fault estimation task for slowly-varying faults, the fault PD observer is considered in the following form [19]

$$\begin{aligned} \dot{\mathbf{q}}_e(t) &= \mathbf{A}\mathbf{q}_e(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{F}\mathbf{f}_e(t) + \\ &+ \mathbf{J}(\mathbf{y}(t) - \mathbf{y}_e(t)) + \mathbf{L}(\dot{\mathbf{y}}(t) - \dot{\mathbf{y}}_e(t)), \end{aligned} \quad (4)$$

$$\mathbf{y}_e(t) = \mathbf{C}\mathbf{q}_e(t), \quad (5)$$

$$\dot{\mathbf{f}}_e(t) = \mathbf{M}(\mathbf{y}(t) - \mathbf{y}_e(t)) + \mathbf{N}(\dot{\mathbf{y}}(t) - \dot{\mathbf{y}}_e(t)), \quad (6)$$

where $\mathbf{q}_e(t) \in \mathbb{R}^n$, $\mathbf{y}_e(t) \in \mathbb{R}^m$, $\mathbf{f}_e(t) \in \mathbb{R}^p$ are estimates of the system states vector, the output variables vector and the fault vector, respectively, and $\mathbf{J}, \mathbf{L} \in \mathbb{R}^{n \times m}$, $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{p \times m}$ is the set of observer gain matrices is to be determined.

To explain and concretize the obtained results, the following well known lemma of Schur complement property is suitable.

Lemma 1. [20] *Considering the matrices $\mathbf{Q} = \mathbf{Q}^T$, $\mathbf{R} = \mathbf{R}^T$ and \mathbf{S} of appropriate dimensions, where $\det \mathbf{R} \neq 0$, then the following statements are equivalent*

$$\begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}^T & -\mathbf{R} \end{bmatrix} < 0 \Leftrightarrow \mathbf{Q} + \mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T < 0, \mathbf{R} > 0 \quad (7)$$

This shows that the above block matrix inequality has a solution if the implying set of inequalities has a solution.

3 Descriptor Principle in Luenberger Observer Design

To formulate the proposed PD observer design approach, the descriptor principle in the observer stability analysis is presented.

If the fault-free system (1), (2) is considered, the Luenberger observer is given as

$$\dot{\mathbf{q}}_e(t) = \mathbf{A}\mathbf{q}_e(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{J}(\mathbf{y}(t) - \mathbf{y}_e(t)), \quad (8)$$

$$\mathbf{y}_e(t) = \mathbf{C}\mathbf{q}_e(t), \quad (9)$$

and using (1), (2), (8), (9), it yields

$$\dot{\mathbf{e}}(t) = (\mathbf{A} - \mathbf{J}\mathbf{C})\mathbf{e}(t), \quad (10)$$

$$(\mathbf{A} - \mathbf{J}\mathbf{C})\mathbf{e}(t) - \dot{\mathbf{e}}(t) = \mathbf{0}, \quad (11)$$

respectively, where

$$\mathbf{e}_q(t) = \mathbf{q}(t) - \mathbf{q}_e(t). \quad (12)$$

Using the descriptor principle, the following lemma presents the Luenberger observer design conditions in terms of LMIs for the fault-free system (1), (2).

Lemma 2. *The Luenberger observer (8), (9) is stable if for given positive scalar $\delta \in \mathbb{R}$ there exist a symmetric positive definite matrix $\mathbf{P}_1 \in \mathbb{R}^{n \times n}$ a regular matrix $\mathbf{P}_3 \in \mathbb{R}^{n \times n}$ and a matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$ such that*

$$\mathbf{P}_1 = \mathbf{P}_1^T > 0, \quad (13)$$

$$\begin{bmatrix} \mathbf{A}^T\mathbf{P}_3 + \mathbf{P}_3^T\mathbf{A} - \mathbf{Y}\mathbf{C} - \mathbf{C}^T\mathbf{Y}^T & * \\ \mathbf{P}_1 - \mathbf{P}_3 + \delta\mathbf{P}_3^T\mathbf{A} - \delta\mathbf{Y}\mathbf{C} & -\delta(\mathbf{P}_3 + \mathbf{P}_3^T) \end{bmatrix} < 0. \quad (14)$$

When the above conditions hold, the observer gain matrix \mathbf{J} is given as

$$\mathbf{J} = (\mathbf{P}_3^T)^{-1}\mathbf{Y}. \quad (15)$$

Hereafter, * denotes the symmetric item in a symmetric matrix.

Proof. Denoting the observer system matrix as

$$\mathbf{A}_e = \mathbf{A} - \mathbf{J}\mathbf{C}, \quad (16)$$

then with the equality

$$\dot{\mathbf{e}}(t) = \dot{\mathbf{e}}(t) \quad (17)$$

the equivalent form of (11) can be written

$$\begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{e}}(t) \\ \ddot{\mathbf{e}}(t) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{e}}(t) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_n \\ \mathbf{A}_e & -\mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{e}(t) \\ \dot{\mathbf{e}}(t) \end{bmatrix}, \quad (18)$$

or, more generally,

$$\mathbf{E}^\diamond \dot{\mathbf{e}}^\diamond(t) = \mathbf{A}_e^\diamond \mathbf{e}^\diamond(t), \quad (19)$$

where

$$\mathbf{e}^{\diamond T}(t) = [\mathbf{e}^T(t) \quad \dot{\mathbf{e}}^T(t)], \quad (20)$$

$$\mathbf{E}^\diamond = \mathbf{E}^{\diamond T} = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{A}_e^\diamond = \begin{bmatrix} \mathbf{0} & \mathbf{I}_n \\ \mathbf{A}_e & -\mathbf{I}_n \end{bmatrix}. \quad (21)$$

Defining the Lyapunov function of the form

$$v(\mathbf{e}^\diamond(t)) = \mathbf{e}^{\diamond T}(t)\mathbf{E}^{\diamond T}\mathbf{P}^\diamond\mathbf{e}^\diamond(t) > 0, \quad (22)$$

where

$$\mathbf{E}^{\diamond T}\mathbf{P}^\diamond = \mathbf{P}^{\diamond T}\mathbf{E}^\diamond \geq 0, \quad (23)$$

then the derivative of (22) becomes

$$\begin{aligned} \dot{v}(\mathbf{e}^\diamond(t)) &= \\ &= \dot{\mathbf{e}}^{\diamond T}(t)\mathbf{E}^{\diamond T}\mathbf{P}^\diamond\mathbf{e}^\diamond(t) + \mathbf{e}^{\diamond T}(t)\mathbf{P}^{\diamond T}\mathbf{E}^\diamond\dot{\mathbf{e}}^\diamond(t) < 0 \end{aligned} \quad (24)$$

and, inserting (19) in (24), it yields

$$\dot{v}(\mathbf{e}^\diamond(t)) = \mathbf{e}^{\diamond T}(t)(\mathbf{P}^{\diamond T}\mathbf{A}_e^\diamond + \mathbf{A}_e^{\diamond T}\mathbf{P}^\diamond)\mathbf{e}^\diamond(t) < 0, \quad (25)$$

$$\mathbf{P}^{\diamond T}\mathbf{A}_e^\diamond + \mathbf{A}_e^{\diamond T}\mathbf{P}^\diamond < 0, \quad (26)$$

respectively. Introducing the matrix

$$\mathbf{P}^\diamond = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 \\ \mathbf{P}_3 & \mathbf{P}_4 \end{bmatrix}, \quad (27)$$

then, with respect to (23), it has to be

$$\begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 \\ \mathbf{P}_3 & \mathbf{P}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^T & \mathbf{P}_3^T \\ \mathbf{P}_2^T & \mathbf{P}_4^T \end{bmatrix} \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \geq 0, \quad (28)$$

which gives

$$\begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^T & \mathbf{0} \\ \mathbf{P}_2^T & \mathbf{0} \end{bmatrix} \geq 0. \quad (29)$$

It is evident that (29) can be satisfied only if

$$\mathbf{P}_1 = \mathbf{P}_1^T > 0, \quad \mathbf{P}_2 = \mathbf{P}_2^T = \mathbf{0}. \quad (30)$$

After simple algebraic operations so (26) can be transformed into the following form

$$\begin{bmatrix} \mathbf{A}_e^T \mathbf{P}_3 + \mathbf{P}_3^T \mathbf{A}_e & * \\ \mathbf{P}_1 - \mathbf{P}_3 + \mathbf{P}_4^T \mathbf{A}_e & -\mathbf{P}_4 - \mathbf{P}_4^T \end{bmatrix} < 0 \quad (31)$$

and, owing to emerged products $\mathbf{P}_3^T \mathbf{A}_e$, $\mathbf{P}_4^T \mathbf{A}_e$ in (31), the restriction on the structure of \mathbf{P}_4 can be enunciated as

$$\mathbf{P}_4 = \delta \mathbf{P}_3, \quad (32)$$

where $\delta > 0$, $\delta \in \mathbb{R}$. Since now

$$\mathbf{P}_4^T \mathbf{A}_e = \delta \mathbf{P}_3^T (\mathbf{A} - \mathbf{J}\mathbf{C}), \quad (33)$$

then, with the notation

$$\mathbf{Y} = \mathbf{P}_3^T \mathbf{J}, \quad (34)$$

(31) implies (14). This concludes the proof. \square

Remark 1. It is naturally to point out that the symmetrical form of Lemma 2, defined for $\mathbf{P}_1 = \mathbf{P}$, $\mathbf{P}_3 = \mathbf{P}_3^T = \mathbf{Q}$, is an equivalent inequality to the enhanced Lyapunov inequality for Luenberger observer design [11].

The above results can be generalized to formulate the descriptor principle in fault PD observer design. The main reason is to eliminate matrix inverse notations from the design conditions.

4 PD Observer Design

If the observer errors between the system state vector and the observer state vector as well as between the fault vector and the vector of its estimate are defined as follows

$$\mathbf{e}_q(t) = \mathbf{q}(t) - \mathbf{q}_e(t), \quad (35)$$

$$\mathbf{e}_f(t) = \mathbf{f}(t) - \mathbf{f}_e(t), \quad (36)$$

then, for slowly-varying faults, it is reasonable to consider [12]

$$\dot{\mathbf{e}}_f(t) = \mathbf{0} - \dot{\mathbf{f}}_e(t) = -\mathbf{M}\mathbf{C}\mathbf{e}_q(t) - \mathbf{N}\mathbf{C}\dot{\mathbf{e}}_q(t). \quad (37)$$

Note, since $\mathbf{f}_e(t)$ can be obtained as integral of $\dot{\mathbf{f}}_e(t)$, an adapting parameter matrix \mathbf{G} can be adjust interactively to set the amplitude of $\mathbf{f}_e(t)$, i.e., as results it is

$$\mathbf{f}_e(t) = \mathbf{G} \int_0^t \dot{\mathbf{f}}_e(\tau) d\tau. \quad (38)$$

To express the time derivative of the system state error $\mathbf{e}_q(t)$, the equations (1), (4) together with (2), (5) can be integrated as

$$\dot{\mathbf{e}}_q(t) = \mathbf{A}_e \mathbf{e}_q(t) + \mathbf{F} \mathbf{e}_f(t) - \mathbf{L}\mathbf{C}\dot{\mathbf{e}}_q(t), \quad (39)$$

where \mathbf{A}_e is given in (16) and the PD observer system matrix is

$$\mathbf{A}_{PDe} = (\mathbf{I}_n + \mathbf{L}\mathbf{C})^{-1} \mathbf{A}_e = (\mathbf{I}_n + \mathbf{L}\mathbf{C})^{-1} (\mathbf{A} - \mathbf{J}\mathbf{C}). \quad (40)$$

Since (37), (39) can be rewritten in the following composed form

$$\begin{bmatrix} \dot{\mathbf{e}}_q(t) \\ \dot{\mathbf{e}}_f(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_e & \mathbf{F} \\ -\mathbf{M}\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{e}_q(t) \\ \mathbf{e}_f(t) \end{bmatrix} - \begin{bmatrix} \mathbf{L}\mathbf{C} & \mathbf{0} \\ \mathbf{N}\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{e}}_q(t) \\ \dot{\mathbf{e}}_f(t) \end{bmatrix}, \quad (41)$$

by denoting

$$\mathbf{e}^{\circ T}(t) = [\mathbf{e}_q^T(t) \quad \mathbf{e}_f^T(t)], \quad (42)$$

$$\mathbf{A}^\circ = \begin{bmatrix} \mathbf{A} & \mathbf{F} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{J}^\circ = \begin{bmatrix} \mathbf{J} \\ \mathbf{M} \end{bmatrix}, \quad \mathbf{L}^\circ = \begin{bmatrix} \mathbf{L} \\ \mathbf{N} \end{bmatrix}, \quad (43)$$

$$\mathbf{I}^\circ = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_p \end{bmatrix}, \quad \mathbf{C}^\circ = [\mathbf{C} \quad \mathbf{0}], \quad (44)$$

where $\mathbf{A}^\circ, \mathbf{I}^\circ \in \mathbb{R}^{(n+p) \times (n+p)}$, $\mathbf{J}^\circ, \mathbf{L}^\circ \in \mathbb{R}^{(n+p) \times m}$, $\mathbf{C}^\circ \in \mathbb{R}^{m \times (n+p)}$, then the equation (41) can be written as

$$(\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ) \dot{\mathbf{e}}^\circ(t) = (\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ) \mathbf{e}^\circ(t), \quad (45)$$

$$\mathbf{A}_e^\circ \mathbf{e}^\circ(t) - \mathbf{D}_e^\circ \dot{\mathbf{e}}^\circ(t) = \mathbf{0}, \quad (46)$$

respectively, where

$$\mathbf{A}_e^\circ = \mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ, \quad \mathbf{D}_e^\circ = \mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ. \quad (47)$$

Introducing the equality

$$\dot{\mathbf{e}}^\circ(t) = \dot{\mathbf{e}}^\circ(t), \quad (48)$$

in analogy to the equation (18), then (48), (46) can be written as

$$\begin{bmatrix} \mathbf{I}^\circ & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{e}}^\circ(t) \\ \ddot{\mathbf{e}}^\circ(t) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{e}}^\circ(t) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}^\circ \\ \mathbf{A}_e^\circ & -\mathbf{D}_e^\circ \end{bmatrix} \begin{bmatrix} \mathbf{e}^\circ(t) \\ \dot{\mathbf{e}}^\circ(t) \end{bmatrix}. \quad (49)$$

Thus, by denoting

$$\mathbf{E}^\bullet = \begin{bmatrix} \mathbf{I}^\circ & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{A}_e^\bullet = \begin{bmatrix} \mathbf{0} & \mathbf{I}^\circ \\ \mathbf{A}_e^\circ & -\mathbf{D}_e^\circ \end{bmatrix}, \quad \mathbf{e}^\bullet(t) = \begin{bmatrix} \mathbf{e}^\circ(t) \\ \dot{\mathbf{e}}^\circ(t) \end{bmatrix}, \quad (50)$$

the obtained descriptor form to PD fault observer is

$$\mathbf{E}^\bullet \dot{\mathbf{e}}^\bullet(t) = \mathbf{A}_e^\bullet \mathbf{e}^\bullet(t), \quad (51)$$

where $\mathbf{A}_e^\bullet, \mathbf{E}^\bullet \in \mathbb{R}^{2(n+p) \times 2(n+p)}$.

The following solvability theorem is proposed to the design PD fault observer in the structure proposed in (4)-(6).

Theorem 1. *The PD fault observer (4)-(6) is stable if for given positive scalar $\delta \in \mathbb{R}$ there exist a symmetric positive definite matrix $\mathbf{P}_1^\circ \in \mathbb{R}^{(n+p) \times (n+p)}$, a regular matrix $\mathbf{P}_3^\circ \in \mathbb{R}^{(n+p) \times (n+p)}$ and matrices $\mathbf{Y}^\circ \in \mathbb{R}^{(n+p) \times m}$, $\mathbf{Z}^\circ \in \mathbb{R}^{(n+p) \times m}$ such that*

$$\mathbf{P}_1^\circ = \mathbf{P}_1^{\circ T} > 0, \quad (52)$$

$$\begin{bmatrix} \mathbf{A}^{\circ T} \mathbf{P}_3^\circ + \mathbf{P}_3^{\circ T} \mathbf{A}^\circ - \mathbf{Y}^\circ \mathbf{C}^\circ - \mathbf{C}^{\circ T} \mathbf{Y}^{\circ T} & * \\ \mathbf{V}_{21}^\circ & \mathbf{V}_{22}^\circ \end{bmatrix} < 0, \quad (53)$$

where

$$\mathbf{V}_{21}^\circ = \mathbf{P}_1^\circ - \mathbf{P}_3^\circ + \delta \mathbf{P}_3^{\circ T} \mathbf{A}^\circ - \delta \mathbf{Y}^\circ \mathbf{C}^\circ - \mathbf{C}^{\circ T} \mathbf{Z}^{\circ T}, \quad (54)$$

$$\mathbf{V}_{22}^\circ = -\delta \mathbf{P}_3^\circ - \delta \mathbf{P}_3^{\circ T} - \delta \mathbf{Z}^\circ \mathbf{C}^\circ - \delta \mathbf{C}^{\circ T} \mathbf{Z}^{\circ T}. \quad (55)$$

If the above conditions hold, the set of observer gain matrices is given by the equations

$$\mathbf{J}^\circ = (\mathbf{P}_3^{\circ T})^{-1} \mathbf{Y}^\circ, \quad \mathbf{L}^\circ = (\mathbf{P}_3^\circ)^{-1} \mathbf{Z}^\circ \quad (56)$$

and the matrices \mathbf{J} , \mathbf{L} , \mathbf{M} , \mathbf{N} can be separated with respect to (43).

Proof. Defining the Lyapunov function of the form

$$v(\mathbf{e}^\bullet(t)) = \mathbf{e}^{\bullet T}(t) \mathbf{E}^{\bullet T} \mathbf{P}^\bullet \mathbf{e}^\bullet(t) > 0, \quad (57)$$

where

$$\mathbf{E}^{\bullet T} \mathbf{P}^\bullet = \mathbf{P}^{\bullet T} \mathbf{E}^\bullet \geq 0, \quad (58)$$

then, using the property (58), the time derivative of (57) along the trajectory of (51) becomes

$$\begin{aligned} \dot{v}(\mathbf{e}^\bullet(t)) &= \\ &= \dot{\mathbf{e}}^{\bullet T}(t) \mathbf{E}^{\bullet T} \mathbf{P}^\bullet \mathbf{e}^\bullet(t) + \mathbf{e}^{\bullet T}(t) \mathbf{P}^{\bullet T} \mathbf{E}^\bullet \dot{\mathbf{e}}^\bullet(t) < 0. \end{aligned} \quad (59)$$

Thus, substituting (51) into (59), it yields

$$\dot{v}(\mathbf{e}^\bullet(t)) = \mathbf{e}^{\bullet T}(t) (\mathbf{P}^{\bullet T} \mathbf{A}_e^\bullet + \mathbf{A}_e^{\bullet T} \mathbf{P}^\bullet) \mathbf{e}^\bullet(t) < 0, \quad (60)$$

which implies

$$\mathbf{P}^{\bullet T} \mathbf{A}_e^\bullet + \mathbf{A}_e^{\bullet T} \mathbf{P}^\bullet < 0. \quad (61)$$

Defining the Lyapunov matrix

$$\mathbf{P}^\bullet = \begin{bmatrix} \mathbf{P}_1^\bullet & \mathbf{P}_2^\bullet \\ \mathbf{P}_3^\bullet & \mathbf{P}_4^\bullet \end{bmatrix}, \quad (62)$$

in analogy with (29) then (58) implies

$$\mathbf{P}_1^\bullet = \mathbf{P}_1^{\bullet T} > 0, \quad \mathbf{P}_2^\bullet = \mathbf{P}_2^{\bullet T} = 0 \quad (63)$$

and, using (50) and (62), (63) in (61), it yields

$$\begin{bmatrix} \mathbf{0} & \mathbf{A}_e^{\circ T} \\ \mathbf{I}^\circ & -\mathbf{D}_e^{\circ T} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1^\bullet & \mathbf{0} \\ \mathbf{P}_3^\bullet & \mathbf{P}_4^\bullet \end{bmatrix} + \begin{bmatrix} \mathbf{P}_1^\bullet & \mathbf{P}_3^{\circ T} \\ \mathbf{0} & \mathbf{P}_4^{\circ T} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{I}^\circ \\ \mathbf{A}_e^\circ & -\mathbf{D}_e^\circ \end{bmatrix} < 0. \quad (64)$$

After some algebraic manipulations, (64) takes the following form

$$\begin{bmatrix} \mathbf{U}_1^\bullet & \mathbf{U}_2^{\bullet T} \\ \mathbf{U}_2^\bullet & \mathbf{U}_3^\bullet \end{bmatrix} < 0, \quad (65)$$

where, with the notation (47),

$$\mathbf{U}_1^\bullet = (\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ)^T \mathbf{P}_3^\bullet + \mathbf{P}_3^{\circ T} (\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ), \quad (66)$$

$$\mathbf{U}_2^\bullet = \mathbf{P}_4^{\circ T} (\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ) + \mathbf{P}_1^\bullet - \mathbf{P}_3^\bullet - \mathbf{C}^{\circ T} \mathbf{L}^{\circ T} \mathbf{P}_3^\bullet, \quad (67)$$

$$\mathbf{U}_3^\bullet = -\mathbf{P}_4^\bullet - \mathbf{P}_4^{\circ T} - \mathbf{P}_4^{\circ T} \mathbf{L}^\circ \mathbf{C}^\circ - \mathbf{C}^{\circ T} \mathbf{L}^{\circ T} \mathbf{P}_4^\bullet. \quad (68)$$

By setting

$$\mathbf{P}_4^\bullet = \delta \mathbf{P}_3^\bullet, \quad \mathbf{Y}^\circ = \mathbf{P}_3^{\circ T} \mathbf{J}^\circ, \quad \mathbf{Z}^\circ = \mathbf{P}_3^{\circ T} \mathbf{L}^\circ, \quad (69)$$

where $\delta > 0$, $\delta \in \mathbb{R}$, then (65)-(68) imply (53)-(55).

Writing (68) as follows

$$\begin{aligned} \mathbf{U}_3^\bullet &= \\ &= -\mathbf{P}_4^{\circ T} (\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ) - (\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ)^T \mathbf{P}_4^\bullet = -\mathbf{R}^\bullet \end{aligned} \quad (70)$$

and comparing (7) and (65), then, if the inequalities (52)-(53) are satisfied, the Schur complement property (7) applied to (65) implies that \mathbf{R}^\bullet is positive definite.

Since \mathbf{P}_4^\bullet is regular, $(\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ)$ is also regular and so \mathbf{A}_{PD_e} given by (40) exists. This concludes the proof. \square

Since there is no restriction on the structure of \mathbf{P}_3 in Theorem 1, it follows that the problem of checking the existence of a stable system matrix of PD adaptive fault observer in a given matrix space may also be formulated with symmetric matrices \mathbf{P}_3 and \mathbf{P}_3 . This limit case of the LMI structure design condition, bound to a single symmetric matrix, is given by the following theorem.

Theorem 2. *The PD observer (4)-(6) is stable if for given positive scalar $\delta \in \mathbb{R}$ there exist a symmetric positive definite matrix $\mathbf{Q}^\circ \in \mathbb{R}^{(n+p) \times (n+p)}$ and matrices $\mathbf{Y}^\circ \in \mathbb{R}^{(n+p) \times m}$, $\mathbf{Z}^\circ \in \mathbb{R}^{(n+p) \times m}$ such that*

$$\mathbf{Q}^\circ = \mathbf{Q}^{\circ T} > 0, \quad (71)$$

$$\begin{bmatrix} \mathbf{A}^{\circ T} \mathbf{Q}^\circ + \mathbf{Q}^\circ \mathbf{A}^\circ - \mathbf{Y}^\circ \mathbf{C}^\circ - \mathbf{C}^{\circ T} \mathbf{Y}^{\circ T} & * \\ \mathbf{W}_{21}^\circ & \mathbf{W}_{22}^\circ \end{bmatrix} < 0, \quad (72)$$

where

$$\mathbf{W}_{21}^\circ = \delta \mathbf{Q}^\circ \mathbf{A}^\circ - \delta \mathbf{Y}^\circ \mathbf{C}^\circ - \mathbf{C}^{\circ T} \mathbf{Z}^{\circ T}, \quad (73)$$

$$\mathbf{W}_{22}^\circ = -2\delta \mathbf{Q}^\circ - \delta \mathbf{Z}^\circ \mathbf{C}^\circ - \delta \mathbf{C}^{\circ T} \mathbf{Z}^{\circ T}. \quad (74)$$

If the above conditions are affirmative, the extended observer gain matrices are given by the equations

$$\mathbf{J}^\circ = (\mathbf{Q}^\circ)^{-1} \mathbf{Y}^\circ, \quad \mathbf{L}^\circ = (\mathbf{Q}^\circ)^{-1} \mathbf{Z}^\circ. \quad (75)$$

Proof. Since there is no restriction on the structure of \mathbf{P}_3 it can be set

$$\mathbf{P}_1^\bullet = \mathbf{P}_3^\bullet = \mathbf{P}_3^{\circ T} = \mathbf{Q}^\circ > 0 \quad (76)$$

and the conditioned structure of \mathbf{P}_4^\bullet , with respect to \mathbf{P}_3^\bullet and \mathbf{A}_e^\bullet , can be taken into account as

$$\mathbf{P}_4^\bullet = \delta \mathbf{P}_3^\bullet = \delta \mathbf{Q}^\circ, \quad (77)$$

where $\delta > 0$, $\delta \in \mathbb{R}$. If these conditions are incorporated into (66)-(68), then

$$\mathbf{P}_3^T \mathbf{A}_e^\bullet = \mathbf{Q}^\circ (\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ) = \mathbf{Q}^\circ \mathbf{A}^\circ - \mathbf{Y}^\circ \mathbf{C}^\circ, \quad (78)$$

$$\mathbf{P}_4^{\circ T} \mathbf{L}^\circ \mathbf{C}^\circ = \delta \mathbf{P}_3^{\circ T} \mathbf{L}^\circ \mathbf{C}^\circ = \delta \mathbf{Q}^\circ \mathbf{L}^\circ \mathbf{C}^\circ = \delta \mathbf{Z}^\circ \mathbf{C}^\circ, \quad (79)$$

where

$$\mathbf{Y}^\circ = \mathbf{Q}^\circ \mathbf{J}^\circ, \quad \mathbf{Z}^\circ = \mathbf{Q}^\circ \mathbf{L}^\circ. \quad (80)$$

Thus, with these modifications, (65)-(68) imply (72)-(74). This concludes the proof. \square

Note, the design conditions formulated in Theorem 2 give potentially more conservative solutions.

5 Illustrative Example

The considered system is represented by the model (1), (2) with the model parameters [10]

$$\mathbf{A} = \begin{bmatrix} 1.380 & -0.208 & 6.715 & -5.676 \\ -0.581 & -4.290 & 0.000 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ 0.048 & 4.273 & 1.343 & -2.104 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.000 & 0.000 \\ 5.679 & 0.000 \\ 1.136 & -3.146 \\ 1.136 & 0.000 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To consider single actuator faults it was set $\mathbf{E} = \mathbf{B}$, and the matrix variables \mathbf{Q}° , \mathbf{Y}° , \mathbf{Z}° satisfying (71)-(74) for $\delta = 0.75$ were as follows

$$\begin{aligned} \mathbf{Q}^\circ &= [\mathbf{Q}_1^\circ \quad \mathbf{Q}_2^\circ], \\ \mathbf{Q}_1^\circ &= \begin{bmatrix} 0.1737 & 0.0012 & 0.1409 \\ 0.0012 & 0.1615 & 0.0195 \\ 0.1409 & 0.0195 & 0.1794 \\ -0.1316 & 0.0252 & -0.1439 \\ -0.0118 & -0.1975 & -0.0464 \\ 0.1461 & -0.0026 & 0.1557 \end{bmatrix}, \end{aligned}$$

$$\begin{aligned}
 \mathbf{Q}_2^\circ &= \begin{bmatrix} -0.1316 & -0.0118 & 0.1461 \\ 0.0252 & -0.1975 & -0.0026 \\ -0.1439 & -0.0464 & 0.1557 \\ 0.2177 & -0.1136 & -0.1255 \\ -0.1136 & 1.4490 & -0.1904 \\ -0.1255 & -0.1904 & 1.3479 \end{bmatrix}, \\
 \mathbf{Y}^\circ &= \begin{bmatrix} 0.1162 & -0.0220 \\ -0.0094 & 0.1404 \\ 0.0814 & 0.1439 \\ -0.0719 & 0.1072 \\ 0.0060 & 0.0171 \\ 0.0003 & 0.2159 \end{bmatrix}, \\
 \mathbf{Z}^\circ &= \begin{bmatrix} -0.0164 & -0.0445 \\ 0.0013 & -0.0528 \\ -0.0728 & 0.1181 \\ 0.0678 & 0.0229 \\ 0.0015 & 0.1434 \\ -0.1062 & 0.1758 \end{bmatrix},
 \end{aligned}$$

where the SeDuMi package [17] was used to solve given set of LMIs.

The PD observer extended matrix gains are then computed using (56) as

$$\begin{aligned}
 \mathbf{J}^\circ &= \begin{bmatrix} 0.8777 & -1.5720 \\ -0.0801 & 0.5621 \\ -0.0624 & 3.7385 \\ 0.1229 & 2.2486 \\ -0.0021 & 0.3934 \\ -0.0767 & 0.1649 \end{bmatrix}, \\
 \mathbf{L}^\circ &= \begin{bmatrix} 0.7391 & -2.0549 \\ 0.0663 & -0.6605 \\ -0.7915 & 3.4010 \\ 0.1994 & 1.3731 \\ -0.0000 & 0.2244 \\ -0.0488 & 0.1187 \end{bmatrix}.
 \end{aligned}$$

Verifying the PD observer system matrix eigenvalue spectrum, the results were

$$\rho(\mathbf{A}_e) = \{ -0.7731, -2.8914, -4.7816, -8.9188 \},$$

$$\rho(\mathbf{A}_{PDe}) = \{ -1.1194, -1.6912, -1.9969, -2.9765 \}.$$

That means the PD observer is stable as well as its "P" part is stable, too. Moreover, also the descriptor form (45) of the PD observer is stable, where

$$\begin{aligned}
 &\rho((\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ)^{-1}(\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ)) = \\
 &= \left\{ \begin{array}{l} -1.7763, -2.0966, \\ -0.6629 \pm 0.7872i, -1.3632 \pm 0.4931i \end{array} \right\}.
 \end{aligned}$$

Comparing with a solution of (52)-(55) for the $\delta = 0.95$, it is possible to verify that in this case

$$\rho(\mathbf{A}_e) = \{ -6.8230, -10.3876, -81.5789, -472.0230 \},$$

$$\rho(\mathbf{A}_{PDe}) = \{ -0.9562, -0.9774, -7.2561, -9.8300 \},$$

$$\begin{aligned}
 &\rho((\mathbf{I}^\circ + \mathbf{L}^\circ \mathbf{C}^\circ)^{-1}(\mathbf{A}^\circ - \mathbf{J}^\circ \mathbf{C}^\circ)) = \\
 &= \left\{ \begin{array}{l} -1.0240, -1.0748, -6.4810, -9.1501 \\ -0.9650 \pm 0.0068i \end{array} \right\},
 \end{aligned}$$

which implies in this case a faster dynamics of the descriptor form of the PD observer but a slower for the PD observer. Note, the exploitation of $\delta = 0.75$ leads in this case to unstable "P" part of the PD observer.

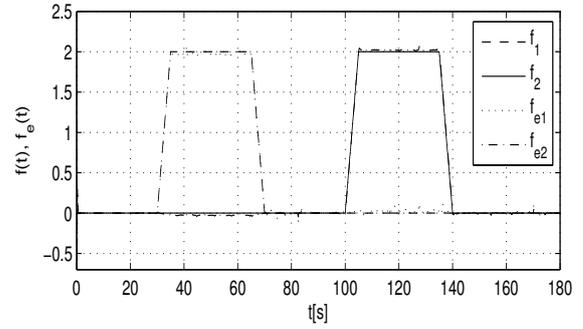


Figure 1: Adaptive fault estimator responses

Although many actuator faults can cause the gain to drift, in practice the faults lead to an abrupt change in gain [21]. To simulate this phenomena, it was assumed that the fault in actuators for (1) was given by

$$f(t) = \begin{cases} 0, & t \leq t_{sa}, \\ \frac{f_h}{t_{sb} - t_{sa}}(t - t_{sa}), & t_{sa} < t_{sb}, \\ f_h, & t_{sb} \leq t_{ca}, \\ -\frac{f_h}{t_{cb} - t_{ca}}(t - t_{cb}), & t_{ca} < t_{cb}, \\ 0, & t \geq t_{cb}, \end{cases}$$

where, analyzing the single first actuator fault estimation, it was set

$$f_h = 2, t_{sa} = 30s, t_{sb} = 35s, t_{ca} = 65s, t_{cb} = 70s,$$

and for the single second actuator fault these parameters were

$$f_h = 2, t_{sa} = 100s, t_{sb} = 105s, t_{ca} = 135s, t_{cb} = 140s.$$

It is demonstrated that for equal f_h in the first and the second actuator faults it is possible for given \mathbf{B} to adjust the common adapting parameter matrix \mathbf{G} in (38) as follows

$$\mathbf{G} = \begin{bmatrix} 40.0 & 5.9 \\ 5.9 & 22.0 \end{bmatrix}.$$

The obtained results are illustrated in Fig. 1 where, just in terms of rendering, all faults responses and their estimates were combined into a single image, and so the demonstration can not be seen as a progressive sequence of single faults in the actuators system. This figure presents the fault signals, as well as their estimations, reflecting the single first actuator fault starting at the time instant $t = 30s$ and applied for $40s$ and then the fault of the second actuator is demonstrated beginning in the time instant $t = 100s$ and lasts for $40s$. The presented simulation was carried out in the system autonomous mode, practically the same results were obtained for forced regime of the system.

The adapting parameter \mathbf{G} and the tuning parameter δ were set interactively considering the maximal value of fault signal amplitude f_h and the fault observer dynamics. It can be seen that there exist very small differences between the signals reflecting single actuator faults and the observer approximate ones for slowly varying piecewise constant actuator faults. The principle can be used directly in the control structures with the fault compensation [4], but can not be directly used to localize actuator faults [14].

6 Concluding Remarks

Based on the descriptor system approach a new PD fault observer design method for continuous-time linear systems and slowly-varying actuator faults is introduced in the paper. Presented version is derived in terms of optimization over LMI constraints using standard LMI numerical procedures to manipulate the fault observer stability and fault estimation dynamics. Presented in the sense of the second Lyapunov method expressed through LMI formulation, design conditions guaranty the asymptotic convergence of the state as well as fault estimation errors. The numerical simulation results show good estimation performances.

Acknowledgments

The work presented in this paper was supported by VEGA, the Grant Agency of the Ministry of Education and the Academy of Science of Slovak Republic, under Grant No. 1/0348/14. This support is very gratefully acknowledged.

References

- [1] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*, Springer-Verlag, Berlin, 2006.
- [2] W. Chen and M. Saif. Observer-based strategies for actuator fault detection, isolation and estimation for certain class of uncertain nonlinear systems. *IET Control Theory & Applications*, 1(6):1672-1680, 2007.
- [3] S. Ding. *Model-Based Fault Diagnosis Techniques. Design Schemes, Algorithms, and Tools*, Springer-Verlag, Berlin, 2008.
- [4] A. Filasová and D. Krokavec. Design principles of active robust fault tolerant control systems. *Robust Control. Theory and Applications*, A. Bartoszewicz (Ed.), InTech, Rijeca, 309-338, 2011.
- [5] E. Fridman and U. Shaked. A descriptor system approach to H_∞ control of linear time-delay systems. *IEEE Transactions on Automatic Control*, 47(2):253-270, 2002.
- [6] Z. Gao. PD observer parametrization design for descriptor systems. *Journal of the Franklin Institute*, 342(5):551-564, 2005.
- [7] Z. Gao and S.X. Ding. Fault estimation and fault-tolerant control for descriptor systems via proportional, multiple-integral and derivative observer design. *IET Control Theory & Applications*, 1(5):1208-1218, 2007.
- [8] J. Guo, X. Huang, and Y. Cui. Design and analysis of robust fault detection filter using LMI tools. *Computers and Mathematics with Applications*, 57(11-12):1743-1747, 2009.
- [9] K. Ilhem, J. Dalel, B.H.A. Saloua, and A.M. Naceur. Observer design for Takagi-Sugeno descriptor system with Lipschitz constraints. *International Journal of Instrumentation and Control Systems* 2(2):13-25, 2012.
- [10] J. Kautsky, N. K. Nichols, and P. Van Dooren. Robust pole assignment in linear state feedback. *International Journal of Control*, 41(5):1129-1155, 1985.
- [11] D. Krokavec and A. Filasová. On observer design methods for a class of Takagi-Sugeno fuzzy systems. *Proceedings of the 3rd International Conference on Control, Modelling, Computing and Applications CMCA 2014*, Dubai, UAE, 1-11, 2014.
- [12] D. Krokavec and A. Filasová. Design of PD observer-based fault estimator for a class of Takagi-Sugeno descriptor systems. *The 1st IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNOM-2015*, Saint-Petersburg, Russia, 2015. (in press)
- [13] G. Lu, D.W.C. Ho, and Y. Zheng. Observers for a class of descriptor systems with Lipschitz constraint. *Proceeding of the 2004 American Control Conference*, Boston, MA, USA, 3474-3479, 2004.
- [14] N. Meskin and K. Khorasani. Fault detection and isolation of actuator faults in overactuated systems. *Proceedings of the 2007 American Control Conference*, New York City, NY, USA, 2527-2532, 2007.
- [15] N. Meskin and K. Khorasani. Actuator fault detection and isolation for a network of unmanned vehicles. *IEEE Transactions on Automatic Control*, 54(4):835-840, 2009.
- [16] R.J. Patton and S. Klinkhieo. Actuator fault estimation and compensation based on an augmented state observer approach. *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, P.R. China, 8482-8487, 2009.
- [17] D. Peaucelle, D. Henrion, Y. Labit, and K. Taitz. *User's Guide for SeDuMi Interface*, LAAS-CNRS, Toulouse, France, 2002.
- [18] J. Ren and Q. Zhang. PD observer design for descriptor systems. An LMI approach. *International Journal of Control, Automation, and Systems*, 8(4):735-740, 2010.
- [19] F. Shi and R.J. Patton. Simultaneous state and fault estimation for descriptor systems using an augmented PD observer. *Preprints of 19th IFAC World Congress*, Cape Town, South Africa, 8006-8011, 2014.
- [20] J.G. VanAntwerp and R.D. Braatz. A tutorial on linear and bilinear matrix inequalities. *Journal of Process Control*, 10:363-385, 2000.
- [21] H. Wang and S. Daley. Actuator fault diagnosis. An adaptive observer-based technique. *IEEE Transactions on Automatic Control*, 41(7):1073-1078, 1996.
- [22] F. Zhang, G. Liu, and L. Fang. Actuator fault estimation based on adaptive H_∞ observer technique. *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*, Changchun, P.R. China, 352-357, 2009.
- [23] K. Zhang, B. Jiang, and V. Cocquemot. Adaptive observer-based fast fault estimation. *International Journal of Control, Automation, and Systems* 6(3): 320-26, 2008.
- [24] K. Zhang, B. Jiang, and P. Shi. Fast fault estimation and accommodation for dynamical systems. *IET Control Theory & Applications*, 3(2):189-199, 2009.
- [25] K. Zhang, B. Jiang, and P. Shi. *Observer-Based Fault Estimation and Accommodation for Dynamic Systems*, Springer-Verlag, Berlin, 2013.

Chronicle based alarm management in startup and shutdown stages

John W. Vasquez^{1,3,4}, Louise Travé-Massuyès^{1,2}, Audine Subias^{1,3}, Fernando Jimenez⁴ and Carlos Agudelo⁵

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

²Univ de Toulouse, LAAS, F-31400 Toulouse, France

³Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

⁴Universidad de los Andes, Colombia.

⁵ECOPETROL ICP, Colombia.

e-mail: jwvasque@laas.fr, louise@laas.fr, subias@laas.fr,
fjimenez@uniandes.edu.co, carlos.agudelo@ecopetrol.com.co

Abstract

The transitions between operational modes (startup/shutdown) in chemical processes generate alarm floods and cause critical alarm saturation. We propose in this paper an approach of alarm management based on a diagnosis process. This diagnosis step relies on situation recognition to provide to the operators relevant information on the failures inducing the alarms flows. The situation recognition is based on chronicle recognition. We propose to use the information issued from the modeling of the system to generate temporal runs from which the chronicles are extracted. An illustrative example in the field of petrochemical plants ends the article.

1 Introduction

The petrochemical industries losses have been estimated at 20 billion dollars only in the U.S. each year, and the AEM (Abnormal Events Management) has been classified as a problem that needs to be solved. Hence the alarm management is one of the aspects of great interest in the safety planning for the different plants. In the process state transitions such as startup and shutdown stages the alarm flood increases and it generates critical conditions in which the operator does not respond efficiently, then a dynamic alarm management is required [1]. Currently, many fault detection and diagnosis techniques for multimode processes have been proposed; however, these techniques cannot indicate fundamental faults in the basic alarm system [2], in the other hand the technical report "Advance Alarm System Requirements" EPRI (The Electric Power Research Institute) suggests a cause-consequence and event-based processing. In this perspective, diagnosis approaches based on complex events processing or situation recognition are interesting issues. Therefore, in this paper, a dynamic alarm management strategy is proposed in order to deal with alarm floods happening during transitions of chemical processes. This approach relies on situations recognition (i.e. chronicle recognition). As, the efficiency of alarm management approaches depends on the operator expertise and process knowledge, our final objective is to develop a diagnosis approach as a decision tool for operators. The paper is divided into 6 sections. Section 2 gives an overview on the relevant literature. The section 3 concerns the modeling of the system. The section 4 is about the chronicle principle and the temporal runs

used for the chronicle design. The section 5 is devoted to the chronicle generation. Finally, an illustrative application on real data from a petrochemical plant is given section 6.

2 State of the art: Alarm management

Alarm management has recently focused the attention of many researchers in themes such as:

Alarm historian visualization and analysis: A combined analysis of plant connectivity and alarm logs to reduce the number of alerts in an automation system was presented by [3]; the aim of the work presented is to reduce the number of alerts presented to the operator. If alarms are related to one another, those alarms should be grouped and presented as one alarm problem. Graphical tools for routine assessment of industrial alarm systems was proposed by [4], they presented two new alarm data visualization tools for the performance evaluation of the alarm systems, known as the high density alarm plot (HDAP) and the alarm similarity color map (ASCM). Event correlation analysis and two-layer cause-effect model were used to reduce the number of alarms in [5]. A Bayesian method has been introduced for multimode process monitoring in [6]. This type of techniques helps us to recognize alarm chattering, grouping many alarms or estimate the alarm limits in transition stages, but the time and the procedure actions are not included.

Process data-based alarm system analysis and rationalization: The evaluation of plant alarm systems by behavior simulation using a virtual subject was proposed by [7]. [8] introduced a technique for optimal design of alarm limits by analyzing the correlation between process variables and alarm variables. In 2009 a framework based on the receiver operating characteristic (ROC) curve was proposed to optimally design alarm limits, filters, dead bands, and delay timers; this work was presented in [9] and a dynamic risk analysis methodology that uses alarm databases to improve process safety and product quality was presented in [10]. In [11] the Gaussian mixture model was employed to extract a series of operating modes from the historical process data and then the local statistic and its normalized contribution chart were derived for detecting abnormalities early and for isolating faulty variables. We can see that the use of virtual subjects could be applied to probe the alarm system and using historical information about the alarm behavior for detecting abnormalities. The problem is presented when the simulation requires a lot time to probe the totally of scenarios and when we have new plants that do not contain information about historical data.

Plant connectivity and process variable causality analysis (causal methods): In the literature, transition monitoring of chemical processes has been reported by many researchers. In [12] was presented a fault diagnosis strategy for startup process based on standard operating procedures, this approach proposes behavior observer combined with dynamic PCA (Principal Component Analysis) to estimate process faults and operator errors at the same time, and in [13] was presented a framework for managing transitions in chemical plants where a trend analysis-based approach for locating and characterizing the modes and transitions in historical data is proposed. Finally, in [14] a hybrid model-based framework was used for alarm anticipation where the user can prepare for the possibility of a single alarm occurrence. For the transition monitoring, these types of techniques are the most used in industrial processes and the hybrid model based framework could be a good representation of our system. We can observe that a causal model allows identify the root of the failures and check the correct evolution in a transitional stage. Our proposal is closer to the third type of approach and seeks to exploit the causal relationships as presented in the next sections.

3 Representation of the system

3.1 Hybrid Causal Model

The hybrid system is represented by an extended transition system [15], whose discrete states represent the different modes of operation for which the continuous dynamics are characterized by a qualitative domain. Formally, a hybrid causal system is defined as a tuple:

$$\Gamma = (\vartheta, D, Conf, Tr, E, CSD, Init) \quad (1)$$

Where

- $\vartheta = \{v_i\}$ is a set of continuous *process variables* which are function of time t .
- D is a set of discrete variables. $D = Q \cup K \cup V_Q$. Q is a set of states q_i of the transition system which represent the system operation modes. The set of auxiliary discrete variables $K = \{K_i, i = 1, \dots, n_c\}$ represents the system configuration in each mode q_i as defined below by $Conf(q_i)$. $V_Q = \{V_i\}$ is a set of qualitative variables whose values are obtained from the behavior of each continuous variable v_i .
- $Conf(q_i): Q \rightarrow \otimes_i D(K_i)$ where \otimes is the Cartesian product and $D(K_i)$ is the domain of $K_i \in K$ that provides the configuration associated to the mode. i.e. the modes of the underlying multimode components (typically, a valve has two normal modes, *opened* and *closed*)
- $E = \Sigma \cup \Sigma^c$ is a finite set of event types noted σ , where:
 - Σ is the set of event type associated to the procedure actions in a startup or shutdown stages.
 - Σ^c is the set of event type associated to the behavior of the continuous process variables.
- $Tr: Q \times \Sigma \rightarrow Q$ is the transition function. The transition from mode q_i to mode q_j with associated event σ is noted (q_i, σ, q_j) or $q_i \xrightarrow{\sigma} q_j$. We assume that the model is deterministic, without loss of generality i.e. whenever $q_i \xrightarrow{\sigma} q_j$ and $q_i \xrightarrow{\sigma} q_k$ then $q_j = q_k$ for each $(q_i, q_j, q_k) \in Q^3$ and each $\sigma \in \Sigma$.

- $CSD \supseteq \bigcup_i CSD_i$ is the *Causal System Description* or the causal model used to represent the constraints underlying in the continuous dynamic of the hybrid system. Every CSD_i associated to a mode q_i , is given by a graph $(G_c = \vartheta \cup K, I)$. I is the set of influences where there is an edge $e(v_i, v_j) \in I$ from $v_i \in \vartheta$ to $v_j \in \vartheta$ if the variable v_i influences variable v_j . Then, the vertices represent the variables and the edges represent the influences between variables and for each edge exists an association with a component in the system. The set of components is noted as $COMP$.
- $Init$ is the initial condition of the hybrid system,

3.2 Qualitative abstraction of continuous behavior

In each mode of operation, variables evolve according to the corresponding dynamics. This evolution is represented with qualitative values. The domain $D(V_i)$ of a qualitative variable $V_i \in V_Q$ is obtained through the function $f_{qual}: D(v_i) \rightarrow D(V_i)$ that maps the continuous values of variable v_i to ranges defined by limit values (High H_i and Low L_i).

$$f(v_i)_{qual} = \begin{cases} V_i^H & \text{if } v_i \geq H_i \wedge \frac{dv_i}{dt} > 0 \\ V_i^M & \text{if } v_i < H_i \wedge \frac{dv_i}{dt} < 0 \\ & \vee \\ & v_i \geq L_i \wedge \frac{dv_i}{dt} > 0 \\ V_i^L & \text{if } v_i < L_i \wedge \frac{dv_i}{dt} < 0 \end{cases} \quad (2)$$

$\frac{dv_i}{dt} > 0$ represents that the continuous variable v_i is increasing and $\frac{dv_i}{dt} < 0$ that it is decreasing. The behavior of these qualitative variables is represented in Figure 1. by the graph $G_{V_i} = (V_Q, \Sigma^c, \gamma)$ where V_Q is the set of the possible qualitative states (V_i^L : Low, V_i^M : Medium, V_i^H : High) of the continuous variable v_i , Σ^c is the finite set of the events associate to the transitions and $\gamma: V_Q \times \Sigma^c \rightarrow V_Q$ is the transition function. The corresponding event generator is

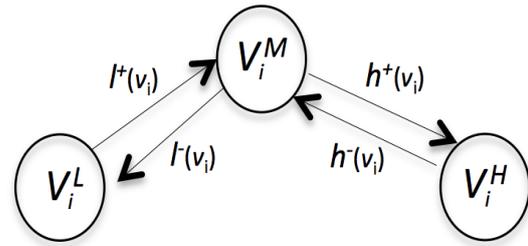


Figure 1: Qualitative values of the process variables

defined by the abstraction function $f_{V_Q \rightarrow \sigma}$

$$f_{V_Q \rightarrow \sigma}: V_Q \times \gamma(V_Q, \Sigma^c) \rightarrow \Sigma^c$$

$$\forall V_i \in V_Q, (V_i^n, V_i^m) \rightarrow \begin{cases} l^+(v_i) & \text{if } V_i^L \rightarrow V_i^M \\ l^-(v_i) & \text{if } V_i^M \rightarrow V_i^L \\ h^+(v_i) & \text{if } V_i^M \rightarrow V_i^H \\ h^-(v_i) & \text{if } V_i^H \rightarrow V_i^M \end{cases} \quad (3)$$

$$V_i^n, V_i^m \in \{V_i^L, V_i^M, V_i^H\}$$

$$\Sigma^c = \bigcup_{v_i \in \vartheta} \{l^+(v_i), l^-(v_i), h^+(v_i), h^-(v_i)\} \quad (4)$$

3.3 Automatic derivation of the causal model

To obtain the causal model of a system in a given operating mode implies to collect the equations that represent the behavior of the system in this mode. The theory of causal ordering issued from the Qualitative Reasoning community can be well applied to obtain automatically the causal structure associated to a set of equations. Now, associating activation conditions to the equations extend the causal ordering to systems with several operating modes [16]. Then these activation conditions can be related in the influences of the resulting causal graph. The proposed algorithm, implemented in the Causalito software makes use of conditions that avoid recomputing a totally new perfect matching for every operating mode, thus reducing the computational cost. In this work, the Causal System Description is given by $CSD = (\vartheta, I)$, where each influence I is labeled with:

- An activation condition indicating the modes in which it is active (or no label if it is active in all modes),
- The corresponding equation,
- The component whose behavior is expressed by the equation.

In the follow section we expose the principle of the chronicle generation where concepts such as *event*, *chronicle* and *temporal run* are described.

4 Chronicles

4.1 Events and chronicles

Let us consider time as a linearly ordered discrete set of instants. The occurrence of different events in time represents the system dynamics and a model can be determined to diagnose the correct evolution. An event is defined as a pair (σ_i, t_i) , where $\sigma_i \in E$ is an event type and t_i is a variable of integer type called the event date. We define E as the set of all event types and a temporal sequence on E is an ordered set of events denoted $S = \langle (\sigma_i, t_i) \rangle_j$ with $j \in \mathbb{N}_l$ where l is the size of the temporal sequence S and \mathbb{N}_l is a finite set of linearly ordered time points of cardinal l . A chronicle is a triplet $C = (\xi, C_T, G)$ such that $\xi \subseteq E$, C_T is the set of temporal constraints. $G = (N, It)$ is a directed graph where N represent event types of E and the arcs It represent the relationship between events $\sigma \in E$, if the event σ_1 occurs t time units after σ_2 , then it exists a directed link from σ_1 to σ_2 associated with a time constraint. Considering the two events (σ_i, t_i) and (σ_j, t_j) , we define the time interval as the pair $\tau_{ij} = [t^-, t^+]$, $\tau_{ij} \in C_T$ corresponding to the lower and upper bounds on the temporal distance between the two event dates t_i and t_j [17]. The idea of our proposal is to design the chronicles from the hybrid causal model of the system. Indeed the evolution of the system can be captured with temporal runs from which chronicles can be learn (See Figure 2). More precisely, the system initiates in the state q_0 and it evolves according to the transitions resulting from the events defined by the procedure actions for specific scenarios (startup/shutdown). For a given system modes $q_i \in Q$, the associated CSD_i is used to generate the set of event types corresponding to the evolution of the continuous process variables. A run is defined by a sequence of event types $\alpha_1, \alpha_2, \dots, \alpha_n$ where $\alpha_i \in E$ generated for each scenario using the startup/shutdown procedures. These runs with time constraints permit to construct the chronicle database of the system. In this preliminary approach, time constraints are obtained by simulation.

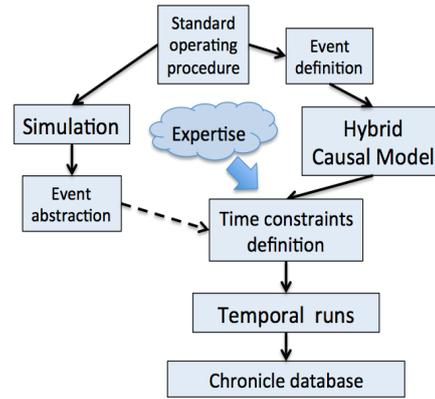


Figure 2: Principle of chronicle generation

4.2 Temporal runs

We denote a temporal run as $\langle R, T \rangle$ where R is a run and T is the time graph of the run that includes the time constraints C_T between each pair of time points where must occurs the events type. Figure 3 gives time graph examples and the possible composition of time graphs. In our approach the

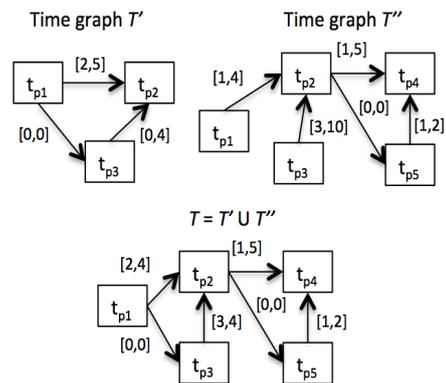


Figure 3: Time graphs example

runs are issued from the system evolution from one operation mode to another. The interleaved sequence of event types $\alpha_1, \alpha_2, \dots, \alpha_n$ represents the procedure actions and the behavior evolution of the process variables. The time constraints between each pair of event types are determined by simulation of the continuous behavior for each process variable, responding to the procedure actions.

5 Generation of Chronicles

5.1 Chronicle database

An industrial or complex process Pr is composed of different areas $Pr = \{Ar_1, Ar_2, \dots, Ar_n\}$ where each area Ar_k has different operational modes such as startup, shutdown, slow march, fast march, etc. The set C_{Ar_k} of chronicles C_{ij}^k for each area Ar_k is presented in the matrix below, where the rows represent the operating modes (i.e. O_1 : *Startup*, O_2 : *Shutdown*, O_3 : *Startup_{type}*, O_4 : *Startup_{type}*, etc)

and the columns the different faults.

$$CAr_k = \begin{matrix} & N & f_1 & f_2 & \dots & \dots & f_n \\ \begin{matrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ \dots \\ \dots \\ O_j \end{matrix} & \begin{bmatrix} C_{01}^k & C_{11}^k & C_{21}^k & \dots & \dots & C_{i1}^k \\ C_{02}^k & C_{12}^k & C_{22}^k & \dots & \dots & C_{i2}^k \\ C_{03}^k & C_{13}^k & C_{23}^k & \dots & \dots & C_{i3}^k \\ C_{04}^k & C_{14}^k & C_{24}^k & \dots & \dots & C_{i4}^k \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ C_{0j}^k & C_{1j}^k & C_{2j}^k & \dots & \dots & C_{ij}^k \end{bmatrix} \end{matrix} \quad (5)$$

The chronicle database used for diagnosis is composed by the entries of all the matrices $\{CAr_k\}$. This chronicle database is submitted to a chronicle recognition system that identifies in an observable flow of events all the possible matching with the set of chronicles from which the situation (normal or faulty) can be assessed.

5.2 Chronicle learning

As explained previously when the system changes mode of operation, a set of event types occurs forming a run R . As this evolution is due to procedure actions. Not only a unique temporal run can occur. Hence, we need to set up the maximal number of temporal runs that it could occur in each scenario represented in the matrix (5). To obtain the chronicle in each scenario is necessary to obtain the larger time graph with as many event types and with the minimal values of the constraints. [18] proposes to determine the chronicles from the temporal runs. They define a partial order relation between two temporal runs as $\langle R, T \rangle \leq \langle R', T' \rangle$ when the set of event types in R' is a subset of event type in R and the time graphs T and T' are related by $T \preceq T'$ determining the result graph where exists a unique equivalent constraint that is the minimal. The relation \preceq expresses that the set of constraints in the time graph T' is a subset of constraints in T , $C_T(t, t') \subseteq C_{T'}(t, t')$. Therefore, we apply the composition (see Figure 3) between the time graphs in order to merge the constraints obtaining the larger and constrained time graph that represents the chronicle in that scenario. Figure 4 gives an example of a chronicle generation from a maximal temporal run. In the next section a case study is presented in

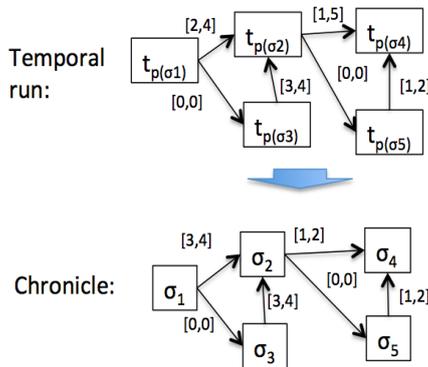


Figure 4: Chronicle example

which the chronicle generation from the temporal runs is illustrated.

6 Case study

6.1 HTG (Hydrostatic Tank Gauging) system

In the Cartagena Refinery currently are being implemented news units and elements. In the startup stage they will need a tool to help the operator to recognize dangerous conditions. We will analyze the startup and shutdown stages in the unit of water injection. This process is a HTG (Hydrostatic Tank Gauging) system composed by the following components: one tank (TK), two normally closed valves ($V1$ and $V2$), one pump (Pu), a level sensor (LT), a pressure sensor (PT), inflow sensor (FT_1) and an outflow sensor (FT_2), see Figure 5.

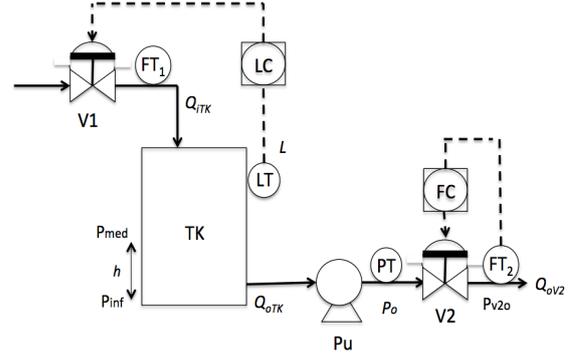


Figure 5: Process diagram

Assuming this system as a hybrid causal model, the underlying discrete event system and the different process operation modes are described in Figure 6 where we can see a possible correct evolution for the startup procedure. The events $V1_{c,o}$, $V2_{c,o}$ represents that the valves $V1, V2$ move from the state closed to the state opened, the events $V1_{o,c}, V2_{o,c}$ represents on the contrary the valves moving from the state opened to the state closed. The event Pu_{f-n} indicates that the pump Pu is turned on and the event Pu_{n-f} indicates that the pump Pu is turned off.

6.2 Identification of causal relationships

The level (L) in the tank is related to the weight (m) of the liquid inside, its density (ρ) and the tank area (A). The density (ρ) is the relationship of the pressures (P_{med}, P_{inf}) in separated points (h). Based on the global material balance, we define that the input flow is equal to the outlet flow. Then, the variation of the weight ($dm(t)/dt$) in the tank is proportional to the difference between the inflow (Q_{iTK}) and the outflow (Q_{oV2}). The differential pressure in the pump and in $V2$ are specified as ΔP_{Pu} and ΔP_{V2} . The outlet pressure in the pump (P_o) is related with the outlet flow tank (Q_{oTK}), the revolutions per minute in the pump (RPM_{Pu}), his capacity (C) and the radio of the outlet pipe (r). The outflow (Q_{oV2}) and inflow (Q_{iTK}) control are related to the percentage aperture of the valves $V1$ ($LV1$) and $V2$ ($LV2$) and differential pressures ($\Delta P_{V1}, \Delta P_{V2}$). In Figure 7 we can see the CSD of the system in the modes q_1 , q_5 and q_7 . For example, the mode q_1 activates the influence of Q_{iTK} to L . The mode q_5 activates the influence of Q_{iTK} to L and the influence of L to P_o and finally the mode q_7 activates the influence of Q_{iTK} to L , L to P_o and P_o to Q_{oV2} .

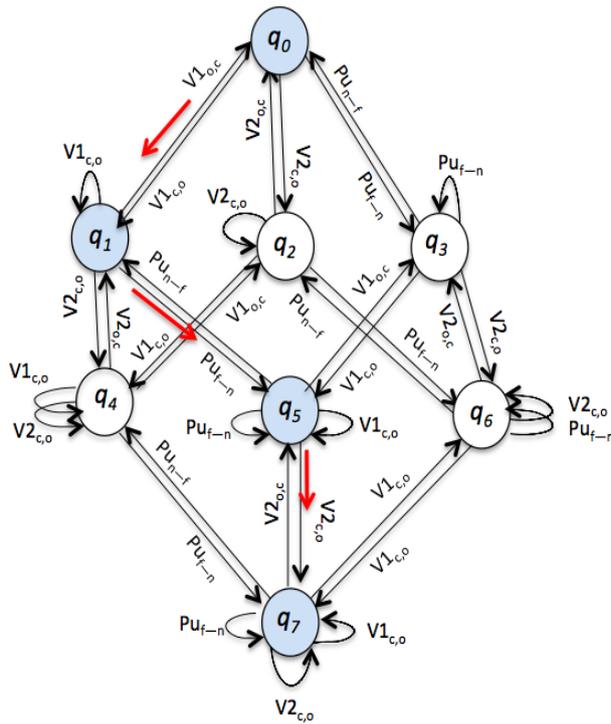
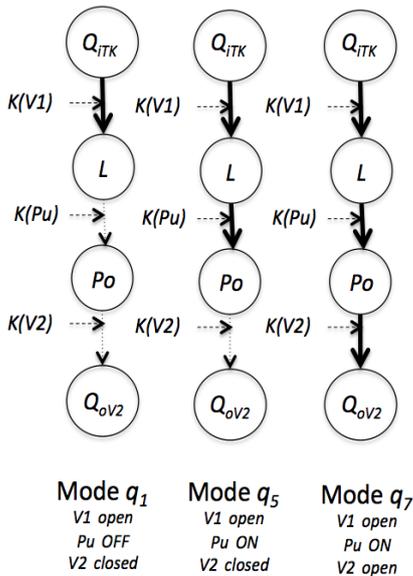


Figure 6: Underlying DES of the HGT system


 Figure 7: CSD in the modes q_1 , q_5 and q_7

6.3 Event identification

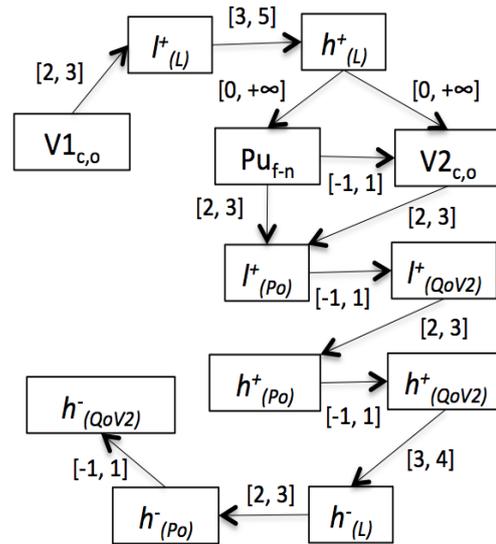
One of the most important steps for fault diagnosis based on chronicle recognition is to determine the set of events that can carry the system to a failure. Each situation pattern (normal or abnormal) is a set of events and temporal constraints between them; then a situation model may also specify events to be generated and actions to be triggered as a result of the situation occurrence. For a startup procedure in the example process, the set of event types Σ that represent the procedure actions is:

$$\Sigma = \{V1_{c,o}, V2_{c,o}, Pu_{f-n}, V1_{o,c}, V2_{o,c}, Pu_{n-f}\} \quad (6)$$

According to the causal graphs associated to the modes involved in the sequence of procedure actions (i.e q_1 , q_5 and q_7 indicated by red arrows on Figure 6), the event types of Σ^c correspond to the behavior of the variables L, P_o and Q_{oV2} .

$$\Sigma^c = \{l_{(L)}^+, l_{(L)}^-, h_{(L)}^+, h_{(L)}^-, l_{(P_o)}^+, l_{(P_o)}^-, h_{(P_o)}^+, h_{(P_o)}^-, l_{(Q_{oV2})}^+, l_{(Q_{oV2})}^-, h_{(Q_{oV2})}^+, h_{(Q_{oV2})}^-\} \quad (7)$$

From the startup/shutdown procedures the different temporal runs are determined and these temporal runs are related to the normal and abnormal situations. The chronicle resulting from a normal startup procedure is presented in Figure 8. The model system was developed in Matlab including


 Figure 8: Chronicle C_{01} for normal behavior startup

the injection water process area. The continuous behavior is related to the evolution of the level L , outlet pump pressure P_o and the outlet flow Q_{oV2} in the system. The discrete evolution is related to the event evolution of the procedures in the startup and shutdown stages. From the different failure modes of the process, the dynamic behavior of the variables is shown with a detection for the possible process states, including the normal procedure without failure. The simulation includes 3 types of startup procedures (OK , $fail_1$ and $fail_2$) with 4 types of fault modes (V_1 , V_2 , $Pump$ and $Drain_{open}$) and 3 types of Shutdown procedure (OK , $Non - actived$ and $Fail$). The evolution of the continuous variables in the startup procedure without failure is shown in Figure 9. The events are generated by the program through the evolution of the differential equations, the

variable conditions and the procedural actions. Recognition of the chronicles was done using the tool *stateflow*.

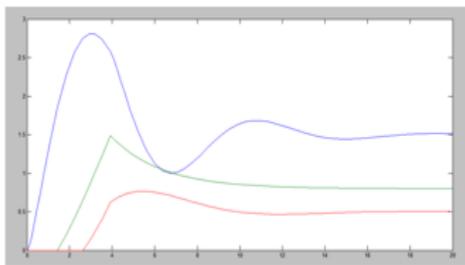


Figure 9: Normal behavior in startup procedure without failure. Blue: Level, Green: Pressure, Red: outletflow

7 Conclusion

A preliminary method for alarm management based on automatically learned chronicles has been proposed. The proposal is based on a hybrid causal model of the system and a chronicle based approach for diagnosis. An illustrative example of an hydrostatic tank gauging has been considered to introduce the main concepts of the approach. In this paper the design of the temporal constraints of the chronicles were performed from simulation results, but further research aim to generate the chronicles from the model of the system. Learning approaches are currently considered for acquiring the chronicle base directly from the sequences of events representing the situations. For this propose the algorithm HC-DAM (Heuristic Chronicle Discovery Algorithm Modified [17]) may be used. The use of HIL (Hardware in the loop) to simulate and validate the proposal is also in our prospects.

8 Acknowledge

The ECOPEPETROL - ICP engineers Jorge Prada, Francisco Cala and Gladys Valderrama help us to develop and validate the simulations.

References

- [1] S. Ferrer D. Beebe and D. Logerot. The connection of peak alarm rates to plant incidents and what you can do to minimize. *Process Safety Progress*, 2013.
- [2] J. Zhao J. Zhu, Y. Shu and F. Yang. A dynamic alarm management strategy for chemical process transitions. *Journal of Loss Prevention in the Process Industries* 30 207e218, 2014.
- [3] N. F. Thornhill M. Schlegel, L. Christiansen and A. Fay. A combined analysis of plant connectivity and alarm logs to reduce the number of alerts in an automation system. *Journal of Process Control* 23 839–851, 2013.
- [4] I. Izadi S. L. Shaha T. Black R. Sandeep, R. Kondaveeti and T. Chen. Graphical tools for routine assessment of industrial alarm systems. *Computers and Chemical Engineering* 46 39–47, 2012.
- [5] T. Takai M. Noda F. Higuchi, I. Yamamoto and H. Nishitani. Use of event correlation analysis to reduce number of alarms. *Computer Aided Chemical Engineering*, 27, 1521e1526., 2009.
- [6] Z. Ge and Z. Song. Multimode process monitoring based on bayesian method. *Journal of Chemometrics*, 23, 636e650., 2009.
- [7] M. Noda X. Liu and H. Nishitani. Evaluation of plant alarm systems by behavior simulation using a virtual subject. *Computers & Chemical Engineering*, 34, 374e386, 2010.
- [8] D. Xiao F. Yang, S. L. Shah and T. Chen. Improved correlation analysis and visualization of industrial alarm data. *18th IFAC World Congress Milano (Italy)*, 2011.
- [9] D.S. Shook S.R. Kondaveeti I. Izadi, S.L. Shah and T. Chen. A framework for optimal design of alarm systems. *7th IFAC symposium on fault detection, supervision and safety of technical processes, Barcelona, Spain*, 2009.
- [10] U.G. Oktem A. Pariyani, W.D. Seider and M. Soroush. Dynamic risk analysis using alarm databases to improve process safety and product quality: Part ii bayesian analysis. *AIChE Journal*, 58, 826e841., 2012.
- [11] J. Liu and D. Chen. Non stationary fault detection and diagnosis for multimode processes. *AIChE Journal*, 56, 207e219., 2010.
- [12] L. Boang Z. Jing and Y. Hao. Fault diagnosis strategy for startup process based on standard operating procedures. *25th Chinese Control and Decision Conference (CCDC)*, 2013.
- [13] H. Vedam R. Srinivasan, P. Viswanathan and A. Nochur. A framework for managing transitions in chemical plants. *Computers & Chemical Engineering*, 29, 305e322., 2005.
- [14] A. Adhitya S. Xu and R. Srinivasan. Hybrid model-based framework for alarm anticipation. *Industrial & Engineering Chemistry Research*, 2014.
- [15] L. Travé-Massuyès R. Pons, A. Subias. Iterative hybrid causal model based diagnosis: Application to automotive embedded functions. *Engineering Applications of Artificial Intelligence*, 2015.
- [16] L. Travé-Massuyès and R. Pons. Causal ordering for multiple mode systems. in: *11th International Workshop on Qualitative Reasoning, Cortona, Italy*, pp. 203–214, 1997.
- [17] L. Travé-Massuyès A. Subias and E. Le Corrionc. Learning chronicles signing multiple scenario instances. *IFAC World Congress, Le Cap, South Africa, 26-29 August, 2014* ; also *25th International Workshop on Principles of Diagnosis (DX-2015), Graz (Austria), 9-11 September*, 2014.
- [18] Bruno Guerraz and Christophe Dousson. Chronicles construction starting from the fault model of the system to diagnose. *DX04 15th International Workshop on Principles of Diagnosis. Carcassonne (France)*, 2004.

Data-Augmented Software Diagnosis

Amir Elmishali¹ and Roni Stern¹ and Meir Kalech¹

¹Ben Gurion University of the Negev

e-mail: amir9979@gmail.com, roni.stern@gmail.com, kalech@bgu.ac.il

Abstract

The task of software diagnosis algorithms is to identify which software components are faulty, based on the observed behavior of the system. Software diagnosis algorithms have been studied in the Artificial Intelligence community, using a model-based and spectrum-based approaches. In this work we show how software fault prediction algorithms, which have been studied in the software engineering literature, can be used to improve software diagnosis. Software fault prediction algorithms predict which software components is likely to contain faults using machine learning techniques. The resulting data-augmented diagnosis algorithm we propose is able to overcome of key problems in software diagnosis algorithms: ranking diagnoses and distinguishing between diagnoses with high probability and low probability. This allows to significantly reduce the outputted list of diagnoses. We demonstrate the efficiency of the proposed approach empirically on both synthetic bugs and bugs extracted from the Eclipse open source project. Results show that the accuracy of the found diagnoses is substantially improved when using the proposed combination of software fault prediction and software diagnosis algorithms.

1 Introduction

Software is prevalent in practically all fields of life, and its complexity is growing. Unfortunately, software failures are common and their impact can be very costly. As a result, there is a growing need for automated tools to identify software failures and isolate the faulty software components, such as classes and functions, that have caused the failure. We focus on the latter task, of *isolating faults in software components*, and refer to this task as software diagnosis.

Model-based diagnosis (MBD) is an approach to automated diagnosis that uses a model of the diagnosed system to infer possible *diagnoses*, i.e., possible explanations of the observed system failure. While MBD was successfully applied to a range of domains [1; 2; 3; 4], it has not been applied successfully yet to software. The reason for this is that in software development, there is usually no formal model of the developed software. To this end, a scalable software diagnosis algorithm called Barinel has been proposed [5].

Barinel is a combination of MBD and Spectrum Fault Localization (SFL). SFL considers traces of executions, and finds diagnoses by considering the correlation between execution traces and which executions have failed. While very scalable, Barinel suffers from one key disadvantage: it can return a very large set of possible diagnoses for the software developer to choose from. To handle this disadvantage, Abreu et al. [5] proposed a Bayesian approach to compute a likelihood score for each diagnosis. Then, diagnoses are prioritize according to their likelihood scores.

Thanks to the open source movement and current software engineering tools such as version control and issue tracking systems, there is much more information about a diagnosed system than revealed by the traces of performed tests. For example, version control systems store all revisions of every source files, and it is quite common that a bug occurs in a source file that was recently revised. Barinel is agnostic to this data. We propose a data-driven approach to better prioritize the set of diagnoses returned by Barinel.

In particular, we use methods from the software engineering literature to learn from collected data how to predict which software components are expected to be faulty. These predictions are then integrated into Barinel to better prioritize the diagnoses it outputs and provide more accurate estimates of each diagnosis likelihood.

The resulting data-augmented diagnosis algorithm is part of a broader software troubleshooting paradigm that we call *Learn, Diagnose, and Plan (LDP)*. In this paradigm, illustrated in Figure 1(a), the troubleshooting algorithm learns which source files are likely to fail from past faults, previous source code revisions, and other sources. When a test fails, a data-augmented diagnosis algorithm considers the observed failed and passed tests to suggest likely diagnoses leveraging the knowledge learned from past data. If further tests are necessary to determine which software component caused the failure, such test are planned automatically, taking into consideration the diagnoses found. This process continues until a sufficiently accurate diagnoses is found.

In this work we implemented this paradigm and simulated its execution on a popular open source software project – the Eclipse CDT. Information from the Git version control and the Bugzilla issue tracking systems was used, as illustrated in Figure 1(b) and explained in the experimental results.

Results show a huge advantage of using our data-augmented diagnoser over Barinel with uniform priors for both finding more accurate diagnoses and for better selecting tests for troubleshooting. Moreover, to demonstrate the potential benefit of our data-augmented approach we also

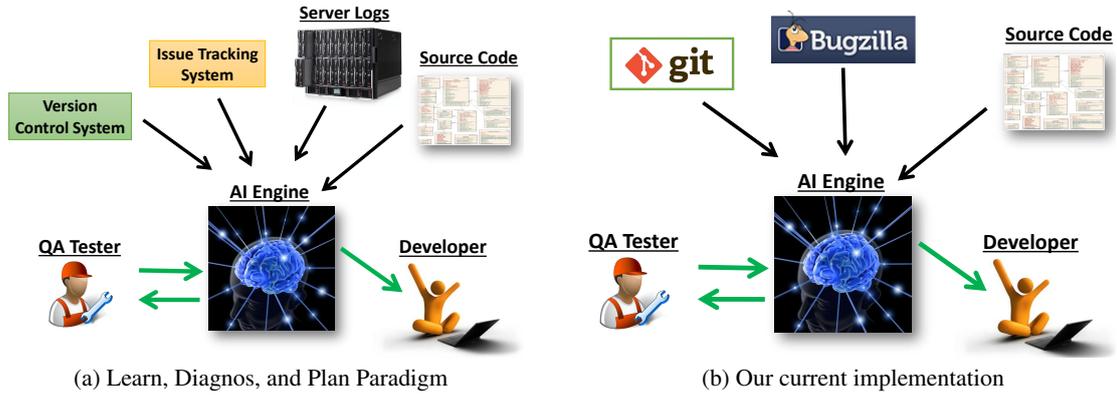


Figure 1: The learn, diagnose, and plan paradigm and our implementation.

experimented with a synthetic fault prediction model that is correctly identifies the faulty component. As expected, using the synthetic fault prediction model is better than using the learned fault prediction model, thus suggesting room for further improvements in future work. To our knowledge, this is the first work to integrate successfully a data-driven approach into software diagnosis.

2 Model-Based Diagnosis for Software

The input to classical MBD algorithms is a tuple $\langle SD, COMPS, OBS \rangle$, where SD is a formal description of the diagnosed system’s behavior, $COMPS$ is the set of components in the system that may be faulty, and OBS is a set of observations. A diagnosis problem arises when SD and OBS are inconsistent with the assumption that all the components in $COMPS$ are healthy. The output of an MBD algorithm is a set of *diagnoses*.

Definition 1 (Diagnosis). *A set of components $\Delta \subseteq COMPS$ is a diagnosis if*

$$\bigwedge_{C \in \Delta} (\neg h(C)) \wedge \bigwedge_{C' \notin \Delta} (h(C')) \wedge SD \wedge OBS$$

is consistent, i.e., if assuming that the components in Δ are faulty, then SD is consistent with OBS .

The set of components ($COMPS$) in software diagnoses can be, for example, the set of classes, or all functions, or even a component per line of code. Low level granularity of components, e.g., setting each line of code as a component, will result in very focused diagnoses (e.g., pointing on the exact line of code that was faulty). Focusing the diagnoses in such way comes at a price of an increase in the computational effort. Automatically choosing the most suitable level of granularity is a topic for future work.

Observations (OBS) in software diagnosis are observed executions of tests. Every observed test t is labeled as “passed” or “failed”, denoted by $passed(t)$ and $failed(t)$, respectively. This labeling is done manually by the tester or automatically in case of automated tests (e.g., failed assertions).

There are two main approaches for applying MBD to software diagnosis, each defining SD somewhat differently. The first approach requires SD to be a logical model of the correct functionality of every software component [6]. This approach allows using logical reasoning techniques to infer diagnoses. The main drawbacks of this approach is that it

does not scale well and modeling the behavior of software component is often infeasible.

2.1 SFL for Software Diagnosis

An alternative approach to software diagnosis has been proposed by Abreu et al. (5; 7), based on *spectrum-based fault localization (SFL)*. In this SFL-based approach, there is no need for a logical model of the correct functionality of every software component in the system. Instead, the *traces* of the observed tests are considered.

Definition 2 (Trace). *A trace of a test t , denoted by $trace(t)$, is the sequence of components involved in executing t .*

Traces of tests can be collected in practice with common software profilers (e.g., Java’s JVMTI). Recent work showed how test traces can be collected with low overhead [8]. Also, many implemented applications maintain a log with some form of this information.

In the SFL-based approach, SD is implicitly defined in SFL by the assumption that a test will pass if all the components in its trace are not faulty. Let $h(C)$ denote the health predicate for a component C , i.e., $h(C)$ is true if C is not faulty. Then we can formally define SD in the SFL-based approach with the following set of Horn clauses:

$$\forall test \left(\bigwedge_{C \in trace(test)} h(C) \rightarrow passed(test) \right)$$

Thus, if a test failed then we can infer that at least one of the components in its trace is faulty. In fact, a trace of a failed test is a *conflict*.

Definition 3 (Conflict). *A set of components $\Gamma \subseteq COMPS$ is a conflict if $\bigwedge_{C \in \Gamma} h(C) \wedge SD \wedge OBS$ is inconsistent.*

Many MBD algorithms use conflicts to direct the search towards diagnoses, exploiting the fact that a diagnosis must be a hitting set of all the conflicts [9; 10; 11]. Intuitively, since at least one component in every conflict is faulty, only a hitting set of all conflicts can explain the unexpected observation (failed test).

Barinel is a recently proposed software MBD algorithm [5] based on exactly this concept: considering traces of tests with failed outcome as conflicts and returning their hitting sets as diagnoses. With a fast hitting set algorithm, such as the STACATTO hitting set algorithm proposed by Abreu et al. [12], Barinel can scale well to large systems. The main drawback of using Barinel is that it often outputs a large set of diagnoses, thus providing weaker guidance to the programmer that is assigned to solve the observed bug.

2.2 Prioritizing Diagnoses

To address this problem, Barinel computes a *score* for every diagnosis it returns, estimating the likelihood that it is true. This serves as a way to prioritize the large set of diagnoses returned by Barinel.

The exact details of how this score is computed is given by Abreu et al. [5]. For the purpose of this paper, it is important to note that the score computation used by Barinel is Bayesian: it computes for a given diagnosis the posterior probability that it is correct given the observed passes and failed tests. As a Bayesian approach, Barinel also requires some assumption about the *prior probability* of each component to be faulty. Prior works using Barinel has set these priors uniformly to all components. In this work, we propose a data-driven way to set these priors more intelligently and demonstrate experimentally that this has a huge impact of the overall performance of the resulting diagnoser.

3 Data-Augmented Software Diagnosis

The prior probabilities used by Barinel represent the a-priori probability of a component to be faulty, without considering any observed system behavior. Fortunately, there is a line of work on *software fault prediction* in the software engineering literature that deals exactly with this question: which software components is more likely to have a bug. We propose to use these software fault predictions as priors to be used by Barinel. First, we provide some background on software fault prediction.

3.1 Software Fault Prediction

Fault prediction in software is a classification problem. Given a software component, the goal is to determine its class – healthy or faulty. Supervised machine learning algorithms are commonly used these days to solve classification problems. They work as follows. As input, they are given a set of *instances*, in our case these are software components, and their correct labeling, i.e., the correct class for each instance. They output a *classification model*, which maps an instance to a class.

Learning algorithm extract *features* from a given instance, and try to learn from the given labeled instances the relation between the features of an instance and its class. Key to the success of machine learning algorithms is the choice of *features* used. Many possible features were proposed in the literature for software fault prediction.

Radjenovic et al. [13] surveyed the features used by existing software prediction algorithms and categorizes them into three families. **Traditional.** These features are traditional software complexity metrics, such as number of lines of code, McCabe [14] and Halstead [15] complexity measures.

Object Oriented. These features are software complexity metrics that are specifically designed for object oriented programs. This includes metrics like cohesion and coupling levels and depth of inheritance.

Process. These features are computed from the software change history. They try to capture the dynamics of the software development process, considering metrics such as lines added and deleted in the previous version and the age of the software component.

It is not clear from the literature which combination of features yields the most accurate fault predictions. In a

preliminary set of experiments we found that the combination of features that performed best is a combination of 68 features from the features listed by Radjenovic et al. [13] worked best. This list of features included the McCabe [14] and Halstead [15] complexity measures, several object oriented measures such as the number of methods overriding a superclass, number of public methods, number of other classes referenced, and is the class abstract, and several process features such as the age of the source file, the number of revisions made to it in the last release, the number of developers contributed to its development, and the number of lines changed since the latest version.

As shown in the experimental results section, the resulting fault prediction model was accurate enough so that the overall data-augmented software diagnoser be more effective than Barinel with uniform priors. However, we are not sure that a better combination of features cannot be found, and this can be a topic for future work. The main novelty of our work is in integrating a software fault prediction model with the Barinel.

3.2 Integrating the Fault Prediction Model

The software fault prediction model generated as described above is a classifier, accepting as input a software component and outputting a binary prediction: is the component predicted to be faulty or not. Barinel, however, requires a real number that estimates the prior probability of each component to be faulty.

To obtain this estimated prior from the fault prediction model, we rely on the fact that most prediction models also output a confidence score, indicating the model's confidence about the classified class. Let $conf(C)$ denote this confidence for component C . We use $conf(C)$ for Barinel's prior if C is classified as faulty, and $1 - conf(C)$ otherwise.

4 Experimental Results

To demonstrate the benefits of the proposed data-augmented approach, we implemented it and evaluated it as follows.

4.1 Experimental Setup

As a benchmark, we used the source files, tests, and bugs reported for the Eclipse CDT open source software project (eclipse.org/cdt). Eclipse CDT is a popular open source Integrated Development Environment (IDE) for C/C++. The first release dates back to December 2003 and the latest release we consider, labeled CDT 8.2.0, was released in June 2013. It consists of 8,502 source code files and have had more than 10,129 bugs reported so far (for all releases). In addition, there are 3,493 automated tests coded using the JUnit unit testing framework.

Determining Faulty Files

Eclipse CDT is developed using the Git version control system and the Bugzilla issue tracking system. Git maintains all versions of each source file in a repository. This enables computing process metrics for every version of every source file. Similarly, Bugzilla is used to maintain all reported bugs. Some source file versions are marked in the Git repository as versions in which a specific bug was fixed. The Git repository for Eclipse CDT contained matching versions of source files for 6,730 out of 10,129 bugs reported as fixed in Bugzilla. We performed our experiments on these 6,730 bugs.

For both learning and testing a fault prediction model, we require a mapping between reported bug and the source files that were faulty and caused it. One possible assumption is that *every* source file revision that is marked as fixing bug X is a faulty file that caused X . We call this the “All files” assumption. The “All files” assumption may overestimate the number of faulty files as some of these files may have been modified due to other reasons, not related to the bug. Even if all changes in a revision are related to fixing a bug, it still does not mean that all these files are faulty. For example, properties files and XML configuration files. As a crude heuristic to overcome this, we also experiment with an alternative assumption that we call the “Most modified” assumption. In the “Most modified” assumption, for a given bug X we only consider a single source file as faulty from all the files associated with bug X . We chose from these source file the one in which the revision made to that source file was the most extensive. The extensiveness of the revision is measured by the number of lines added, updated, and deleted to the source file in this revision. Below we present experiments for both “All files” and “Most modified” assumptions. Śliwerski et al. [16] proposed a more elaborate method to heuristically identify the source files that are caused the bug, when analyzing a similar data set.

Training and Testing Set

The sources files and reported bugs from 5 releases, 8.0.0–8.1.1, were used to train the model of our data-augmented diagnoser, and the source files and reported bugs from release 8.1.2 were used to evaluate it.

4.2 Comparing Fault Prediction Accuracy

As a preliminary, we evaluated the quality of the fault prediction models used by our data-augmented diagnoser on our Eclipse CDT benchmark.

All files	Precision	Recall	F-Measure	AUC
Random Forest	0.56	0.09	0.16	0.84
J48	0.44	0.17	0.25	0.61
Naive Bayes	0.27	0.31	0.29	0.80
Most modified	Precision	Recall	F-Measure	AUC
Random Forest	0.44	0.04	0.08	0.76
J48	0.15	0.03	0.05	0.55
Naive Bayes	0.08	0.31	0.12	0.715

Table 1: Faulty prediction performance.

We used the Weka software package (www.cs.waikato.ac.nz/ml/weka) to experiment with several learning algorithms and compared the resulting fault prediction models. Specifically, we evaluated the following learning algorithms: Random Forest, J48 (Weka’s implementation of a decision tree learning algorithm), and Naive Bayes. Table 1 shows the precision, recall, F-measure, and AUC of the fault prediction models generated by each of these learning algorithms. These are standard metrics for evaluating classifiers. In brief, *precision* is the ratio of faulty files among all files identified by the evaluated model as faulty. *Recall* is the number of faulty files identified as such by the evaluated model divided by the total number of faulty files. *F-measure* is a known combination of precision and recall. The AUC metric addresses the known tradeoff between recall and precision, where high recall often comes at the price of low precision. This tradeoff can be controlled by setting different sensitivity thresholds to the evaluated model. AUC

is the area under the curve plotting the accuracy as a function of the recall (every point is a different threshold value).

All metrics range between zero and one (where one is optimal) and are standard metrics in machine learning and information retrieval. The unfamiliar reader can find more details in Machine Learning books, e.g. Mitchell’s classical book [17].

The results for both “All files” and “Most modified” assumptions show that the Random Forest classifier obtained the overall best results. This corresponds to many recent works. Thus, in the results reported henceforth, we only used the model generated by the Random Forest classifier in our data-augmented diagnoser. The precision and especially recall results are fairly low. This is understandable, as most files are healthy, and thus the training set is very imbalanced. This is a known inhibitor to performance of standard learning algorithms. We have experimented with several known methods to handle this imbalanced setting, such as SMOTE and random under sampling, but these did not produce substantially better results. However, as we show below, even this imperfect prediction model is able to improve the existing data-agnostic software diagnosis algorithm. Note that we also experimented with other popular learning algorithms such as Support Vector Machine (SVM) and Artificial Neural Network (ANN), but their results were worse than those shown in Table 1.

Next, we evaluate the performance of our data-augmented diagnoser in two diagnostic tasks: finding diagnoses and guiding test generation.

4.3 Diagnosis Task

First, we compared the data-agnostic diagnoser with the proposed data-augmented diagnoser in the task of finding accurate diagnoses. The input is a set of tests, with their traces and outcomes and the output is a set of diagnoses, each diagnosis having a score that estimates its correctness. This score was computed by Barinel as described earlier in the paper, where the data-agnostic diagnoser uses uniform priors and the proposed data-augmented diagnoser uses the predicted fault probabilities from the learned model.

Diagnoser	Most modified		All files	
	Precision	Recall	Precision	Recall
Data-agnostic	0.72	0.27	0.55	0.26
Data-augmented	0.90	0.32	0.73	0.35
Syn. (0.6,0.01)	0.97	0.39	0.96	0.45
Syn. (0.6,0.1)	0.84	0.35	0.89	0.42
Syn. (0.6,0.2)	0.77	0.34	0.83	0.39
Syn. (0.6,0.3)	0.73	0.33	0.78	0.37
Syn. (0.6,0.4)	0.69	0.32	0.74	0.36

Table 2: Comparison of diagnosis accuracy.

To compare the set of diagnoses returned by the different diagnosers, we computed the *weighted average* of their precision and recall. This was computed as follows. First, the precision and recall for every diagnoses was computed. Then, we averaged these values, weighted by the score given to the diagnoses by Barinel. This enables aggregating the precision and recall of a set of diagnoses while incorporating which diagnoses are regarded as more likely according to Barinel’s. For brevity, we will refer to this weighted average precision and weighted average recall as simply precision and recall.

Table 2 shows the precision and recall results of the data-agnostic diagnoser and our data-augmented diagnoser, for both “Most modified” and “All files” assumptions. Each result in the table is an average over the precision and recall obtained for 50 problem instances. A problem instance consists of (1) a bug from one of the bugs reported for release 8.1.2. of Eclipse CDT, and (2) a set of 25 tests, chosen randomly, while ensuring that at least one tests would pass through the faulty files.

Both precision and recall of the data-augmented and data-agnostic diagnosers support the main hypothesis of this work: a data-augmented diagnoser can yield substantially better diagnoses than a data-agnostic diagnoser. For example, the precision of the data-augmented diagnoser under the “Most modified” assumption is 0.9 while that of the data-agnostic diagnoser is only 0.72. The superior performance of the data-augmented diagnoser is shown for both “Most modified” and “All files” assumptions. Another observation that can be made from the results in Table 2 is that while the precision of the data-augmented diagnoses is very high and is substantially better than that of the data-agnostic diagnoser, the improvement in recall is relatively more modest. This can be explained by the precision and recall results of the learned model, shown in Table 1 and discussed earlier. There too, the recall results was far worse than the precision results (recall that we are using the model learned by the Random Forest learning algorithm). It is possible that learning a model with higher recall may result in higher recall for the resulting diagnoses. We explore the impact of learning more accurate fault prediction model next.

Synthetic Priors

The data-augmented diagnoser is based on the priors generated by the learned fault prediction model. Building better fault prediction models is an active field of study [13] and thus future fault prediction models may be more accurate than the ones used by our data-augmented diagnoser. To evaluate the benefit of a more accurate fault prediction model on our data-augmented diagnoser, we created a *synthetic fault prediction model*, in which faulty source files get P_f probability and healthy source files get P_h , where P_f and P_h are parameters. Setting $P_h = P_f$ would cause the data-augmented diagnoser to behave in a uniform distribution exactly like the data-agnostic diagnoser, setting the same prior probability for all source files to be faulty. By contrast, setting $P_h = 0$ and $P_f = 1$ represent an optimal fault prediction model, that exactly predicts which files are faulty and which are healthy.

The lines marked “Syn. (X,Y)” in Table 2 mark the performance of the data-augmented diagnoser when using this synthetic fault prediction model, where $X = P_f$ and $Y = P_h$. Note that we experimented with many values of P_f and P_h , and presented above a representative subset of these results.

As expected, setting lowering the value of P_h results in more better diagnoses being found. Setting a very low P_h value improves the precision significantly up to almost perfect precision (0.97 and 0.96 for the “Most modified” and “All files”, respectively). The recall results, while also improving as we lower P_h , do not reach a very high value. For $P_h = 0.01$, the obtained recall is almost 0.39 and 0.45 for the “Most modified” and “All files”, respectively.

A possible explanation for these low recall results lays in the fact that all the evaluated diagnosers use the Barinel di-

agnosis algorithm with different fault priors. Barinel uses these priors only to prioritize diagnoses, but Barinel considers as diagnoses hitting sets of faulty traces. Thus, if two faulty components are used in the same trace, only one of them will be detected even if both have very high likelihood of being faulty according to the fault prediction model.

Considering More Tests

Next, we investigate the impact of adding more tests to the accuracy of the returned diagnoses.

Figure 2 shows the precision and recall results (Figures 2 (a) and (b), respectively), as a function of the number of observed tests. We compared the different diagnosers, given 25, 40, 70, 100, and 130 observed tests.

The results show two interesting trends in both precision and recall. First, as expected, the data-agnostic diagnoser performs worse than the data-augmented diagnoser, which in terms performs worse than the diagnoser using a synthetic fault prediction model, with $P_h = 0.01$. This supports our main hypothesis — that data-augmented diagnosers can be better than a data-agnostic diagnoser. Also, the better performance of Syn. (0.6, 0.01) demonstrates that future research on improving the fault prediction model will result in a better diagnoser.

The second trend is that adding more tests reduces the precision and recall of the returned diagnoses. This, at first glance, seem counter-intuitive, as we would expect more tests to allow finding more accurate diagnoses and thus higher recall and precision. This non-intuitive results can be explained by how tests were chosen. As explained above, the observed tests were chosen randomly, only verifying that at least one test passes through each faulty source file. Adding randomly selected tests adds noise to the diagnoser. By contrast, intelligent methods to choose which tests to add can improve the accuracy of the diagnoses [18]. This is explored in the next section. Another reason for the degraded performance when adding more tests is that more tests may pass through more fault source files, in addition to those from the specific reported bug used to generate the problem instance in the first place. Thus, adding more tests increases the amount of faulty source files to detect.

4.4 Troubleshooting Task

Efficient diagnosers are key components of *troubleshooting algorithms*. Troubleshooting algorithms choose which tests to perform to find the most accurate diagnosis. Zamir et al. [18] proposed several troubleshooting algorithms specifically designed to work with Barinel for troubleshooting software bugs. In the below preliminary study, we evaluated the impact of our data-augmented diagnoser on the overall performance of troubleshooting algorithms. Specifically, we implemented the so-called *highest probability* (HP) troubleshooting algorithm, in which tests are chosen in the following manner. HP chooses a test that is expected to pass through the source file having the highest probability of being faulty, given the diagnoses probabilities.

We run the HP troubleshooting algorithm with each of the diagnosers mentioned above (all rows in Table 2). We compared the HP troubleshooting algorithm using different diagnosers by counting the number of tests were required to reach a diagnoses of score higher than 0.7.

Table 3 shows the average number of tests performed by the HP troubleshooting algorithm until it halts (with a suitable diagnosis). The results show the same over-arching

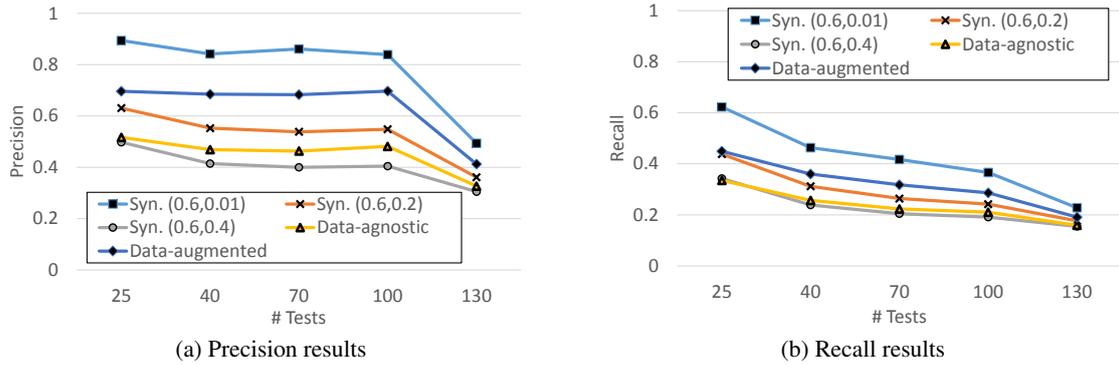


Figure 2: Diagnosis accuracy as a function of # tests given to the diagnoser.

Algorithm	Most modified	All files
Data-agnostic	20.24	18.06
Data-augmented	10.80	15.45
Syn. (0.6,0.01)	3.94	14.91
Syn. (0.6,0.1)	15.44	17.83
Syn. (0.6,0.2)	19.78	18.99
Syn. (0.6,0.3)	20.90	19.24
Syn. (0.6,0.4)	20.74	19.18

Table 3: Avg. additional tests for troubleshooting.

theme: the data-augmented diagnoser is much better than the data-agnostic diagnoser for this troubleshooting task. Also, using the synthetic fault prediction model can result in even further improvement, thus suggesting future work for improving the learned fault prediction model.

5 Conclusion, and Future Work

We presented a method for using information about the diagnosed system to improve Barinel, a scalable, effective, software diagnosis algorithm [7]. In particular, we incorporated a software fault prediction model into Barinel. The resulting data-augmented diagnoser is shown to outperform Barinel without such a fault prediction model. This was verified experimentally using a real source code system (Eclipse CDT), real reported bugs and information from the software’s source control repository. Results also suggest that future work on improving the learned fault prediction model will result in an improved diagnosis accuracy. In addition, it is worthwhile to incorporate the proposed data-augmented diagnosis methods with other proposed improvements of the based SFL-based software diagnosis, as those proposed by Hofer et al. [19; 20].

References

- [1] Brian C. Williams and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Conference on Artificial Intelligence (AAAI)*, pages 971–978, 1996.
- [2] Alexander Feldman, Helena Vicente de Castro, Arjan van Gemund, and Gregory Provan. Model-based diagnostic decision-support system for satellites. In *IEEE Aerospace Conference*, pages 1–14. IEEE, 2013.
- [3] Peter Struss and Chris Price. Model-based systems in the automotive industry. *AI magazine*, 24(4):17–34, 2003.
- [4] Dietmar Jannach and Thomas Schmitz. Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. *Automated Software Engineering*, 1:1–40, 2014.
- [5] Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. Simultaneous debugging of software faults. *Journal of Systems and Software*, 84(4):573–586, 2011.
- [6] Franz Wotawa and Mihai Nica. Program debugging using constraints – is it feasible? *Quality Software, International Conference on*, 0:236–243, 2011.
- [7] Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. Spectrum-based multiple fault localization. In *Automated Software Engineering (ASE)*, pages 88–99. IEEE, 2009.
- [8] Alexandre Perez, Rui Abreu, and André Ribeiro. A dynamic code coverage approach to maximize fault localization efficiency. *Journal of Systems and Software*, 2014.
- [9] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, 1987.
- [10] Brian C. Williams and Robert J. Ragno. Conflict-directed A* and its role in model-based embedded systems. *Discrete Appl. Math.*, 155(12):1562–1595, 2007.
- [11] Roni Stern, Meir Kalech, Alexander Feldman, and Gregory M. Provan. Exploring the duality in conflict-directed model-based diagnosis. In *AAAI*, 2012.
- [12] Rui Abreu and Arjan JC van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *SARA*, volume 9, pages 2–9, 2009.
- [13] Danijel Radjenovic, Marjan Hericko, Richard Torkar, and Ales Zivkovic. Software fault prediction metrics: A systematic literature review. *Information & Software Technology*, 55(8):1397–1418, 2013.
- [14] Thomas J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.
- [15] Maurice H. Halstead. *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc., New York, NY, USA, 1977.
- [16] Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. When do changes induce fixes? *ACM sigsoft software engineering notes*, 30(4):1–5, 2005.
- [17] Tom Mitchell. *Machine learning*. McGraw Hill, 1997.
- [18] Tom Zamir, Roni Stern, and Meir Kalech. Using model-based diagnosis to improve software testing. In *AAAI Conference on Artificial Intelligence*, 2014.
- [19] Birgit Hofer, Franz Wotawa, and Rui Abreu. Ai for the win: Improving spectrum-based fault localization. *ACM SIGSOFT Software Engineering Notes*, 37:1–8, 2012.
- [20] Birgit Hofer and Franz Wotawa. Spectrum enhanced dynamic slicing for better fault localization. In *ECAI*, pages 420–425, 2012.

Faults isolation and identification of Heat-exchanger/ Reactor with parameter uncertainties

Mei ZHANG^{1,4,5}, Boutaïeb DAHOU^{2,3} Michel CABASSUD^{4,5} Ze-tao LI¹

¹Guizhou University
gzgylzt@163.com

²CNRS LAAS, Toulouse, France
boutaib.dahhou@laas.fr

³Université de Toulouse, UPS, LAAS, Toulouse, France

⁴ Université de Toulouse, UPS, Laboratoire de Génie Chimique
michel.cabassud@ensiacet.fr

⁵ CNRS, Laboratoire de Génie Chimique

Abstract

This paper deals with sensor and process fault detection, isolation (FDI) and identification of an intensified heat-exchanger/reactor. Extended high gain observers are adopted for identifying sensor faults and guaranteeing accurate dynamics since they can simultaneously estimate both states and uncertain parameters. Uncertain parameters involve overall heat transfer coefficient in this paper. Meanwhile, in the proposed algorithm, an extended high gain observer is fed by only one measurement. In this way, observers are allowed to act as soft sensors to yield healthy virtual measures for faulty physical sensors. Then, healthy measurements, together with a bank of parameter interval filters are processed, aimed at isolating process faults and identifying faulty values. Effectiveness of the proposed approach is demonstrated on an intensified heat-exchanger/ reactor developed by the Laboratoire de Génie Chimique, Toulouse, France.

1 Introduction

Nowadays, safety is a priority in the design and development of chemical processes. Large research efforts contributed to the improvement of new safety tools and methodology. Process intensification can be considered as an inherently safer design such as intensified heat exchangers (HEX) reactors in [1], the prospects are a drastic reduction of unit size and solvent consumption while safety is increased due to their remarkable heat transfer capabilities. However, risk assessment presented in [2] shows that potential risk of thermal runaway exists in such intensified process. Further, several kinds of failures may compromise safety and productivity: actuator failures (e.g., pump failures, valves failures), process failures (e.g., abrupt variations of some process parameters) and sensor failures. Therefore, supervision like FDI is required prior to the implementation of an intensified process.

For complex systems (e.g. heat-exchanger/reactors), fault detection and isolation are more complicated for the reason that some sensors cannot be placed in a desirable place, and for some variables (concentrations), no sensor exists. In addition, complete state and parameters measurements (i.e. overall heat transfer coefficient) are usually not available.

Supervision studies in chemical reactors have been reported in the literature concerning process monitoring, fouling detection, fault detection and isolation. Existing approaches can be roughly divided into data based method as in [3], neural networks as in [4] and model based method as in [5,6,7,8,9]. Among the model based approach, observer based methods are said to be the most capable [10,11,12,13,14] if analytical models are available.

Most of previous approaches focus on a particular class of failures. This paper deals with integrated fault diagnosis for both sensor and process failures. Using temperature measurements, together with state observers, an integrated diagnosis scheme is proposed to detect, isolate and identify faults. As for sensor faults, a FDI framework is proposed based on the extended observer developed in [15]. Extended high gain observers are adopted in this paper due to its capability of simultaneous estimation of both states and parameters, resulting in more accurate system dynamics. The estimates information provided by the observers and the sensors measurements are processed so as to recognize the faulty physical sensors, thus achieving sensor FDI. Moreover, the extended high gain observers will work as soft sensors to output healthy virtual measurements once there are sensor faults occurred. Then, the healthy measures are utilized to feed a bank of parameter intervals filters developed in [11] to generate a bank of residuals. These residuals are processed for isolating and identifying process faults which involves jumps in overall heat transfer coefficient in this work.

It should be pointed out that the contribution of this work does not lie with the soft sensor design or the parameter interval filter design as either part has individually already been addressed in the existing literature. However, the authors are not aware of any studies where both tasks are combined for integrated FDI, besides, there is no report whereby parameter estimation capacity of the extended high gain observer is used to adapt the coefficient, rather than parameter FDI, thus together with sensor FDI framework forms the contribution of this work.

2 System modelling

2.1 Process description

The key feature of the studied intensified continuous heat-exchanger/reactor is an integrated plate heat-exchanger

technology which allows for the thermal integration of several functions in a single device. Indeed, by combining a reactor and a heat exchanger in only one unit, the heat generated (or absorbed) by the reaction is removed (or supplied) much more rapidly than in a classical batch reactor. As a consequence, heat exchanger/reactors may offer better safety (by a better thermal control of the reaction), better selectivity (by a more controlled operating temperature).

2.2 Dynamic model

Supervision like FDI study can be much more efficient if a dynamic model of the system under consideration is available to evaluate the consequences of variables deviations and the efficiency of the proposed FDI scheme.

Generally speaking, intensified continuous heat-exchanger/reactor is treated as similar to a continuous reactor [16,17], then flow modelling is therefore based on the same hypothesis as the one used for the modelling of real continuous reactors, represented by a series of N perfectly stirred tank reactors (cells). According to [18], the number of cells N should be greater than the number of heat transfer units, and the heat transfer units is related with heat capacity flowrate. The modelling of a cell is based on the expression of balances (mass and energy) which describes the evolution of the characteristic values: temperature, mass, composition, pressure, etc. Given the specific geometry of the heat-exchanger/reactor, two main parts are distinguished. The first part is associated with the reaction and the second part encompasses heat transfer aspect. Without reaction, the basic mass balance expression for a cell is written as:

{Rate of mass flow in – Rate of mass flow out = Rate of change of mass within system}

The state and evolutions of the homogeneous medium circulating inside cell k are described by the following balance:

2.2.1 Heat balance of the process fluid ($J \cdot s^{-1}$)

$$\rho_p^k V_p^k C_{p,p}^k \frac{dT_p^k}{dt} = h_p^k A^k (T_p^k - T_u^k) + \rho_p^k F_p^k C_{p,p}^k (T_p^{k-1} - T_p^k) \quad (1)$$

where ρ_p^k is density of the process fluid in cell k (in $kg \cdot m^{-3}$), V_p^k is volume of the process fluid in cell k (in m^3), $C_{p,p}^k$ specific heat of the process fluid in cell k (in $J \cdot kg^{-1} \cdot K^{-1}$), h_p^k is the overall heat transfer coefficient (in $J \cdot m^{-2} \cdot K^{-1} \cdot s^{-1}$).

2.2.2 Heat balance of the utility fluid ($J \cdot s^{-1}$)

$$\rho_u^k V_u^k C_{p,u}^k \frac{dT_u^k}{dt} = h_u^k A^k (T_u^k - T_n^k) + \rho_u^k F_u^k C_{p,u}^k (T_u^{k-1} - T_u^k) \quad (2)$$

where ρ_u^k is density of the utility fluid in cell k (in $kg \cdot m^{-3}$), V_u^k is volume of the utility fluid in cell k (in m^3), $C_{p,u}^k$ specific heat of the utility fluid in cell k (in $J \cdot kg^{-1} \cdot K^{-1}$), h_u^k is overall heat transfer coefficient (in $J \cdot m^{-2} \cdot K^{-1} \cdot s^{-1}$).

The eq. (1) (2) represent the dynamic reactor compartment. The two equations represent the evolution of two states (T_p : reactor temperature and T_u : utility fluid temperature). The heat transfer coefficient (h) is considered as a variable which undergoes either an abrupt jumps (by an expected fault in the process) or a gradual variation (essentially due to degradation). The degradation can be attributed to fouling. Fouling in intensified process is tiny due to the micro

channel volume and cannot be a failure leads to fatal accident normally, but it may influence the dynamic of the process and it is rather difficult to calculate the changes online. In this paper, we treat the parameter uncertainty as an unmeasured state, and employ an observer as soft sensor to estimate it, unlike other literature, the estimation here is not for fouling detection but for more accurate model dynamics, and to ensure the value of the variable is within acceptable parameter, (e.g., upper and lower bounds of the process variable value).

To rewrite the whole model in the form of state equations, due to the assumption that every element behaves like a perfectly stirred tank, we suppose that one cell can keep the main feature of the qualitative behavior of the reactor. For the sake of simplicity, only one cell has been considered. Let us delete the subscript k for a given cell.

Define the state vector as $x_1^T = [x_{11}, x_{12}]^T = [T_p, T_u]^T$, unmeasured state $x_2^T = [x_{21}, x_{22}]^T = [h_u, h_p]^T$, $\frac{dh_p}{dt} = \frac{dh_u}{dt} = \varepsilon(t)$, $\varepsilon(t)$ is an unknown but bounded function refers to variation of h , the control input $u = T_{ui}$, the output vector of measurable variables $y^T = [y_1, y_2]^T = [T_p, T_u]^T$, then the equation (1) and (2) can be rewritten in the following state-space form:

$$\begin{cases} \dot{x}_1 = F_1(x_1)x_2 + g_1(x_1, u) \\ \dot{x}_2 = \varepsilon(t) \\ y = x_1 \end{cases} \quad (3)$$

$$\text{Where, } F_1(x_1) = \begin{pmatrix} \frac{A}{\rho_p C_{p,p} V_p} (T_p - T_u) & 0 \\ 0 & \frac{A}{\rho_u C_{p,u} V_u} (T_u - T_p) \end{pmatrix},$$

$$\text{and } g_1(x) = \begin{pmatrix} \frac{(T_{pi} - T_p)F_p}{V_p} \\ \frac{(T_{ui} - T_u)F_u}{V_u} \end{pmatrix}, T_{pi}, T_{ui} \text{ is the output of previ-}$$

ous cell, for the first cell, it is the inlet temperature of process fluid and utility fluid.

In this case, the full state of the studied system is given as:

$$\begin{cases} \dot{x} = F(x_1)x + G(x_1, u) + \bar{\varepsilon}(t) \\ y = Cx \end{cases} \quad (4)$$

$$\text{Where } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, F(x_1) = \begin{pmatrix} 0 & F_1(x_1) \\ 0 & 0 \end{pmatrix}, G(x_1, u) = \begin{pmatrix} g_1(x, u) \\ 0 \end{pmatrix}, C = \begin{pmatrix} I & 0 \end{pmatrix}, \bar{\varepsilon}(t) = \begin{pmatrix} 0 \\ \varepsilon(t) \end{pmatrix}$$

3 Fault detection and diagnose scheme

3.1 Observer design for sensor FDI

The extended high gain observer proposed by [15] can be used like an adaptive observer for estimation both states and parameters simultaneously, in this paper, the latter capability is utilized to estimate incipient degradation of overall heat transfer coefficient (due to fouling), thus guaranteeing a more accurate approximation of the temperature. It is quite useful in chemical processes since parameters are usually with uncertainties and unable to be measured.

Consider a nonlinear system as the form:

$$\begin{cases} \dot{x} = F(x_1)x + G(x_1, u) \\ y = Cx \end{cases} \quad (5)$$

where $x = (x_1, x_2)^T \in \mathcal{R}^{2n}$, $x_1 \in \mathcal{R}^n$ is the state, $x_2 \in \mathcal{R}^n$ is the unmeasured state, $x_2 = \epsilon(t)$, $u \in \mathcal{R}^m$, $y \in \mathcal{R}^p$ are input and output, $\epsilon(t)$ is an unknown bounded function which may depend on $u(t)$, $y(t)$, noise, etc., and

$$F(x_1) = \begin{pmatrix} 0 & F_1(x_1) \\ 0 & 0 \end{pmatrix}, G(x_1, u) = \begin{pmatrix} g_1(x, u) \\ 0 \end{pmatrix}, C(I \ 0),$$

$F_1(x_1)$ is a nonlinear vector function, $g_1(x, u)$ is a matrix function whose elements are nonlinear functions.

Supposed that assumptions related boundedness of the states, signals, functions etc. in [15] are satisfied, the extended high gain observer for the system can be given by:

$$\begin{cases} \dot{\hat{x}} = F(\hat{x}_1)x + G(\hat{x}_1, u) - \Lambda^{-1}(\hat{x}_1)S_\theta^{-1}C^T(\hat{y} - y) \\ \dot{\hat{y}} = C\hat{x} \end{cases} \quad (6)$$

Where: $\Lambda(\hat{x}_1) = \begin{bmatrix} I & 0 \\ 0 & F_1(\hat{x}_1) \end{bmatrix}$

S_θ is the unique symmetric positive definite matrix satisfying the following algebraic Lyapunov equation:

$$\theta S_\theta + A^T S_\theta + S_\theta A - C^T C = 0 \quad (7)$$

Where $A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$, $\theta > 0$ is a parameter define by [15] and the solution of eq. (7) is:

$$S_\theta = \begin{bmatrix} \frac{1}{\theta} I & -\frac{1}{\theta^2} I \\ -\frac{1}{\theta^2} I & \frac{2}{\theta^3} I \end{bmatrix} \quad (8)$$

Then, the gain of estimator can be given by:

$$H = \Lambda^{-1}(\hat{x}_1)S_\theta^{-1}C^T = \Lambda(\hat{x}_1) \begin{bmatrix} 2\theta I \\ \theta^2 F_1^{-1}(\hat{x}_1) \end{bmatrix} \quad (9)$$

Notice that larger θ ensures small estimation error. However, very large values of θ are to be avoided in practice due to noise sensitiveness. Thus, the choice of θ is a compromise between fast convergence and sensitivity to noise.

3.2 Sensor fault detection and isolation scheme

The above observer could guarantee the heat-exchanger/reactor dynamics ideally. Then, a bank of the proposed observers, together with sensor measurements, are used to generate robust residuals for recognizing faulty sensor. Thus, we propose a FDI scheme to detect, meanwhile, isolate and recovery the sensor fault.

3.2.1 Sensor faulty model

A sensor fault can be modeled as an unknown additive term in the output equation. Supposed θ_{sj} is the actual measured output from j th sensor, if j th sensor is healthy, $\theta_{sj} = y_j$, while if j th sensor is faulty, $\theta_{sj} = y_j^f = y_j + f_{sj}$, (f_{sj} is the fault), for $t \geq t_f$ and $\lim_{t \rightarrow \infty} |y_j - \theta_{sj}| \neq 0$. That means y_j^f is the actual output of the j th sensor when it is faulty, while y_j is the expected output when it is healthy, that is:

$$\theta_{si} = \begin{cases} y_i; & j\text{th sensor when it is faulty} \\ y_i^f = y_i + f_{si}; & j\text{th sensor when it is faulty} \end{cases} \quad (10)$$

With this formulation, the faulty model becomes:

$$\begin{cases} \dot{x} = F(x_1)x + G(x_1, u) + \bar{\epsilon}(t) \\ y = Cx + F_s f_s \end{cases} \quad (11)$$

F_s is the fault distribution matrix and we consider that fault vector $f_s \in \mathcal{R}^p$ (f_{sj} is the j th element of the vector) is also a bounded signal. Notice that, a faulty sensor may lead to incorrect estimation of parameter. That is why we emphasized healthy measurement for parameter fault isolation as mentioned above.

3.2.2 Fault detection and isolation scheme

The proposed sensor FDI framework is based on a bank of observers, the number of observers is equal to the number of sensors. Each observer use only one sensor output to estimate all the states and parameters. First, assumed the sensor used by i th observer is healthy, let y_i denotes the i th system output used by the i th observer. Then we form the observer as:

$$1 \leq i \leq p \begin{cases} \dot{\hat{x}}^i = F(\hat{x}_1^i)x + G(\hat{x}_1^i, u) + H_i(y_i - \hat{y}_i^i) \\ \dot{\hat{y}}^i = C\hat{x}^i \end{cases} \quad (12)$$

Define $e_x^i = \hat{x}^i - x$, $e_y^i = Ce_x^i$, $e_{y_j}^i = \hat{y}_j^i - y_j$, $r_j^i(t) = \|\hat{y}_j^i - y_j\|$, $\mu_i = \|r_j^i(t)\| := \sup \|r_j^i(t)\|$, for $t \geq 0$.

Where i denotes the i th observer, \hat{y}_i^i, \hat{y}_j^i denotes the i th, j th estimated system output generated by the i th observer, H_i is the gain of i th observer determined by the following equation :

$$H_i = \Lambda^{-1}(\hat{x}_1)S_{\theta_i}^{-1}C^T = \Lambda(\hat{x}_1) \begin{bmatrix} 2\theta_i I \\ \theta_i^2 F_1^{-1}(\hat{x}_1) \end{bmatrix}$$

Then we get:

Theorem 1:

If the l th sensor is faulty, then for system of form (4), the observer (12) has the following properties:

For $i \neq l$, $\hat{y}^i = y$ asymptotically

For $i = l$, $\hat{y}^i \neq y$

Proof: If the l th sensor is faulty, then:

For $i \neq l$, means that $f_{si} = 0$, $y_i = \theta_{si}$, we have:

$$\lim_{t \rightarrow \infty} e_x^i = \lim_{t \rightarrow \infty} (\hat{x}^i - x) = 0 \quad (13)$$

Then the vector of the estimated output \hat{y}^i generated by i th observer guarantee $\hat{y}^i = y$ after a finite time.

For $i = l$, means that $\theta_{sl} = y_l^f = y_l + f_{sl}$, $f_{sl} \neq 0$, the observer is designed on the assumption that there is no fault occurs, because there is fault f_{sl} exit, so the estimation error $e_x^l = 0$ asymptotically cannot be satisfied, then :

$$\lim_{t \rightarrow \infty} (\hat{x}^l - x) = \lim_{t \rightarrow \infty} (\hat{x}^l - x) \neq 0 \quad (14)$$

we have:

$$e_x^l = F(\hat{x}_1^l, u) e_x^l - H_l G(\hat{x}_1^l, u, f_{sl}) e_x^l \quad (15)$$

Then the vector of the estimated output \hat{y}^l generated by the i th observer is different from y , that is $\hat{y}^l \neq y$. \square

As mentioned above, the observers are deigned under the assumption that no fault occurs, furthermore, each observer just subject to one sensor output. Residual r_l^l is the difference between the i th output estimation \hat{y}_i^l determined by the i th observer and the i th system output y_i , then Theorem 2 formulates the fault detection and isolation scheme.

Theorem 2:

If the l th sensor is faulty, then:

For $i \neq l$, we have:

$$f_{si} = 0, y_i = \theta_{si} \quad (16)$$

thus \hat{y}_i^i converges to y_i asymptotically, we get:

$$r_i^i = \|\hat{y}_i^i - y_i\| \leq \mu_i \quad (17)$$

For $i = l$, we have:

$f_{sl} \neq 0, \theta_{sl} = y_l^f = y_l + f_{sl} \neq y_l$, then \hat{y}_l^l could not track y_l correctly:

$$r_l^l = \|\hat{y}_l^l - y_l\| \geq \mu_l \quad (18)$$

Therefore, in practice, we can check all the residuals r_i^i , for $1 \leq i \leq p$, if $r_i^i \geq \mu_i$ denotes that i th sensor is faulty, then the sensor fault detection and isolation is achieved.

The residuals are designed to be sensitive to a fault that comes from a specific sensor and as insensitive as possible to all the others sensor faults. This residual will permit us to treat not only with single faults but also with multiple and simultaneous faults.

Let r_{si} denotes the fault signature of the i th sensor, define:

$$r_{si}(t) = \begin{cases} 1 & \text{if } r_i^i \geq \mu_i; \text{ ith sensor is faulty} \\ 0 & \text{if } r_i^i \leq \mu_i; \text{ ith sensor is health} \end{cases} \quad (19)$$

3.2.3 Fault identification and handling mechanism

1) Fault identification

Supposed there are m healthy sensors and $p - m$ faulty ones, then to identify the faulty size of i th sensor, use m estimated output \hat{y}_i^m generated by m observers which use healthy measures, $1 \leq m \leq p - 1, m \neq i$, define \hat{f}_{si} as the estimated faulty value of the i th sensor, then:

$$\hat{f}_{si} = \frac{1}{m} \sum_{i=1}^m |\hat{y}_i^m - \theta_{si}| \xrightarrow{\Delta} f_{si} \quad (20)$$

2) Fault recovery

As mentioned above, the extended high gain observer is also worked as a software sensor to provide an adequate estimation of the process output, thus replacing the measurement given by faulty physical sensor.

θ_{si} is the actual measured output from i th sensor:

$$\theta_{si} = \begin{cases} y_i \\ y_i^f = y_i + f_{si} \end{cases} \quad (21)$$

Let m observers use healthy measurements as the soft sensor for i th sensor, define:

$$\bar{y}_i = \frac{1}{m} \sum_{i=1}^m \hat{y}_i^m \quad (22)$$

If i th sensor is healthy, let the sensor actual output as θ_{si} its output, while if it is faulty, let \bar{y}_i to replace θ_{si} , that is:

$$y_i = \begin{cases} \theta_{si}, & \text{if ith sensor healthy} \\ \bar{y}_i, & \text{if ith sensor faulty} \end{cases} \quad (23)$$

3.3 process fault diagnose

In order to achieve process FDD, healthy measurements are fed to a bank of parameter intervals filters developed in [11] to generate a bank of residuals. These residuals are processed for identifying parameter changes, which involves variation of overall heat transfer coefficient in this paper. The main idea of the method is as follows.

The practical domain of the value of each system parameter is divided into a certain number of intervals. After verifying

all the intervals whether or not one of them contains the faulty parameter value of the system, the faulty parameter value is found, the fault is therefore isolated and estimated. The practical domain of each parameter is partitioned into a certain number of intervals. For example, parameter h_p is partitioned into q intervals, their bounds are denoted by $h_p^{(0)}, h_p^{(1)}, \dots, h_p^{(i)}, \dots, h_p^{(q)}$. The bounds of i th interval are $h_p^{(i-1)}$ and $h_p^{(i)}$, are also noted as h_p^{bi} and h_p^{ai} , and the nominal value for h_p denotes by h_{p0} .

To verify if an interval contains the faulty parameter value of the post-fault system, a parameter filter is built for this interval. A parameter filter consists of two isolation observers which correspond to two interval bounds, and each isolation observer serves two neighboring intervals. An interval which contains a parameter nominal value is unable to contain the faulty parameter value, so a parameter filter will not be built for it.

Define Eq. (3) into a simple form as:

$$\begin{cases} \dot{x}_1 = F_1(x_1)x_2 + g_1(x_1, u) \\ y = x_1 \end{cases} = \begin{cases} \dot{x}_1 = f(x_1, h_p, u) \\ y = x_1 \end{cases} \quad (24)$$

The parameter filter for i th interval of h_p is given below.

The isolation observers are:

$$\begin{cases} \dot{\hat{x}}^{ai} = f(\hat{x}_1, h_{p0}^{ai}, u) + H(y - \hat{y}^{ai}) \\ \dot{\hat{y}}^{ai} = c\hat{x}^{ai} \\ \varepsilon^{ai} = y - c\hat{x}^{ai} \end{cases} \quad (25)$$

$$\begin{cases} \dot{\hat{x}}^{bi} = f(\hat{x}_1, h_{p0}^{bi}, u) + H(y - \hat{y}^{bi}) \\ \dot{\hat{y}}^{bi} = c\hat{x}^{bi} \\ \varepsilon^{bi} = y - h\hat{x}^{bi} \end{cases} \quad (26)$$

Where:

$$h_{p0}^{ai}(t) = \begin{cases} h_{p0}, & t < t_f \\ h_p^{(i)}, & t \geq t_f \end{cases}, h_{p0}^{bi}(t) = \begin{cases} h_{p0}, & t < t_f \\ h_p^{(i-1)}, & t \geq t_f \end{cases}, \quad (27)$$

The isolation index of this parameter filter is calculated by:

$$v^i(t) = \text{sgn}(\varepsilon^{ai})\text{sgn}(\varepsilon^{bi}) \quad (28)$$

As soon as $v^i(t) = 1$, the parameter filter sends the 'non-containing' signal to indicate that this interval does not contain the faulty parameter value. And if the fault is in the i th interval. Let:

$$\hat{h}A = \frac{1}{2}(h^{ai}A + h^{bi}A) \quad (29)$$

to represent the faulty value, fault isolation and identification is then achieved.

4 Numerical simulation

A case study is developed to test the effectiveness of the proposed scheme. The real data is from a laboratory pilot of a continuous intensified heat-exchanger/reactor. The pilot is made of three process plates sandwiched between five utility plates, shown in Fig.1. More Relative information could be found in [2]. As previously said, the simulation model is considered just for one cell which may lead to moderate inaccuracy of the dynamic behavior of the realistic reactor. However, this point may not affect the application and demonstration of the proposed FDD algorithm encouraging results are got.



Figure 1 (a) Reactive channel design; (b) utility channel design; (c) the heat exchanger/reactor after assembly. The constants and physical data used in the pilot are given in table1.

Table 1. Physical data used in the pilot

Constant	Value	units
hA	214.8	W.K ⁻¹
A	4e ⁻⁶	m ³
V _p	2.685e ⁻⁵	m ³
V _u	1.141e ⁻⁴	m ³
ρ _p , ρ _u	1000	kg.m ⁻³
c _{p,p} , c _{p,u}	4180	J.kg ⁻¹ .k ⁻¹

4.1 operation conditions

The inlet fluid flow rate in utility fluid and process fluid are $F_u = 4.17e^{-6}m^3$, $F_p = 4.22e^{-5}m^3s^{-1}$. The inlet temperature in utility fluid is time-varying between 15.6°C and 12.6°C, which is a classical disturbance in the studied system, as shown in Fig.2. The inlet temperature in process fluid is 76°C. Initial condition for all observers and models are supposed to be $\hat{T}_p^0 = \hat{T}_u^0 = 30^\circ C$, $hA = 214.8 W.K^{-1}$.

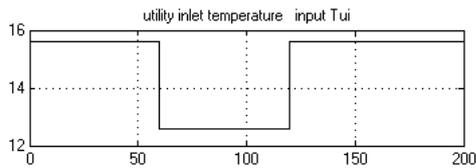


Fig.2 utility inlet temperature T_{ui}

4.2 High gain observer performance

To prove the convergence of the observers and show their tracking capabilities, suppose the heat transfer coefficient subjects to a decreasing of $h = (1 - 0.01t)h$ and follows by a sudden jumps of 15 at $t = 100s$. These variations and observer estimation results are reported in Fig.3.

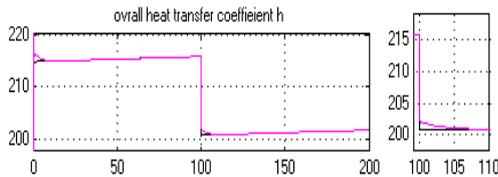


Fig.3. simulation and estimation of heat transfer coefficient variation.

Black curve simulates the actual changes of the parameter while the red one illustrates the estimation generated by the proposed observer, it can be seen from Fig. 3 that the estimation value tracks behavior of the real value with a good accuracy, thus ensuring a good dynamics.

4.3 Sensor FDI and recovery demonstration

In order to show effectiveness of the proposed method on sensor FDI, multi faults and simultaneous faults in the temperature sensors are considered in case 1 and case 2 respectively. Besides, the pilot is suffered to parameter uncertainties caused by heat transfer coefficient decreases with $h = (1 - 0.01t)h$. Two extended high gain observers are designed to generate a set of residuals achieving fault detection and isolation in individual sensors. Observer 1 is fed by output of sensor T_p to estimate the whole states and parameter while observer 2 uses output of sensor T_u . Advantages of the proposed FDI methodology drop on that if one sensor is faulty, we can use the estimated value generated by the healthy one to replace the faulty physical value, thus providing a healthy virtual measure.

Case 1: abrupt faults occur at output of sensor T_p at $t=80s$, $100s$, with an amplitude of 0.3°C, 0.5°C respectively. the results are reported in Fig.5-8.

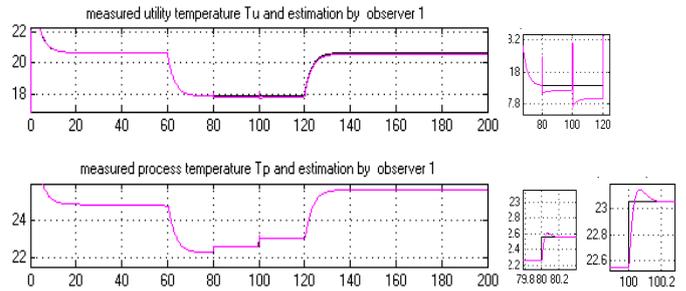


Fig. 5 output temperature of both fluid in case 1 by observer 1, red curve demonstrates the estimated value while black one is the measured value.

It is obviously that since $t=80s$, \hat{T}_u (red curve) cannot track T_u (black curve) correctly, while it needs about 0.2s for \hat{T}_p to track T_p at $t=80s$ and $t=100s$. It suggests that faults occur, then the following task is to identify size and location of faulty sensors. Fig.6 and Fig.7 achieves the goal. It takes 0.1s and 0.3s for isolating the faults at 80s, 100s respectively.

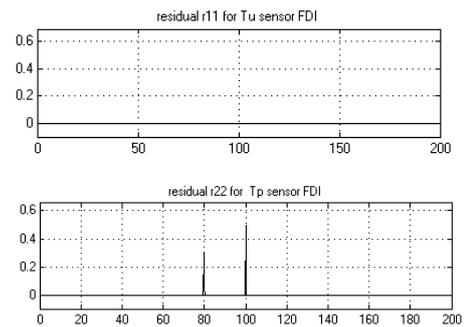
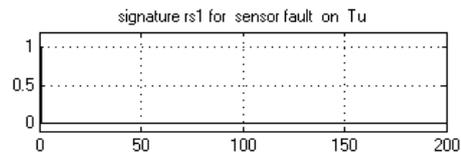


Fig.6 isolation residual in case 1.



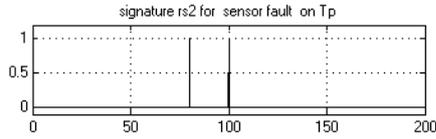


Fig.7b fault signature in case 1, obviously, faults only occur at output of sensor T_p .

For fault recovery, we can employ observer 2 as soft sensor to generate a health value for faulty sensor T_p . Observer 2 uses only measured T_u to estimate all states and parameters. Therefore, \hat{T}_u, \hat{T}_p generated by observer 2 are only decided by T_u . In case 1, faults occur only on sensor T_p , sensor T_u is healthy, that is to say \hat{T}_u, \hat{T}_p generated by observer 2 will be satisfied their expected values. As shown in Fig.8, we can see that since T_u is healthy, estimated value \hat{T}_u tracks measured T_u perfectly, while estimated value \hat{T}_p (red curve) does not track the faulty measured value T_p (black curve), \hat{T}_p (red curve) illustrates the expected value for sensor T_p , we can use estimate \hat{T}_p (red curve) to replace measured faulty value T_p (black curve) for fault recovery.

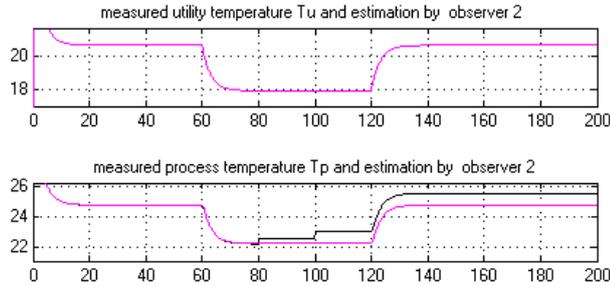


Fig.8 fault recovery in case 1, red curve demonstrates the estimated value while black one is the measured value.

If there are faults occurred only on output of sensor T_u , the same results can be yield easily. For multi and simultaneous faults on both sensors, we can still isolate the faults correctly. Case 2 will verify this point.

Case 2: simultaneous faults imposed to the outputs of sensors T_p as in case 1 and T_u at $t=80s$ with amplitude of $0.6^\circ C$. Results are reported in Fig.9-10. Residuals are beyond their threshold obviously at time 80s, 100s.

It can be seen from Fig.9, Fig .10 that the proposed FDI scheme can isolate faults correctly, and it takes 0.25s, 0.4s for isolating the faults in sensor T_p at 80s, 100s and 0.2s for isolating that in sensor T_u at $t=80s$ respectively. Compared with Case 1, more times is needed in this Case 2.

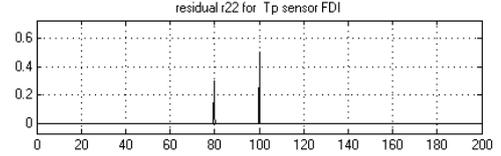
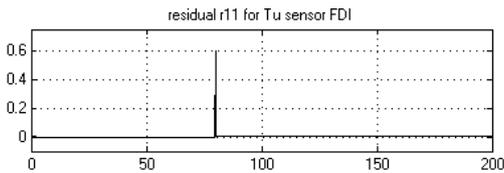


Fig. 9 isolation residual in case 2

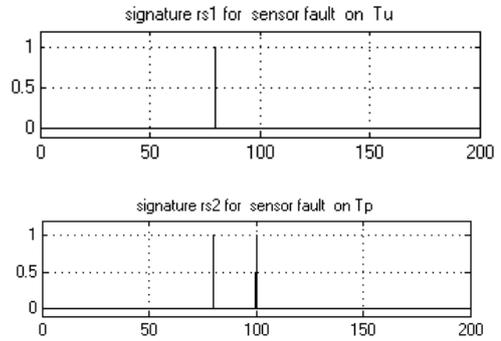


Fig 10. Fault signature in case 2

4.4 Fast process fault isolation and identification

Process fault is related to variation of overall heat transfer coefficient (h). The heat transfer coefficient is considered as variable which undergoes either an abrupt jumps (by an expected fault in the flow rate) or a gradual variation (essentially due to fouling). For incipient variation, since fouling in intensified heat-exchanger/reactor is tiny and only influence dynamics, we have employed extended high observers to ensure the dynamic influenced by this slowly variation. Therefore, the abrupt changes in heat transfer coefficient h can only be because of sudden changes in mass flow rate. It implies that the root cause of process fault is due to actuator fault in this system.

Supposed an abrupt jumps in h at $t=40$ from 214.8 to 167.

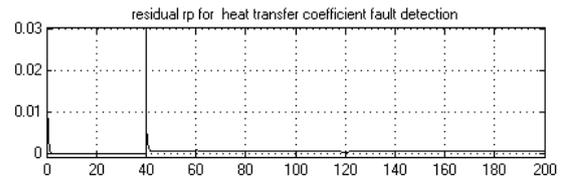


Fig.11 detection residual in process faulty case

From Fig.11, at $t=40s$, unlike sensor fault cases, the residual leaves zero and never goes back, this indicates that process fault occurs. For fast fault isolation and identification, we use the methodology of parameter interval filters developed in [11]. In [2], heat transfer coefficient h changes between 130.96 and 214.8, then h is divided into 4 intervals as shown in table 2 and simulation results are shown in Fig.12. It can be seen at $t=40s$, only index for interval 150-170 goes to zero rapidly, then there is a fault in this interval. The faulty value is estimated by $\hat{h}A = \frac{1}{2}(h^a A + h^b A) = \frac{1}{2}(150 + 170) = 160$. We can see it is closely to actual faulty value 167, and if more intervals are divided, the estimated value may be closer to the actual faulty value.

Table 2 parameter filter intervals

Interval NO.	1	2	3	3
$h^a A$	130	150	170	190
$h^b A$	150	170	190	214

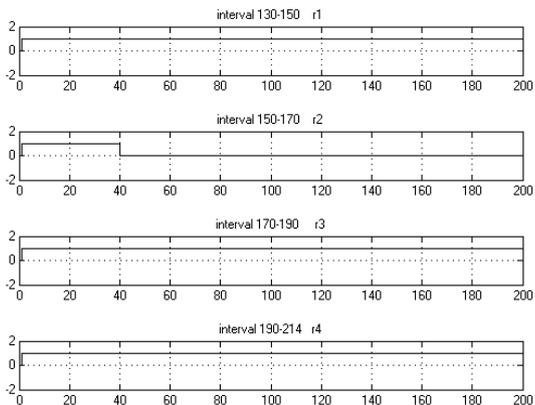


Fig.12 “non_containing fault” index sent by parameter filter

5 Conclusion

An integrated approach for fault diagnose in intensified heat-exchange/reactor has been developed in this paper. The approach is capable of detecting, isolating and identifying failures due to both sensors and parameters. Robustness of the proposed FDI for sensors is ensured by adopting a soft sensor with respect to parameter uncertainties. Ideal isolation speed for process fault is guaranteed due to adoption of parameter interval filter. It should be notice that the proposed method is suitable for a large kind of nonlinear systems with dynamics models as the studied system. Application on the pilot heat-exchange/reactor confirms the effectiveness and robustness of the proposed approach.

References

- [1] F. Théron, Z. Anxionnaz-Minvielle, M. Cabassud, C. Gourdon, and P. Tochon, “Characterization of the performances of an innovative heat-exchanger/reactor,” *Chem. Eng. Process. Process Intensif.*, vol. 82, pp. 30–41, 2014.
- [2] N. Di Miceli Raimondi, N. Olivier-Maget, N. Gabas, M. Cabassud, and C. Gourdon, “Safety enhancement by transposition of the nitration of toluene from semi-batch reactor to continuous intensified heat exchanger reactor,” *Chem. Eng. Res. Des.*, vol. 94, pp. 182–193, 2015.
- [3] P. Kesavan and J. H. Lee, “A set based approach to detection and isolation of faults in multivariable systems,” *Chem. Eng.*, vol. 25, pp. 925–940, 2001.
- [4] D. Ruiz, J. M. Nougues, Z. Calderon, A. Espuna, L. Puigjaner, J. Maria, Z. Caldern, and A. Espufia, “Neural network based framework for fault diagnosis in batch chemical plants,” *Comput. Chem. Eng.*, vol. 24, pp. 777–784, 2000.
- [5] O. a. Z. Sotomayor and D. Odloak, “Observer-based fault diagnosis in chemical plants,” *Chem. Eng. J.*, vol. 112, pp. 93–108, 2005.
- [6] M. Du and P. Mhaskar, “Isolation and handling of sensor faults in nonlinear systems,” *Automatica*, vol. 50, no. 4, pp. 1066–1074, 2014.
- [7] F. Xu, Y. Wang, and X. Luo, “Soft sensor for inputs and parameters using nonlinear singular state observer in chemical processes,” *Chinese J. Chem. Eng.*, vol. 21, no. 9, pp. 1038–1047, 2013.
- [8] F. Caccavale and F. Pierri, “An integrated approach to fault diagnosis for a class of chemical batch processes,” *J. Process Control*, vol. 19, no. 5, pp. 827–841, 2009.
- [9] M. Du, J. Scott, and P. Mhaskar, “Actuator and sensor fault isolation of nonlinear proces systems,” *Chem. Eng. Sci.*, vol. 104, pp. 2940–303, 2013.
- [10] D. Fragkoulis, G. Roux, and B. Dahhou, “Detection, isolation and identification of multiple actuator and sensor faults in nonlinear dynamic systems: Application to a waste water treatment process,” *Appl. Math. Model.*, vol. 35, no. 1, pp. 522–543, 2011.
- [11] Z. Li and B. Dahhou, “A new fault isolation and identification method for nonlinear dynamic systems: Application to a fermentation process,” *Appl. Math. Model.*, vol. 32, pp. 2806–2830, 2008.
- [12] X. Zhang, M. M. Polycarpou, and T. Parisini, “Fault diagnosis of a class of nonlinear uncertain systems with Lipschitz nonlinearities using adaptive estimation,” *Automatica*, vol. 46, no. 2, pp. 290–299, 2010.
- [13] R. F. Escobar, C. M. Astorga-Zaragoza, J. a. Hernández, D. Juárez-Romero, and C. D. García-Beltrán, “Sensor fault compensation via software sensors: Application in a heat pump’s helical evaporator,” *Chem. Eng. Res. Des.*, pp. 2–11, 2014.
- [14] F. Bonne, M. Alamir, and P. Bonnay, “Nonlinear observer of the thermal loads applied to the helium bath of a cryogenic Joule–Thompson cycle,” *J. Process Control*, vol. 24, no. 3, pp. 73–80, 2014.
- [15] M. Farza, K. Busawon, and H. Hammouri, “Simple nonlinear observers for on-line estimation of kinetic rates in bioreactors,” *Automatica*, vol. 34, no. 3, pp. 301–318, 1998.
- [16] W. Benaïssa, N. Gabas, M. Cabassud, D. Carson, S. Elgue, and M. Demissy, “Evaluation of an intensified continuous heat-exchanger reactor for inherently safer characteristics,” *J. Loss Prev. Process Ind.*, vol. 21, pp. 528–536, 2008.
- [17] S. Li, S. Bahroun, C. Valentin, C. Jallut, and F. De Panthou, “Dynamic model based safety analysis of a three-phase catalytic slurry intensified continuous reactor,” *J. Loss Prev. Process Ind.*, vol. 23, no. 3, pp. 437–445, 2010.
- [18] P. S. Varbanov, J. J. Klemeš, and F. Friedler, “Cell-based dynamic heat exchanger models-Direct determination of the cell number and size,” *Comput. Chem. Eng.*, vol. 35, pp. 943–948, 2011.

LPV subspace identification for robust fault detection using a set-membership approach: Application to the wind turbine benchmark

Chouiref. H, Boussaid. B, Abdelkrim. M.N¹, Puig. V² and Aubrun.C³

¹Research Unit of Modeling, Analysis and Control of Systems (MACS), Gabès University
e-mail: houda.chouiref@gmail.com, dr.boumedyen.boussaid@ieee.org, naceur.abdelkrim@enig.rnu.tn

²Advanced Control Systems Group (SAC), Technical University of Catalonia
e-mail: vicenc.puig@upc.edu

³Centre de Recherche en Automatique de Nancy (CRAN), Lorraine University
e-mail: christophe.aubrun@univ-lorraine.fr

Abstract

This paper focuses on robust fault detection for Linear Parameter Varying (LPV) systems using a set-membership approach. Since most of models which represent real systems are subject to modeling errors, standard fault detection (FD) LPV methods should be extended to be robust against model uncertainty. To solve this robust FD problem, a set-membership approach based on an interval predictor is used considering a bounded description of the modeling uncertainty. Satisfactory results of the proposed approach have been obtained using several fault scenarios in the pitch subsystem considered in the wind turbine benchmark introduced in IFAC SAFEPROCESS 2009.

1 Introduction

The fault diagnosis of industrial processes has become an important topic because of its great influence on the operational control of processes. Reliable diagnosis and early detection of incipient faults avoid harmful consequences. Typically, faults in sensors and actuators and the process itself are considered. In the case of the wind turbine benchmark, a set of pre-defined faults with different locations and types are proposed in [1] where the dynamic change in the pitch system is treated. The procedure of fault detection is based either on the knowledge or on the model of the system [2]. Model-based fault detection is often necessary to obtain a good performance in the diagnosis of faults. The methods used in model-based diagnosis can be classified according if they are using state observers, parity equations and parameter estimation [3]. For linear time invariant systems (LTI), the FD task is largely solved by powerful tools. However, physical systems generally present nonlinear behaviors. Using LTI models in many real applications is not sufficient for high performance design. In order to achieve good performance while using linear like techniques, Linear Parameter Varying systems are recently received considerable attention [4]. Recently, many model-based applications using such systems and the subspace identification method were published [5]. In model-based FD, a residual vector is used to describe the consistency check between the predicted and the real behavior of the monitored system. Ideally, the residuals should only be affected by the faults. However, the presence of disturbances, noises and modeling errors yields the residual to become non zero. To take into account these errors, the fault detection algorithm

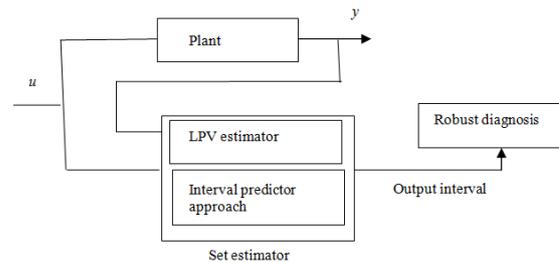


Figure 1: Fault diagnosis with set estimator schema

must be robust. When modeling uncertainty in a deterministic way, there are two robust estimation methods: the first method is the bounded error estimation that assumes the parameters are considered time invariant and there is only an additive error [6]. On the other hand, the second approach is the interval predictor that takes into account the variation of parameters and which considers additive and multiplicative errors [7], [8]. Here, the interval predictor is combined with existing nominal LPV identification presented by [9], allowing to include robustness and minimizing false alarms (see Fig. 1) [10]. Thus, this paper contributes with a new set-membership estimator approach that combines the interval predictor scheme with the LPV identification through subspace methods in one step. To illustrate the methodology proposed in this work, the pitch subsystem of wind turbine system proposed as a benchmark in IFAC SAFEPROCESS 2009 will be used. First, this subsystem is modeled as an LPV model using the hydraulic pressure as the scheduling variable. On the hypothesis that damping ratio and natural frequency have an affine variation with hydraulic pressure, this affine LPV model is estimated by means of the subspace LPV estimation algorithm. Second, the residue is synthesized to take into account the robustness against the uncertainties in the parameters. This work is organized as follows: In Section 2, the LPV subspace estimation method is recalled. In Section 3, the interval predictor approach combined with the LPV subspace method is proposed as tool for robust fault detection. In Section 4, the modeling of the pitch system as a LPV model is introduced. Section 5 deals with simulation experiments that illustrate the implementation and performance of the proposed approach applied to the robust fault detection of wind turbine pitch system. Fi-

nally, Section 6 gives some concluding remarks.

2 LPV Subspace Identification method

In the literature, there are two methods for LPV identification: First, the ones based on global LPV estimation. Second, the ones based on the interpolation of local models [11], However, those approaches could lead to unstable representations of the LPV structure while the original system is stable [12]. That is why in this paper, we propose to use a subspace identification algorithm proposed (see [9] and [13]) to identify LPV systems which does not require interpolation or identification of local models and avoid instability problems.

2.1 Problem formulation

In the model used in identification in [9], the system matrices depend linearly on the time varying scheduling vector as follows:

$$x_{k+1} = \sum_{i=1}^m \mu_k^{(i)} (A^{(i)} x_k + B^{(i)} u_k + K^{(i)} e_k) \quad (1)$$

$$y_k = C x_k + D u_k + e_k \quad (2)$$

with $x_k \in R^n$, $u_k \in R^r$, $y_k \in R^l$ are the state, input and output vectors and e_k denotes the zero mean white innovation process and m is the number of local model or scheduling parameters:

$$\mu_k = [1, \mu_k^{(2)}, \dots, \mu_k^{(m)}]^T$$

Eqs.(1) and (2) can be written in the predictor form:

$$x_{k+1} = \sum_{i=1}^m \mu_k^{(i)} (\tilde{A}^{(i)} x_k + \tilde{B}^{(i)} u_k + K^{(i)} y_k) \quad (3)$$

with

$$\begin{aligned} \tilde{A}^{(i)} &= A^{(i)} - K^{(i)} C \\ \tilde{B}^{(i)} &= B^{(i)} - K^{(i)} D \end{aligned}$$

2.2 Assumptions and notation

Defining $z_k = [u_k^T, y_k^T]^T$ and using a data window of length p to define the following vector:

$$\tilde{z}_k^p = \begin{bmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+p-1} \end{bmatrix}$$

and introducing the matrix obtained using the Kronecker product \otimes :

$$P_{p/k} = \mu_{k+p-1} \otimes \dots \otimes \mu_k \otimes I_{r+l}$$

we can define

$$N_k^p = \begin{bmatrix} p_{p/k} & \cdot & \cdot & \cdot & 0 \\ \cdot & p_{p-1/k+1} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & p_{1/k+p-1} \end{bmatrix}$$

Now, by defining the matrices U , Y and Z :

$$U = [u_{p+1}, \dots, u_N] \quad (4)$$

$$Y = [y_{p+1}, \dots, y_N] \quad (5)$$

$$Z = [N_1^p \tilde{z}_1^p, \dots, N_{N-p+1}^p \tilde{z}_{N-p+1}^p] \quad (6)$$

the controllability matrix can be expressed as:

$$\kappa^p = [l_p, \dots, l_1]$$

with

$$l_1 = [\bar{B}^{(1)}, \dots, \bar{B}^{(m)}]$$

and

$$l_j = [\tilde{A}^{(1)} l_{j-1}, \dots, \tilde{A}^{(m)} l_{j-1}]$$

If the matrix $[Z^T, U^T]$ has full row rank, the matrix $C \kappa^p$ and D can be estimated by solving the following linear regression problem [14]:

$$\min_{C \kappa^p, D} \|Y - C \kappa^p Z - D U\|_F^2 \quad (7)$$

where $\|\cdot\|_F$ represents the Frobenius norm. This problem can be solved by using traditional least square methods as in the case of LTI identification for time varying systems. Moreover, the observability matrix for the first model is calculated as follows:

$$\Gamma^p = \begin{bmatrix} C \\ C \tilde{A}^{(1)} \\ \vdots \\ C (\tilde{A}^{(1)})^{p-1} \end{bmatrix}$$

with

$$\bar{\kappa}_p^k = [\varphi_{p-1, k+1} \bar{B}_k, \dots, \varphi_{1, k+p-1} \bar{B}_{k+p-2}, \bar{B}_{k+p-1}]$$

and

$$\bar{B}_k = [\tilde{B}, K_k]$$

Then, Eq.(3) can be transformed into:

$$\begin{aligned} x_{k+p} &= \varphi_{p, k} x_k + \bar{\kappa}_p^k \tilde{z}_k^p \\ x_{k+p} &= \varphi_{p, k} x_k + \kappa^p N_k^p \tilde{z}_k^p \end{aligned}$$

where

$$\varphi_{p, k} = \tilde{A}_{k+p-1} \dots \tilde{A}_{k+1} \tilde{A}_k$$

If the system (3) is uniformly exponentially stable the approximation error can be made arbitrarily small then:

$$x_{k+p} \approx \kappa^p N_k^p \tilde{z}_k^p$$

To calculate the observability matrix Γ^p times the state X , we first calculate the matrix $\Gamma^p \kappa^p$:

$$\Gamma^p \kappa^p = \begin{bmatrix} C l_p & C l_{p-1} & \cdot & \cdot & C l_1 \\ 0 & C \tilde{A}^{(1)} l_{p-1} & \cdot & \cdot & C \tilde{A}^{(1)} l_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & C (\tilde{A}^{(1)})^{p-1} l_1 \end{bmatrix}$$

Then, using the following Singular Value Decomposition (SVD):

$$\widehat{\Gamma^p \kappa^p Z} = [v \quad v_{\sigma \perp}] \begin{bmatrix} \sum^n & 0 \\ 0 & \sum \end{bmatrix} \begin{bmatrix} V \\ V_{\perp} \end{bmatrix}$$

the state is estimated by:

$$\widehat{X} = \sum_n V$$

Finally, C and D matrix are estimated using output equation (2) and A and B are estimated using the state equation (1). This algorithm can be summarized as follows [9]:

- Create the matrices U , Y and Z using (4),(5) and (6),
- Solve the linear problems given in (7) ,
- Construct Γ^p times the state X ,
- Estimate the state sequence,
- With the estimated state, use the linear relations to obtain the system matrices.

In the case of a very small p , we have in general a biased estimate. However, when the bias is too large, it will be a problem. That is why a large p would be chosen. In the case of a very large p , this method suffers from the curse of dimensionality [13] and the number of rows of Z grows exponentially with the size of the past window. In fact, the number of rows is given by:

$$\rho_Z = (r + \ell) \sum_{j=1}^p m^j$$

To overcome this drawback, the kernel method will be introduced in the next subsection [15].

2.3 Kernel method

The equation (7) has a unique solution if the matrix $[Z^T \ U^T]$ has full row rank and is given by:

$$\begin{bmatrix} \widehat{C\kappa^p} & \widehat{D} \end{bmatrix} = Y [Z^T \ U^T] \left(\begin{bmatrix} Z \\ U \end{bmatrix} [Z^T \ U^T] \right)^{-1}$$

When this is not the case, that will occurs when p is large, the solution is computed by using the SVD of the matrix:

$$\begin{bmatrix} Z \\ U \end{bmatrix} = [v \ v_{\perp}] \begin{bmatrix} \sum_m^m & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V^T \\ V_{\perp}^T \end{bmatrix}$$

Then, the solution of the minimum norm is given by:

$$\begin{bmatrix} \widehat{C\kappa^p} & \widehat{D} \end{bmatrix} = YV \sum_m^{-1} v^T$$

To avoid computations in a large dimensional space, the minimum norm results in:

$$\min_{\alpha} \|\alpha\|_F^2 \quad (8)$$

with

$$Y - \alpha [Z^T Z + U^T U] = 0$$

where α are the Lagrange multipliers and $[Z^T Z + U^T U]$ is referred as the kernel matrix.

The matrix Γ times the state X can be constructed as follows:

$$\Gamma \kappa^p Z = \begin{bmatrix} \alpha \sum_{j=1}^p (Z^{1,j})^T Z^{1,j} \\ \alpha \sum_{j=2}^p (Z^{2,j})^T Z^{1,j} \\ \vdots \\ \alpha \sum_{j=p}^p (Z^{p,j})^T Z^{1,j} \end{bmatrix} \quad (9)$$

with

$$(Z^{i,j})^T Z^{1,j} = \left(\prod_{v=0}^{p-j} \mu_{\bar{N}+v+j-i}^T \mu_{\bar{N}+v+j-1} \right) (z_{\bar{N}+j-i}^T z_{\bar{N}+j-1})$$

$$Z^T Z = \sum_{j=1}^p (Z^{1,j}) Z^{1,j} \quad (10)$$

Finally, the estimate sequence is obtained by solving the original SVD problem.

The kernel method can be summarized as follows [9]:

- Create the matrices $U^T U$ using (4) and $Z^T Z$ and $(Z^{i,j})^T (Z^{i,j})$ using (10),
- Solve the linear problem given in (8),
- Construct Γ times the state X using (9) and (10),
- Estimate the state sequence,
- With the estimated state, use the linear relation to obtain the system matrices.

3 Interval predictor approach

To add robustness to the LPV subspace identification approach presented in the previous section, it will be combined with the interval predictor approach [16]. The interval predictor approach is an extension of classical system identification methods in order to provide the nominal model plus the uncertainty bounds for parameters guaranteeing that all collected data from the system in non-faulty scenarios will be included in the model prediction interval. This approach considers separately the additive and multiplicative uncertainties. Additive uncertainty is taken into account in the additive error term $e(k)$ and modeling uncertainty is considered to be located in the parameters that are represented by a nominal value plus some uncertainty set around. In the literature, there are many approximation of the set uncertain parameter Θ . In our case, this set is described by a zonotope [10]:

$$\Theta = \theta^0 \oplus HB^n = \{\theta^0 + Hz : z \in B^n\} \quad (11)$$

where: θ^0 is the nominal model (here obtained with the identification approach, H is matrix uncertainty shape, B^n is a unitary box composed of n unitary ($B = [-1, 1]$) interval vectors and \oplus denotes the Minkowski sum. A particular case of the parameter set is used that corresponds to the case where the parameter set Θ is bounded by an interval box [17]:

$$\Theta = [\underline{\theta}_1, \overline{\theta}_1] \times \dots [\underline{\theta}_i, \overline{\theta}_i] \times \dots [\underline{\theta}_{n_{\theta}}, \overline{\theta}_{n_{\theta}}] \quad (12)$$

where $\underline{\theta}_i = \theta_i^0 - \lambda_i$ and $\overline{\theta}_i = \theta_i^0 + \lambda_i$ with $\lambda_i \geq 0$ and $i = 1, \dots, n_{\theta}$. In particular, the interval box can be viewed as a zonotope with center θ^0 and H equal to an $n_{\theta} \times n_{\theta}$ diagonal matrix:

$$\theta^0 = \left(\frac{\overline{\theta}_1 + \underline{\theta}_1}{2}, \frac{\overline{\theta}_2 + \underline{\theta}_2}{2}, \dots, \frac{\overline{\theta}_{n_{\theta}} + \underline{\theta}_{n_{\theta}}}{2} \right) \quad (13)$$

$$H = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{n_{\theta}}) \quad (14)$$

For every output, a model can be extracted in the following regressor form:

$$y(k) = \varphi(k)\theta(k) + e(k) \quad (15)$$

where

- $\varphi(k)$ is the regressor vector of dimension $1 \times n_\theta$ which can contain any function of inputs $u(k)$ and outputs $y(k)$.
- $\theta(k) \in \Theta$ is the parameter vector of dimension $n_\theta \times 1$.
- Θ is the set that bounds parameter values.
- $e(k)$ is the additive error bounded by a constant where $|e(k)| \leq \sigma$.

In the interval predictor approach, the set of uncertain parameters Θ should be obtained such that all measured data in fault-free scenario will be covered by the interval predicted output.

$$y(k) \in [\hat{y}(k) - \sigma, \overline{\hat{y}(k)} + \sigma] \quad (16)$$

where

$$\hat{y}(k) = \hat{y}^0(k) - \|\varphi(k)H\|_1 \quad (17)$$

$$\overline{\hat{y}(k)} = \hat{y}^0(k) + \|\varphi(k)H\|_1 \quad (18)$$

and $\hat{y}^0(k)$ is the model output prediction with nominal parameters with $\theta^0 = [\theta_1, \theta_2, \dots, \theta_{n_\theta}]^T$ obtained using the LPV identification algorithm:

$$\hat{y}^0(k) = \varphi(k)\theta^0(k) \quad (19)$$

Then, fault detection will be based on checking if (16) is satisfied. In case that, it is not satisfied a fault can be indicated. Otherwise, nothing can be said.

4 Case study: wind turbine benchmark system

In this work, a specific variable speed turbine is considered. It is a three blade horizontal axis turbine with a full converter. The energy conversion from wind energy to mechanical energy can be controlled by changing the aerodynamics of the turbine by pitching the blades or by controlling the rotational speed of the turbine relative to the wind speed. The mechanical energy is converted to electrical energy by a generator fully coupled to a converter. Between the rotor and the generator, a drive train is used to increase the rotational speed from the rotor to the generator [18]. This model can be decomposed into submodels: Aerodynamic, Pitch, Drive train and Generator [19] [20]. In this paper, we focus on faults in the pitch subsystem as explained in the following subsection.

4.1 Pitch system model

In the wind turbine benchmark model, the hydraulic pitch is a piston servo mechanism which can be modeled by a second order transfer function [21] [1]:

$$\frac{\beta(s)}{\beta_r(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (20)$$

Notice that β_r refers to reference values of pitch angles. The pitch model can be written in the following state space:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -2\zeta\omega_n x_2 - \omega_n^2 x_1 + \omega_n^2 u \end{cases} \quad (21)$$

with

$$x_1 = \beta, x_2 = \dot{\beta}, u = \beta_r$$

which can be discretised using an Euler approximation. Then, the following system is obtained:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases} \quad (22)$$

with

$$A = \begin{bmatrix} 1 & T_e \\ -T_e\omega_n^2 & -2T_e\zeta\omega_n + 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ T_e\omega_n^2 \end{bmatrix}$$

$$C = [1 \ 0]$$

4.2 LPV Pitch system model

The pitch parameters w_n and ξ are variable with hydraulic pressure P [1] [22]. Then, the pitch model can be written as the following LPV model according to [23] using P as the scheduling variable ϑ :

$$\begin{cases} x(k+1) = A(\vartheta)x(k) + B(\vartheta)u(k) \\ y(k) = Cx(k) \end{cases} \quad (23)$$

with

$$A(\vartheta) = \begin{bmatrix} 1 & T_e \\ -T_e\omega_n^2(P) & -2T_e\xi(P)\omega_n(P) + 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ T_e\omega_n^2(P) \end{bmatrix}$$

$$y(k) = x_1(k) = \beta(k)$$

4.3 Regressor form pitch system model

The pitch model can be transformed to the following regression form [24]:

$$y(k) = \varphi(k)\theta(k) \quad (24)$$

where, $\varphi(k)$ is the regressor vector which can contain any function of inputs $u(k)$ and outputs $y(k)$. $\theta(k) \in \Theta$ is the parameter vector. Θ is the set that bounds parameter values.

In particular

$$\varphi(k) = [y(k-2) \ y(k-1) \ u(k-2)]$$

$$\theta = [\theta_1 \ \theta_2 \ \theta_3]^T$$

$$\theta_1 = (-T_e^2\omega_n^2 + (2w_n\xi T_e - 1))$$

$$\theta_2 = -2w_n\xi T_e + 2$$

$$\theta_3 = T_e^2\omega_n^2$$

5 Results

The pitch systems, which in this case are hydraulic, could be affected by faults in any of the three blades. The considered faults in the hydraulic system can result in changed dynamics due to a drop in the main line pressure. This dynamic change induces a change in the system parameters: the damping ratio between 0.6 rad/s and 0.9 rad/s and the frequency between 3.42 rad/s and 11.11 rad/s according to [23]. In this work, a fault detection subspace estimator is designed to determine the presence of a fault. To distinguish between fault and modeling errors, an interval predictor approach is applied and a residual generation is used for

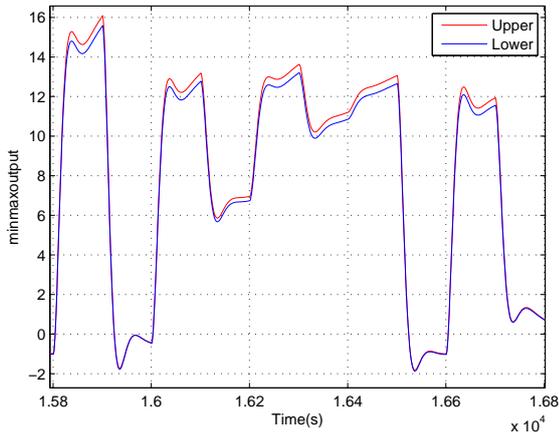


Figure 2: Upper (red line) and lower (blue line) bounds

deciding if there is a fault. To illustrate the performance of this robust fault detection approach: $\xi \in [0.6 \ 0.63]$ and $w_n \in [10.34 \ 11.11]$ are considered. Then, a parameter set Θ is bounded by an interval box:

$$\Theta = [\underline{\theta}_1, \overline{\theta}_1] \times [\underline{\theta}_2, \overline{\theta}_2] \times [\underline{\theta}_3, \overline{\theta}_3] \quad (25)$$

and for $i = 1, \dots, 3$

$$\lambda_i = \left(\frac{\overline{\theta}_i - \underline{\theta}_i}{2} \right) \quad (26)$$

$$\theta_i^0 = \left(\frac{\overline{\theta}_i + \underline{\theta}_i}{2} \right) \quad (27)$$

using equations (17) and (18), the output bounds are calculated to be used in fault detection test which are given in Fig. 2. $\hat{y}^0(k)$ is obtained by the use of the identification approach described in Section 2. To validate this algorithm two cases are used:

- Case 1: In this case, the pressure varies after time 10000s while parameters vary in the interval of parametric uncertainties, that is, damping ratio varies between 0.6 rad/s and 0.63 rad/s and the frequency between 10.34 rad/s and 11.11 rad/s. These parameters are presented respectively in Figures. 3 and 4. The pitch angle in this case is given in Fig. 5 altogether with the prediction intervals. For fault detection, the residual signal, based on the comparison between the measured pitch angle and the estimated one at each sampling instance, is calculated and it is shown in Fig. 6. For fault decision, a fault indicator signal is used and the decision is taken in function of this indicator. If the actual angle is not within the predicted interval given in Eq.(16), the fault indicator is equal to 1 and the system is faulty. Otherwise, it is equal to 0 and the system is fault-free. The fault indicator signal given in Fig. 7 shows that there is no fault despite the pressure variation. The parameters variation is considered as a modeling error.

- Case 2: In this case, the pressure P varies between time $t = 10000s$ and $t = 17000s$ outside its nominal value. In this time interval, the damping ratio varies between 0.63 rad/s and 0.72 rad/s and the frequency varies between

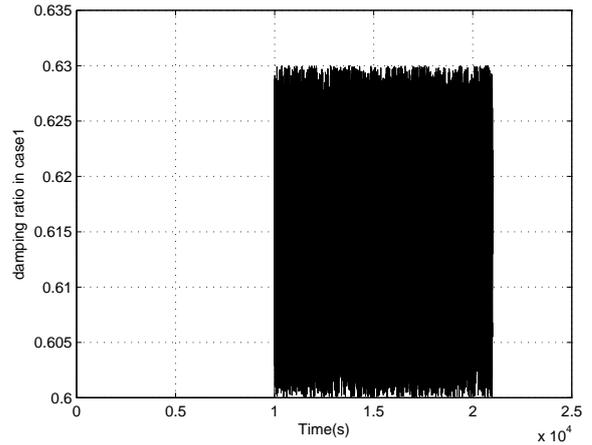


Figure 3: Damping ratio in non-faulty case

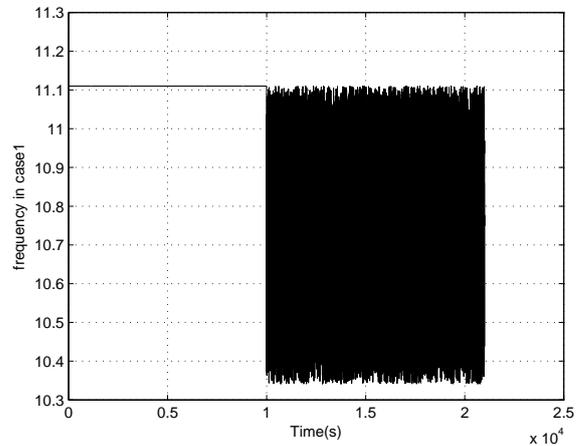


Figure 4: Frequency in non-faulty case

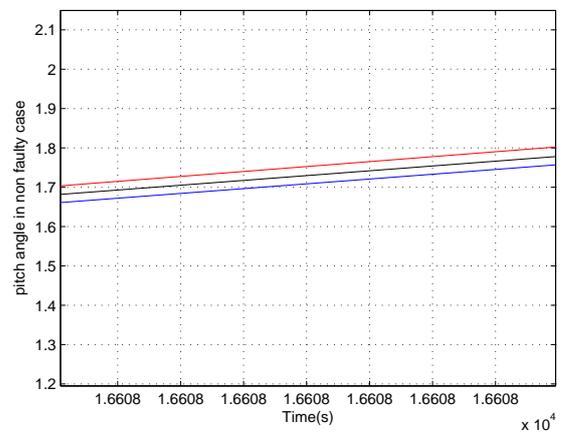


Figure 5: Pitch angle in non-faulty case

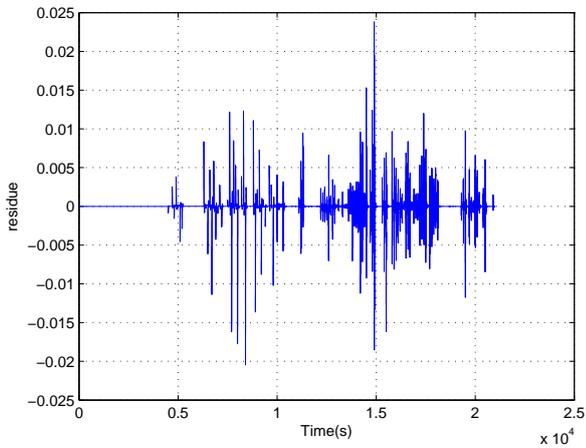


Figure 6: Residual in non-faulty case

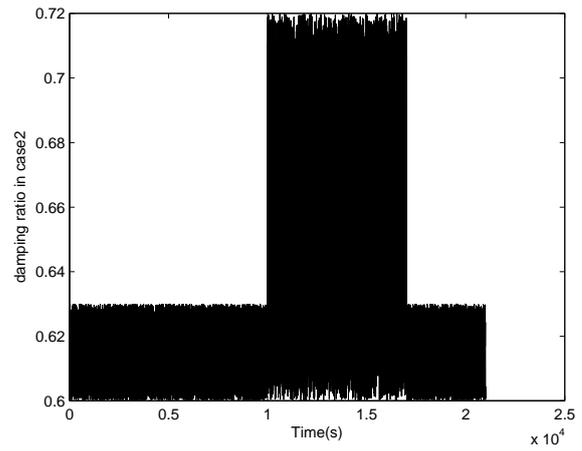


Figure 8: Damping ratio in faulty case

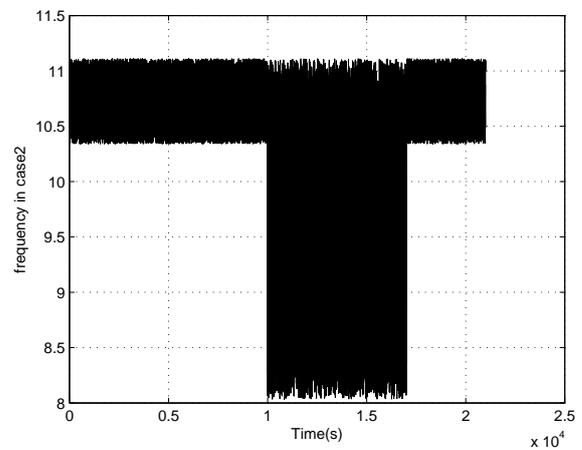


Figure 9: Frequency in faulty case

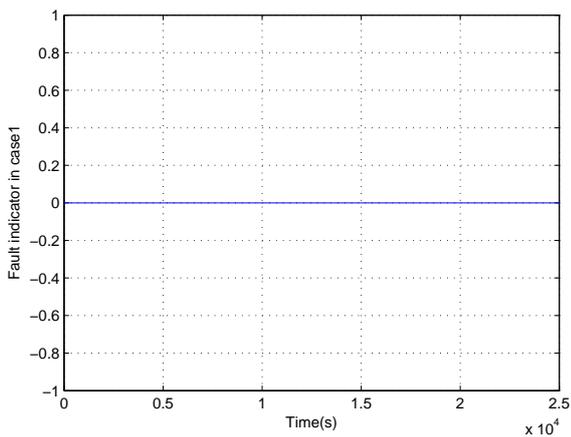


Figure 7: Fault indicator in non-faulty case

8.03 *rad/s* and 10.34 *rad/s*. On the other hand, the damping ratio varies between 0.6 *rad/s* and 0.63 *rad/s* and the natural frequency varies between 10.34 *rad/s* and 11.11 *rad/s* outside as shown in Figures 8 and 9. In this case, the pitch angle is given in Fig. 10, while the residual and fault indicator signals are presented in Fig. 11 and Fig. 12, respectively.

Fig. 12 shows that the fault indicator signal changes its signature between time 10000s and 17000s which induce that the parameters vary larger than the modeling range due to actuator fault in wind turbine benchmark system between instants $t = 10000s$ and $17000s$.

6 Conclusions

The proposed approach is based on an LPV estimation approach to generate a residual as the difference between the real and the nominal behavior of the monitored system. When a fault occurs, this residual goes out of the interval which represents the uncertainty bounds in non faulty case. These bounds are generated by means of an interval predictor approach that adds robustness to this fault detection method, by means of propagating the parameter uncertainty to the residual or predicted output. The proposed

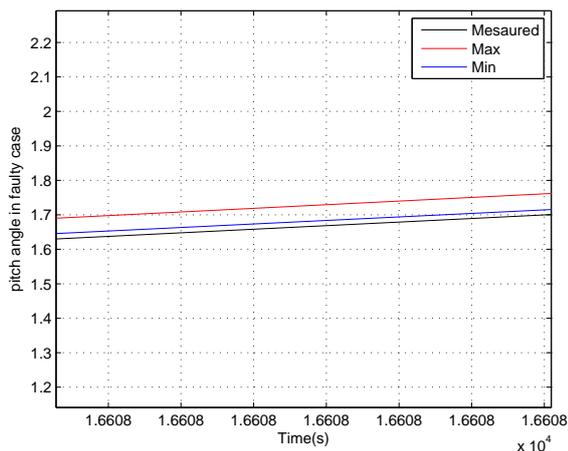


Figure 10: Pitch angle in faulty case

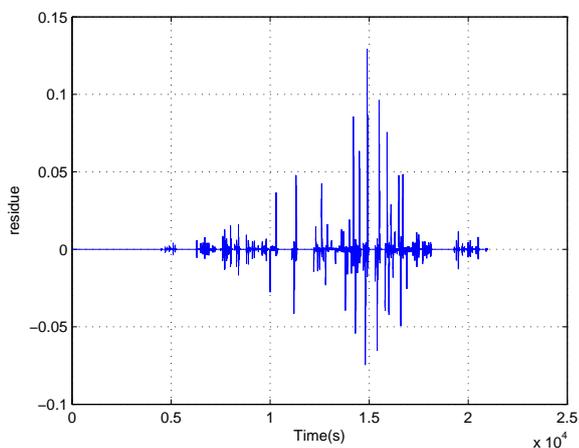


Figure 11: Residual signal

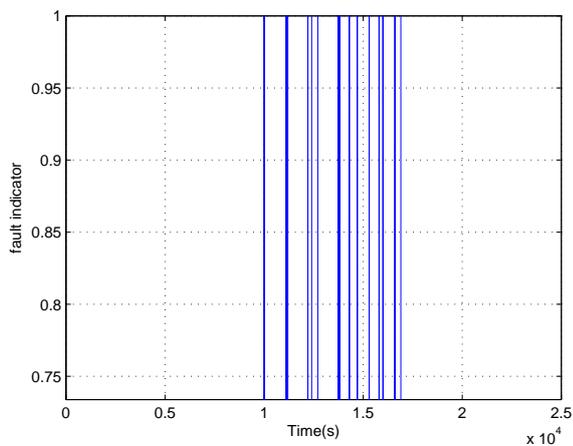


Figure 12: Fault indicator

approach is illustrated by implementing a robust fault detection scheme for a pitch subsystem of the wind turbine benchmark. Simulations show satisfactory fault detection performance despite model uncertainties.

References

- [1] P. Odgaard, J. Stoustrup, and M. Kinnaert. Fault tolerant control of wind turbines-a benchmark model. In *7th IFAC symposium on fault detection, supervision and safety of technical processes, Barcelona, Spain, 2009*.
- [2] R. Isermann. *Fault diagnosis systems: an introduction from fault detection to fault tolerance*. 2006.
- [3] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J Schröder. *Diagnosis and fault-tolerant control*. 2006.
- [4] G. Mercere, M. Lovera, and E Laroche. Identification of a flexible robot manipulator using a linear parameter-varying descriptor state-space structure. In *Proc. of the IEEE conference on decision and control, Orlando, Florida, USA, 2011*.
- [5] J. Dong, B. Kulcsár, and M Verhaegen. Fault detection and estimation based on closed-loop subspace identification for linear parameter varying systems. In *DX, Stockholm, 2009*.
- [6] J. Bravo, T. Alamo, and E.F. Camacho. Bounded error identification of systems with time-varying parameters. *IEEE Transactions on Automatic Control*, 51:1144–1150., 2006.
- [7] D. Efimov, L. Fridman, T. Raissi, A. Zolghadri, and R. Seydou. Interval estimation for lpv systems applying high order sliding mode techniques. *Automatica*, 48:2365–2371, 2012.
- [8] D. Efimov, T. Raissi, and A. Zolghadri. Control of nonlinear and lpv systems: interval observer-based framework. *IEEE Transactions on Automatic Control.*, 2013.
- [9] J. Van Willem and M Verhagen. Subspace identification of bilinear and lpv systems for open-and closed-loop data. *Automatica*, 45:371–381, 2009.
- [10] J. Blesa, V. Puig, J Romera, and J Saludes. Fault diagnosis of wind turbines using a set-membership approach. In *the 18th IFAC world congress, Milano, Italy, 2011*.
- [11] H. Tanaka, Y Ohta, and Y Okimura. A local approach to lpv-identification of a twin rotor mimo system. In *in proceedings of the 47th IEEE Conference on Decision and Control Cancun, Mexico, 2008*.
- [12] R. Toth, F. Felici, P. Heuberger, and P Van den Hof. Discrete time lpv i/o and state-space representations, differences of behavior and pitfalls of interpolation. In *in proceedings of the European Control Conference (ECC), Kos, Greece, 2007*.
- [13] J. Van Willem and M Verhagen. Subspace identification of mimo lpv systems: the pbsid approach. In *in Proceedings of the 47th IEEE Conference on Decision and Control Cancun, Mexico, 2008*.

- [14] P. Gebraad, J. Van Wingerden, G. Van der Veen, and M Verhaegen. Lpv subspace identification using a novel nuclear norm regularization method. In *American Control Conference on O'Farrell Street, San Francisco, CA, USA*, 2011.
- [15] V. Verdult and M Verhaegen. Kernel methods for subspace identification of multivariable lpv and bilinear systems. *Automatica*, 41:1557–1565, 2005.
- [16] J. Blesa, V. Puig, and J Saludes. Identification for passive robust fault detection using zonotope based set membership approaches. *International journal of adaptive control and signal processing*, 25:788–812, 2011.
- [17] P. Puig, V. Quevedo, T. Escobet, F. Nejjari, and S De las Heras. Passive robust fault detection of dynamic processes using interval models. *IEEE Transactions on Control Systems Technology*, 16:1083 –1089, 2008.
- [18] B. Boussaid, C. Aubrun, and M.N Abdelkrim. Set-point reconfiguration approach for the ftc of wind turbines. In *the 18th World Congress of the International Federation of Automatic Control (IFAC), Milano, Italy*, 2011.
- [19] B. Boussaid, C. Aubrun, and M.N Abdelkrim. Two-level active fault tolerant control approach. In *The Eighth International Multi-Conference on Systems, Signals Devices (SSD'11), Sousse, Tunisia*, 2011.
- [20] B. Boussaid, C. Aubrun, and M.N Abdelkrim. Active fault tolerant approach for wind turbines. In *The International Conference on Communications, Computing and Control Applications (CCCA'11), Hammamet, Tunisia*, 2011.
- [21] P. Odgaard, J. Stoustrup, and M Kinnaert. Fault tolerant control of wind turbines-a benchmark model. *IEEE Transactions on control systems Technology*, 21:1168–1182, 2013.
- [22] P. Odgaard and J Stoustrup. Results of a wind turbine fdi competition. In *8th IFAC symposium on fault detection, supervision and safety of technical processes, Mexico*, 2012.
- [23] C. Sloth, T. Esbensen, and J Stoustrup. Robust and fault tolerant linear parameter varying control of wind turbines. *Mechatronics*, 21:645–659, 2011.
- [24] H. Chouiref, B. Boussaid, M.N Abdelkrim, V. Puig, and C Aubrun. Lpv model-based fault detection: Application to wind turbine benchmark. In *International conference on electrical sciences and technologies (cistem'14), Tunis*, 2014.

Processing measure uncertainty into fuzzy classifier

Thomas Monrousseau¹, Louise Travé-Massuyès¹ and Marie-Véronique Le Lann^{1,2}

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

²Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

e-mails: thomas.monrousseau@laas.fr, louise@laas.fr ,mvlelann@laas.fr

Abstract

Machine learning such as data based classification is a diagnosis solution useful to monitor complex systems when designing a model is a long and expensive process. When used for process monitoring the processed data are available thanks to sensors. But in many situations it is hard to get an exact measure from these sensors. Indeed measure is done with a lot of noise that can be caused by the environment, a bad use of the sensor or even the conversion from analogic to numerical measure. In this paper we propose a framework based on a fuzzy logic classifier to model the uncertainty on the data by the use of crisp (non fuzzy) or fuzzy intervals. Our objective is to increase the number of good classification results in the presence of noisy data. The classifier is named LAMDA (Learning Algorithm for Multivariate Data Analysis) and can perform machine learning and clustering on different kind of data like numerical values, symbols or interval values.

1 Introduction

Data classification is the process of dividing pattern space using hard, fuzzy or probabilistic partitions into a number of regions [1]. Classification algorithms are more and more used nowadays in a world where it is not always simple to get a model of complex process. On the opposite it is easier to get data on systems by monitoring and store it. Different types of classifiers can be used depending on the situation. The principal ones described in the literature are artificial neural networks, k-nearest neighbors, support vector machine, decision trees, fuzzy classifiers and statistical methods.

Most of the time, data are issued from sensor measurements and are corrupted by noise. This noise can have different origins, for example environment disturbances, bad use of the sensor, hysteresis effect or numerical conversion and representation of the data. Many domains of application have to deal with noise problems like medical diagnosis [2], biologic identifications [3] or image recognition [4]. Uncertainty can be understood in two ways: the first is the uncertainty directly present in the data like noise and the second can be assimilated as the reliability of a feature inside a class. In this paper we consider only the first case. To avoid noise problems in classification some solutions have been provided previously, for example the transformation of

data [5] [6] [7], the use of fuzzy logic type-1 or type-2 [3] or statistical models.

Fuzzy logic is a multi-valued logic framework introduced by Zadeh [8] that is known to be more efficient for representing uncertainty and impreciseness than binary logic. In previous work, a fuzzy classifier named Learning Algorithm for Multivariate Data Analysis (LAMDA) has been proposed by Aguilar [9]. This classifier can originally process simultaneously two different types of data: quantitative data and qualitative data. A real number contains an infinite amount of precision whereas human knowledge is finite and discrete, thus LAMDA is interesting because there is no solution proposed in the literature to process in a uniform way heterogeneous data and to handle in a same problem quantitative data and qualitative data is often a complex subject. A new type of data, the interval, has been introduced by Hedjazi [10] to model uncertainties by means of crisp intervals. In this paper we propose an extension to fuzzy intervals in order to improve its application to process noisy data measurements but with the capacity to handle others features types like “clean” data or qualitative features. Moreover the algorithm should stay low cost in term of memory and computation time to enable the method to be embedded on small systems.

In the first part of the paper the LAMDA algorithm is shortly presented then in a second time a method to use the algorithm to classify noisy data is introduced. This method is in two parts: the first presents a general solution to model uncertainty on data with crisp intervals based on confidence intervals and the second shows an improvement to model Gaussian noise with fuzzy intervals. In both cases examples of application are introduced to show the improvement of the method compared to the use of the data without transformation.

2 LAMDA algorithm (Learning Algorithm for Multivariate Data Analysis)

This section presents the principle of the LAMDA algorithm.

2.1 General principle

LAMDA is a classification algorithm based on fuzzy logic created on an original idea of Aguilar [9] and can achieve machine learning and clustering on large data sets.

The algorithm takes as input a sample x made up of N features. The first step is to compute for each feature of x , an

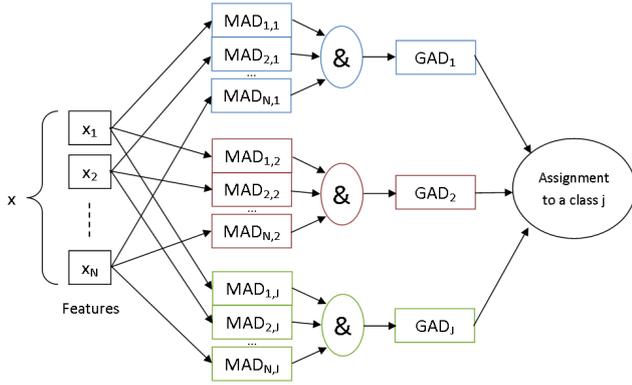


Figure 1: Summarized scheme of the LAMDA algorithm

adequacy degree to each class C_j , $j = 1..J$ where J is the total number of class. This is obtained by the use of a fuzzy adequacy function. So J vectors of N adequacy degrees are computed, these vectors are called Marginal Adequacy Degree vectors (MAD). At this point, all the features are in a common space. Then the second step is to take all the MADs and aggregate them into one global adequacy degree (GAD) by means of a fuzzy aggregation function. Thus the J MAD vectors (composed of N MADs) become J scalar GADs, the higher the GAD, the better the adequacy to the class. The simplest way to assign the sample x to a class is to keep as result the class with the biggest GAD.

All the process is summarized in Fig. 1.

2.2 Fuzzy membership computation

During the learning step, the algorithm creates prototype data for each class and for each feature. These data are called classe descriptors or prototypes; they can be for example means or variances. We define as $C_{j,n}$ the class prototype of the n -th feature for the class j .

As previously mentioned the first step of the algorithm is a comparison between the sample vector x and all the $C_{j,n}$. This operation is performed with membership functions and gives as result a membership adequacy degree. Thus $MAD_{j,n}$ is the MAD for the j -th class and the n -th feature. As the framework is based on fuzzy logic, all memberships are numbers in the $[0,1]$ interval. The general membership function is:

$$MAD_{j,n} = f(C_{j,n}, x_n) \quad (1)$$

The class prototype $C_{j,n}$ depends on two things: the type of data and the function used. Some functions may require only one data into $C_{j,n}$ whereas others need a list of parameters.

In the following section, some examples of membership functions are presented.

- Quantitative data:

Many functions are available for this kind of data. For example the Gaussian:

$$f(x_n) = e^{-\frac{(x_n - \rho_{j,n})^2}{2\sigma_{j,n}^2}} \quad (2)$$

or the binomial function:

$$f(x_n) = \rho_{j,n}^{x_n} \cdot (1 - \rho_{j,n})^{1-x_n} \quad (3)$$

Where x_n is the n -th feature of the sample x , $\rho_{j,n}$ is the mean of the n -th feature for the class j and $\sigma_{j,n}$ is the standard deviation of the n -th feature for the class j .

- Qualitative data:

Qualitative can take values in a set of modalities. The membership function of qualitative data returns the frequency of modality taken by the feature into the class during the learning phase. We introduce a qualitative variable with K modality $\{Q^1, \dots, Q^K\}$ and the frequency Φ_j^k of the modality Q^k for the class j . The membership is described by:

$$f(x_n) = (\Phi_{j,n}^1)^{q_1} * \dots * (\Phi_{j,n}^K)^{q_K} \quad (4)$$

$$\text{with } \begin{cases} q^k = 0 & \text{if } x_n \neq Q_k \\ q^k = 1 & \text{if } x_n = Q_k \end{cases}$$

- Intervals:

The membership function for interval data is a function which tests the similarity between two fuzzy intervals. In this case similarity is defined by two components: the distance between the intervals and the surface that these intervals have in common. Indeed the class prototype for crisp interval data is a mean interval. The similarity function is:

$$S(A, B) = \frac{1}{2} \left(\frac{\int_V \mu_{A \cap B}(\xi) d\xi}{\int_V \mu_{A \cup B}(\xi) d\xi} + 1 - \frac{\partial[A, B]}{\varpi[V]} \right) \quad (5)$$

where $\mu_X(x)$ is the value of x in the fuzzy set X , $\partial[A, B]$ is the distance between intervals $A = [a^-, a^+]$ and $B = [b^-, b^+]$ and $\varpi[X]$ is the size of a fuzzy set into a V universe. This is described by:

$$\varpi[X] = \int_V \mu_X(\xi) d\xi \quad (6)$$

In the case of crisp intervals and in a universe between 0 and 1:

$$S(A, B) = \frac{1}{2} \left(\frac{\varpi[A \cap B]}{\varpi[A \cup B]} + 1 - \partial[A, B] \right) \quad (7)$$

where $\varpi[X]$ in this case can be replaced by the length of the interval:

$$\varpi[X] = \text{upperbound}(X) - \text{lowerbound}(X) \quad (8)$$

and distance $\partial[A, B]$ is defined as:

$$\partial[A, B] = \max[0, \max(a^-, b^-) - \min(a^+, b^+)] \quad (9)$$

In the case where an interval feature is used the prototype for a class j is given by $[\rho_j^{n-}, \rho_j^{n+}]$ where ρ_j^{n-} , respectively ρ_j^{n+} represents the mean value of lower bounds (respectively upper bounds) of all the elements belonging to class j for this feature.

Once the MAD are computed whatever the feature type, it is possible to perform any type of processing as described on Fig. 2

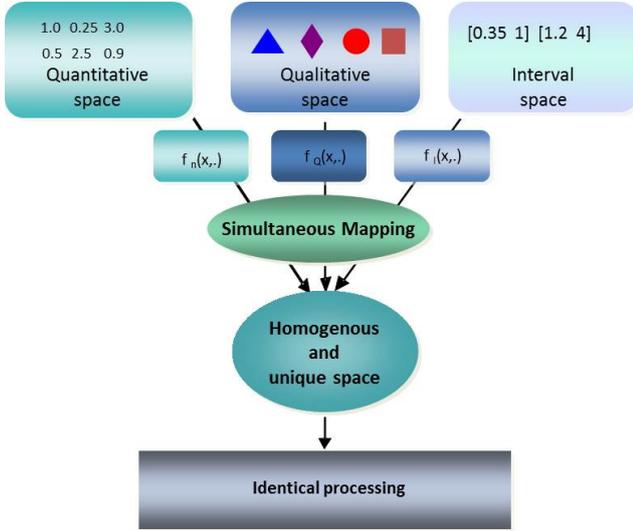


Figure 2: Projection principle for heterogeneous feature types

2.3 Marginal adequacy degree merging

Once all the features are grouped into the membership space the next step of the algorithm is to transform the MAD vectors into a set of single value which depicts the global membership of the sample to a class. These values were introduced in section 2.1 and are called GAD. To perform this transformation a fuzzy aggregation function Ψ is used.

The aggregation function is the following:

$$\Psi(MAD) = \alpha \cdot \gamma(MAD) + (1 - \alpha) \cdot \beta(MAD) \quad (10)$$

where γ is a fuzzy T-norm and β is a fuzzy T-conorm. α parameter is called exigency indicator. It enables to give more or less significance to the union operation and the intersection operation. Two fuzzy T-norm and T-conorm are currently implemented in the algorithm, the min-max and the probabilistic. For example if min-max is used, (10) becomes:

$$\Psi(MAD) = \alpha \cdot \min(MAD) + (1 - \alpha) \cdot \max(MAD) \quad (11)$$

When all GAD are computed they give the membership of the data x to each class. The final result depends on the application but the simplest way to give a result is to class the sample in the class which has the highest GAD. A limit membership can also be fixed: if no GAD is higher than the limit, the sample is defined as unclassifiable.

3 Uncertainty modeled with crisp intervals

3.1 Method presentation

Every data measurement is performed with noise. In some cases noise has enough bad effect to increase the error of classification. Thus the point is to model the imprecision of the data to decrease the number of bad classifications.

A technique used in several fields of application is the use of intervals to symbolize data uncertainty [11] [12]. So we are suggesting a framework where numerical data are transformed into intervals to model imprecision.

In a situation where the probability law followed by the noise on a variable is unknown, it may be possible to obtain a confidence interval. It is an interval in which the real value of the measure is present with a certain amount

of confidence (for example a confidence interval of 95% is an interval in which the exact value of the measure can be found with a probability of 95%). Introducing \hat{x} the measured value and l the length of a centered on zero confidence interval based on the measurement error, the interval used by the algorithm is calculated: $X = [\hat{x} - \frac{l}{2}; \hat{x} + \frac{l}{2}]$.

The main aim of the transformation is to improve the classification on the transition zones where data is really sensitive to noise and a small change can modify the output of the classifier. The use of intervals to model uncertainty is effective only if the “clean” data is relevant for the classification problem. If it is not the case a better solution is to remove the irrelevant feature. It will in most cases provide better output results. This expresses the fact that if the “clean” data is difficult to classify it is not improved by using confidence intervals.

3.2 Experiments

A set of data has been created for an application test which can be interpreted as sensors time evolution of a continuous process. This set of data is composed by three quantitative (numerical) features of 101 samples that are shown on the Fig. 3. Three classes are specified and used as targets for the classifier. These classes are chosen arbitrarily to represent different behaviors of a system that could be healthy or failure modes. Nevertheless the classes are built to make all the data relevant for the system monitoring which means the three features do not have a global negative impact on the classification results.

The three features x , y and z are defined by the following time functions:

- $x = e^{-\frac{t}{2}}$
- $y = \frac{1}{2} \cdot e^{\frac{t}{4}} - 1$
- $z = \tanh(t - 5)$

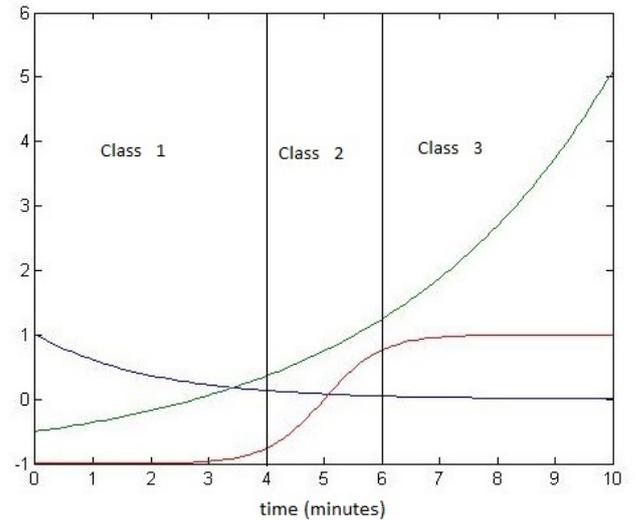


Figure 3: Data used to test the intervals method

This example is used to measure the improvement in the classification results in the case of all data are noisy. Artificial noise is added by the following: x is the ideal variable without noise and \hat{x} the noisy variable, $\hat{x} = x + Y$ with

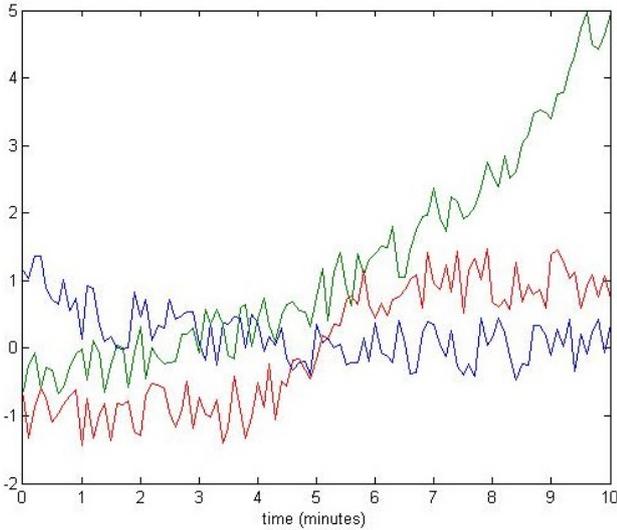


Figure 4: An example of data corrupted with a noise in the interval $[-0.5 ; 0.5]$

Y a random variable following a uniform distribution on an interval I .

The experiment has been performed with these conditions: α parameter of (10) is set at 0.8 with the $[\min, \max]$ functions to compute the fuzzy aggregation and the membership function used for quantitative data is the binomial. $[\min, \max]$ aggregation is chosen because experiments on the algorithm showed that this kind of aggregation provides better results on noisy data than the probabilistic one. A first classification without any noise gives a result of 91% of good classification. Then the experiment is repeated a great many times to avoid statistical mistakes. In this case, the experiment has been run fifty thousand times, \hat{x} is re-computed at each new run. Results are given on table 1.

Interval for random data	$[-0.3 ; 0.3]$	$[-0.5 ; 0.5]$	$[-2 ; 2]$
Mean success percentage with binomial function	89.9%	84.7%	79.6%
Mean success percentage with interval function	91.9%	89.8%	70.3%

Table 1: Table of results for the crisp intervals method

As it can be seen, this method provides an improvement on the results in the two first cases where noise deteriorates the classification with the quantitative method but when the data is still globally consistent. In these cases, the intervals method gives better results than binomial method 82% of the time. But when noise amplitude is much higher than the data like in the $[-2 ; +2]$ error interval, the interval method does worse in general than the binomial function.

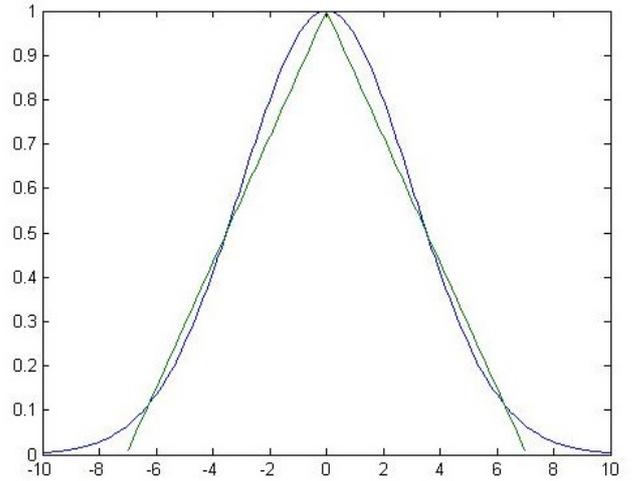


Figure 5: Example of approximation of a Gaussian fuzzy interval by a triangular fuzzy interval

4 Modeling Gaussian noise with fuzzy intervals

4.1 Fuzzy interval method presentation

Most of the time, noise on physical measure follows a Gaussian distribution centered on the real value. Thus it is interesting to model this specific kind of uncertainty. Nevertheless, it is difficult to handle fuzzy intervals with an exact Gaussian shape. That is why we suggest approximating the Gaussian with a triangular fuzzy interval. This interval is described with a lower boundary x^- and an upper boundary x^+ : $X = [x^- ; x^+]$ which leads to a similar description as crisp intervals. So:

$$\mu_X(x^-) = 0 \text{ and } \mu_X(x^+) = 0 \text{ and } \mu_X\left(\frac{x^+ + x^-}{2}\right) = 1$$

with $\mu_X(x)$ the fuzzy value of x into the fuzzy set X . As a Gaussian of ρ mean is centered on the true measure value the maximum fuzzy value of the triangle $\frac{x^+ + x^-}{2}$ is equal to ρ . To compute x^- and x^+ we propose to use the full width at half maximum (FWHM) that can be calculated this way:

$$FWHM = 2\sqrt{2\ln(2)} \cdot \sigma \quad (12)$$

with σ that is the standard deviation of the measure. Thus for a Gaussian function that has a mean value ρ and a standard deviation σ the approximated interval X is defined by $X = [\rho - 2\sqrt{2\ln(2)} \cdot \sigma ; \rho + 2\sqrt{2\ln(2)} \cdot \sigma]$. An example of this approximation is given on Fig. 5.

Until now all the implementations of the LAMDA algorithm were using only crisp intervals despite the fact that the general method was introduced. The class prototype is now a triangle interval computed with the means of upper and lower boundaries of the data used to train the algorithm. Thus the membership function is still a similarity measure between two fuzzy intervals like in (5) but it is necessary to redefine the distance function between the intervals. A solution has been proposed to measure a distance with the center of gravity of triangular fuzzy intervals [13]. In the present situation:

$$\partial[A, B] = \left| \frac{a_+ + a_-}{2} - \frac{b_+ + b_-}{2} \right| \quad (13)$$

with $A = [a_-; a_+]$ and $B = [b_-; b_+]$, A and B being triangular fuzzy intervals like described in this section.

The intersection $A \cap B$ needed in (5) is calculated with an analytical solution based on geometry and trigonometry. It avoids numerical integration that could be less precise and longer to compute.

4.2 Experiments

As we did previously with the crisp method, a test is performed with a Gaussian noise on the same data set (Fig. 3). The test is done in the same conditions as in the previous section. The difference is on the construction of the noisy data $\hat{x} = x + Y$. Y is now a random variable that follows a normal distribution of standard deviation σ and centered on 0. Results of the simulation are given on the table 2.

σ	0.2	0.5	0.7	1
Mean success percentage with binomial function	83.2%	79.8%	79.8%	79.6%
Mean success percentage with crisp interval function	86.8%	82.5%	77.2%	71.3%
Mean success percentage with fuzzy interval function	93.1%	84.5%	79.3%	74.8%

Table 2: Table of results for the fuzzy intervals method

Similarly to the previous test, the interval method increases the rate of good classifications until the standard deviation σ becomes too high and the binomial function provides better results. This point is reached here for $\sigma = 0.7$ which corresponds to a signal to noise ratio (SNR) of 6 dB for the signal with the smallest amplitude. Also it is important to notify that in all cases the fuzzy interval provides better results than the crisp interval method.

4.3 Experiments on iris dataset

As a second example we use the classical iris dataset[14]. This dataset contains four features: sepal length in cm, sepal width in cm, petals length in cm and petal width in cm. All these features are measured for three types of flower: iris Setosa, iris Versicolour and iris Virginica which constitute three classes. It is easy to classify without any error the iris dataset by using only the petals information that are in general most relevant than the sepals ones. Thus only the sepal sizes are kept in this test to simulate the noise. The figure 6 shows the repartition of the data in the 2D space of the sepal features.

We assume that the data follow a normal distribution centered on a mean $\mu_{j,n}$ and with a standard-deviation $\sigma_{j,n}$. This hypothesis can be verified by using a statistical test. The Kolmogorov-Smirnov test has been used for each class with a 5% significance level, it shows that the hypothesis is true for the iris Setosa and the iris Versicolour but not for the iris Virginica. Nevertheless all the data are processed as if they follow a normal distribution.

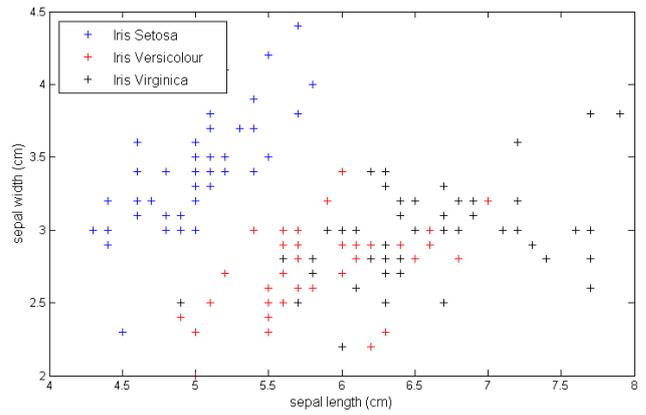


Figure 6: Representation of iris data by class

The classifications are performed using the cross-validation method. The percentages of well classified data for the two methods are:

- using binomial function (scalar): 81.3%
- using fuzzy triangular intervals: 94.0%

Once again the classification rate is increased by the use of the fuzzy interval method instead of the binomial one.

5 Conclusion

We presented in this article two methods to model uncertainty for classification applications. An example showed that these methods can improve classification results even when the signal to noise ratio is high. The second method based on fuzzy intervals demonstrated that try to model more precisely the probability law of the noise can provide better results than use confidence intervals modelled by crisp intervals. However this process to model uncertainty reveals limits when the SNR reaches a low level. A future important work is to limit the classification error of the interval method at the level of the numerical method.

These methods will now be tested on data out coming from a real industrial process.

Another way to manage uncertainty on classifiers like LAMDA could be to use type-2 fuzzy functions [15]. This is an expansion of classical fuzzy logic where the membership functions give in output a fuzzy interval which can be used to model variance of the data.

To provide a better solution to manage uncertainty in the LAMDA classifier it can be useful to extend the problem to the qualitative features. It is often difficult to determine if a qualitative element is close to another, for example the color "orange" is closer to "red" than "blue". But on small training dataset consider this kind of information can improve final classification results. This could be done by using similarity matrix which are already used in some artificial intelligence problems.

LAMDA algorithm can work with a feature selection algorithm named MEMBAS (Membership Margin Based Feature Selection) [16]. This algorithm uses LAMDA classes definitions and its membership functions to provide an analytical solution for the feature selection. A future work will be to measure the impact of the interval use on MEMBAS algorithm to perform selection on noisy data.

References

- [1] J. C. Bezdek. A review of probabilistic, fuzzy, and neural models for pattern recognition. *Journal of Intelligent and Fuzzy Systems*, Vol. 1, No. 1:pp 1–25, 1993.
- [2] E. Alba, J. Garcia-Nieto, L. Jourdan, and E. Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *Evolutionary Computation, CEC 2007. IEEE Congress on*, pages 284–290, Sept. 2007.
- [3] Scott Ferson, H. Resit Akqakaya, and Amy Dunham. Using fuzzy intervals to represent measurement error and scientific uncertainty in endangered species classification. In *Fuzzy Information Processing Society, 1999. NAFIPS. 18th International Conference of the North American on*, pages pp 690–694, Jul 1999.
- [4] Zhang Weiyu, S.X. Yu, and Shang-Hua Teng. Power svm: Generalization with exemplar classification uncertainty. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages pp 2144–2151, June 2012.
- [5] Arafat Samer, Dohrmann Mary, and Skubic Marjorie. Classification of coronary artery disease stress eegs using uncertainty modeling. In *Computational Intelligence Methods and Applications, 2005 ICSC Congress, 2005*.
- [6] Kynan E. Graves and Romesh Nagarajah. Uncertainty estimation using fuzzy measures for multiclass classification. *Neural Networks, IEEE Transactions on*, Vol. 18:pp. 128–140, 2007.
- [7] Prabha Verma and R.D.S. Yadava. Fuzzy c-means clustering based uncertainty measure for sample weighting boosts pattern classification efficiency. In *Computational Intelligence and Signal Processing (CISP), 2012 2nd National Conference on*, pages 31–35, 2012.
- [8] L.A. Zadeh. Fuzzy sets. *Information and Control*, vol. 8:pp. 338–353, June 1965.
- [9] Carrete N.P. and Aguilar-Martin J. Controlling selectivity in nonstandard pattern recognition algorithms. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 21, pages 71–82. IEEE, Jan/Feb 1991.
- [10] Hedjazi L., Aguilar-Martin J., Le Lann M.V., and Kempowsky T. Towards a unfinied principle for reasoning about heterogeneous data: a fuzzy logic framework. *International Journal of Uncertainty, Fuzzyness and Knowledge-Based Systems*, Vol. 20, No. 2:pp. 281–302, 2012.
- [11] B. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. The MIT Press,Cambridge, Massachusetts, london edition, 1994.
- [12] Lynne Billard. Some analyses of interval data. *Journal of Computing and Information Technology*, CIT 16:pp 225–233, 2008.
- [13] Hsieh C. H. and Chen S. H. Similarity of generalized fuzzy numbers with graded mean integration representation. In *Proceedings of the Eighth International Fuzzy Systems Association World Congress*, volume vol. 2, pages pp. 551–555, Taipei, Taiwan, Republic of China, 1999.
- [14] Fisher R.A. {UCI} machine learning repository, 1936. <http://archive.ics.uci.edu/ml>.
- [15] J.M. Mendel, R.I. John, and F. Liu. Interval type-2 fuzzy logic systems made simple. *Fuzzy Systems, IEEE Trans. on*, Vol. 14, No. 6:pp 808–821, Dec. 2006.
- [16] L.Hedjazi, J.Aguilar-Martin, and M.V. Le Lann. Similarity-margin based feature selection for symbolic interval data. *Pattern Recognition Letters*, Vol.32, No4:pp. 578–585, March 2012.

Tools/Benchmarks

Random generator of k -diagnosable discrete event systems

Yannick Pencolé^{1 2}

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

²Univ de Toulouse, LAAS, F-31400 Toulouse, France

e-mail: yannick.pencole@laas.fr

Abstract

This paper presents a random generator of discrete event systems that are by construction k -diagnosable. The aim of this generator is to provide an almost infinite set of diagnosable systems for creating benchmarks. The goal of such benchmarks is to provide a solid set of examples to test and compare algorithms that solve many problems around diagnosable discrete event systems.

1 Introduction

For many years, the problem of fault diagnosis in discrete event systems has been actively addressed by different scientific communities such as DX (AI-based diagnosis) [1; 2], FDI (Fault Detection and Isolation), DES (Discrete Event Systems) [3]. Depending on the community, many different aspects of the same problem have been addressed such as the design of efficient diagnosers, the checking of diagnosability properties, the effective modelling of real systems. When dealing with performance, most of the contributions present experimental results on specific examples of their own usually inspired or based on real world systems. The main problem about these contributions is that they are not really comparable as they are not applied on the same benchmarks. Moreover, used benchmarks may not be always completely defined in a paper due, most of the time, to confidential data that cannot be published so other academic contributors cannot use them for comparison purposes. In order to analyse and boost the effective performance of algorithms addressing the fault diagnosis problem in DES, common and fully available benchmarks become a necessity.

This paper addresses the random generation of k -diagnosable systems. We here propose the possibility to generate (and store on a web page) k -diagnosable systems that have been generated without any kind of bias that would come from a specific diagnosis/diagnosability method. By doing so, we propose to design a random category for benchmarks as the SAT community proposed for SAT problems and to get the same advantages by comparing different diagnosis/diagnosability approaches on the same but random systems. The choice of generating k -diagnosable systems is motivated by the fact that they can be used as examples for:

1. diagnosis algorithms: given a fault f , we know by construction that the most precise algorithm will determine its occurrence with certainty within the next k observations after the occurrence of f ;

2. diagnosability algorithms: the fact that a fault f is k -diagnosable is usually the worst case for this type of algorithms (as they all look for the existence of an ambiguous scenario to conclude the system is not diagnosable).

The paper is organised as follows. After formally recalling the problem that motivates the generation of benchmarks, we describe the fundamental property which is being used for the effective generation of systems where a given fault f is k -diagnosable. Then the description of the algorithm of the generator is provided as well as some details about its effective implementation.

2 Background

This paper addresses the random generation of benchmarks for the problem of the fault diagnosis of discrete event system. This problem is briefly recalled in this section. We assume that the reader is familiar with the notations of the language theory (notion of Kleene closure, prefixes,...).

2.1 Modelling

We suppose that the system under monitoring behaves as an event generator that can be modelled as an automaton.

Definition 1 (System description). *The model (system description) SD of a discrete event system S is a finite state automaton $SD = (Q, \Sigma, T, q_0)$ where:*

- Q is a finite set of states;
- Σ is a finite set of events;
- $T \subseteq Q \times T \times Q$ is a finite set of transitions;
- q_0 is the initial state of the system.

Σ is the set of events that the system can produce. Among Σ we distinguish events that are observable $\Sigma_o \subseteq \Sigma$ and events that are not observable. When the system operates, its effective behaviour is represented by a trace of the automaton (also called a run).

Definition 2 (Trace). *A trace $\tau \in \Sigma^*$ of the system is a finite sequence of events associated with a transition path from the initial state q_0 to a state q in the model of the system.*

The set of traces of the system is the language generated by its model and is denoted $\mathcal{L}(S)$ (so the automaton SD generates the language $\mathcal{L}(S)$). Let $P_{\Sigma'}(\tau)$ be the classical projection of a sequence τ of Σ^* on the alphabet Σ' recursively defined as follows:

1. $P_{\Sigma'}(\varepsilon) = \varepsilon$;

2. $P_{\Sigma'}(\tau.e) = P_{\Sigma'}(\tau)$ if $e \notin \Sigma'$;
3. $P_{\Sigma'}(\tau.e) = P_{\Sigma'}(\tau).e$ if $e \in \Sigma'$.

Based on this notion of projection, we can associate with any trace of the system its observable part.

Definition 3 (Observable trace). *Let τ be a trace of the system, the observable trace σ_τ is the projection of τ over the set of observable events Σ_o :*

$$\sigma_\tau = P_{\Sigma_o}(\tau).$$

2.2 Diagnosis problem and solution

Now we are ready to define the classical Fault diagnosis problem on DES.

Definition 4 (Fault). *A fault is a non-observable event $f \in \Sigma$.*

A fault is represented as a special type of non-observable event that can occur on the underlying system. Once the event has occurred, we say that the fault is *active* in the system, otherwise it is *inactive*. We consider here the problem of permanent faults as initially introduced in [4].

Definition 5 (Diagnosis problem). *A diagnosis problem is a triple (SD, OBS, FAULTS) where SD is the model of a system, OBS is the sequence of observations of Σ_o^* and FAULTS is the set of fault events defined over SD.*

Informally speaking, (SD, OBS, FAULTS) represents the problem of finding the set of active faults from FAULTS that have occurred relying on the model SD and the sequence of observations OBS.

Definition 6 (Diagnosis Candidate). *A diagnosis candidate is a couple (q, F) where q is a state of SD ($q \in Q$) and F is a set of faults.*

A diagnosis candidate represents the fact that the underlying system is in state q and the set F of faults has occurred before reaching state q .

Definition 7 (Solution Diagnosis). *The solution Δ of the problem (SD, OBS, FAULTS) is the set of diagnosis candidates (q, F) such that there exists for each of them at least one trace τ of SD such that:*

1. *the observable trace of τ is exactly the sequence $OBS = o_1 \dots o_m$ and the last event of τ is o_m ;*
2. *the set of fault events that has occurred in τ is exactly F ;*
3. *the final state of τ is q .*

Informally, candidate (q, F) is part of the solution if it is possible to find out in SD a behaviour of the system satisfying OBS which leads to the state q after the last observation of OBS and in which the faults F have occurred.

2.3 Diagnosability

Diagnosability is a property of the system that asserts whether a fault f of a system S can be always diagnosed with certainty after the observation of a finite set of observations [4]. In other words, once the fault f has occurred in S , it is sufficient to wait a certain amount of observations to ensure that any candidate (q, F) of the solution contains f ($f \in F$).

Definition 8 (Diagnosability). *The fault f is diagnosable in a system S if:*

$$\exists n \in \mathbb{N}^+, \text{Diagnosable}(n)$$

where $\text{Diagnosable}(n)$ stands for:

$$\begin{aligned} \forall \tau_1.f \in \mathcal{L}(S), \forall \tau_2 : \tau_1.f.\tau_2 \in \mathcal{L}(S) \\ |P_{\Sigma_o}(\tau_2)| \geq n \Rightarrow \\ (\forall \tau \in \mathcal{L}(S), (P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f.\tau_2) \Rightarrow f \in \tau)). \end{aligned}$$

Definition 9 (k -Diagnosability). *The fault f is k -diagnosable, $k \in \mathbb{N}^+$, in a system S if:*

$$\text{Diagnosable}(k) \wedge \neg \text{Diagnosable}(k-1).$$

Diagnosability is a property that relies on the liveness of the observability of the system which means that, to be (k)-diagnosable, a system must not generate unbounded sequences of unobservable events (no cycle of unobservable events in SD). Throughout this paper, we consider that the observability of the system is live.

3 Random Generator

The aim of this section is to present the algorithm that is being used to randomly generate a discrete event systems where a fault f is k -diagnosable and that has been implemented inside the Diades software. We focus on the generation of a system with one fault only. (see the conclusion for the generation for $n, n > 1$ faults).

3.1 Signatures and fault ambiguity

The algorithm that generates a k -diagnosable system relies on the notion of signatures. Let f be a faulty event, the signature of f is the set of observable traces resulting from the projection of system traces that contain at least one occurrence of an event f before the last observation of the trace.

Definition 10 (Signature). *The signature of an event f into a system S is the language $\text{Sig}(f) \subseteq \Sigma_o^*$ such that*

$$\begin{aligned} \text{Sig}(f) = \{ \sigma_\tau | \tau = \tau_1.o.\tau_2 \in \mathcal{L}(S), \\ f \in \tau_1, o \in \Sigma_o, \tau_2 \in \Sigma^*, \sigma_\tau = P_{\Sigma_o}(\tau) \}. \end{aligned}$$

In the following, we will also denote by $\text{Sig}(\neg f)$ the set of observable traces associated with the traces of the system that do not contain any fault f before the last observation. Intuitively speaking, as long as the current observable trace is in $\text{Sig}(\neg f) \cap \text{Sig}(f)$, we know that the system may have produced a faulty trace or a non-faulty trace before the last observation. k -diagnosability ensures that the ambiguity can last at most for k observations. The principle of the generator relies on the following result that formalizes this intuition. Let $\text{LARGESTPREFIXES}(\tau, n) = \{ \tau'_i : \tau = \tau'_i \tau_i, |\tau_i| = i, i \in \{0, \dots, n-1\} \}$ be the set of the n largest prefixes of τ (τ being a prefix of itself).

Theorem 1. *In the system S , the event f is k -diagnosable if and only if:*

1. *For any observable trace σ in $\text{Sig}(\neg f) \cap \text{Sig}(f)$, there exists $n < k$ such that $\text{LARGESTPREFIXES}(\sigma, n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$ and $\text{LARGESTPREFIXES}(\sigma, n+1) \not\subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$.*
2. *There exists at least one observable trace σ in $\text{Sig}(\neg f) \cap \text{Sig}(f)$ such that $\text{LARGESTPREFIXES}(\sigma, k-1) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$ and an observable o such that $\sigma o \in \text{Sig}(f)$.*

Proof: (\Rightarrow) Let $\tau_1.f$ be a trace of the system S . As S is k -diagnosable, there exists $m \leq k$ such that $\forall \tau_2 : \tau_1.f.\tau_2 \in \mathcal{L}(S), |P_{\Sigma_o}(\tau_2)| \geq m \Rightarrow (\forall \tau \in \mathcal{L}(S), (P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f.\tau_2) \Rightarrow f \in \tau))$. Consider one of these trace $\tau_1.f.\tau_2$ such that τ_2 contains exactly m observations ($P_{\Sigma_o}(\tau_2) = o_1 \dots o_m$). k -diagnosability implies that there exists a minimal integer $n \in \{1, \dots, m-1\}$ such that $P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1} \Rightarrow f \in \tau$ as soon as $\tau \in \mathcal{L}(S)$ and $P_{\Sigma_o}(\tau) = P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1}$, therefore $P_{\Sigma_o}(\tau_1.f).o_1 \dots o_{n+1} \in \text{Sig}(f) \setminus \text{Sig}(\neg f)$ and $\forall i \in \{1, \dots, n\}, P_{\Sigma_o}(\tau_1.f).o_1 \dots o_i \in \text{Sig}(f) \cap \text{Sig}(\neg f)$. So $\text{LARGESTPREFIXES}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$. So for any $\tau_1.f$ there exists $n < k$ such that $\text{LARGESTPREFIXES}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$. Now, remark that for any observable sequence σ that belongs to $\text{Sig}(\neg f) \cap \text{Sig}(f)$, there must exist a trace $\tau_1.f.\tau_2$ of the system, with τ_2 containing at least one observable event, such that $\sigma = P_{\Sigma_o}(\tau_1.f.\tau_2)$, so there must exist $n < k$ such that $\sigma \in \text{LARGESTPREFIXES}(P_{\Sigma_o}(\tau_1.f).o_1 \dots o_n, n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$ so, for any σ that belongs to $\text{Sig}(\neg f) \cap \text{Sig}(f)$, there is no set $\text{LARGESTPREFIXES}(\sigma, n+1)$ that only contains ambiguous signatures.

Finally, as S is k -diagnosable, we know that there exists at least one trace $\tau.f.\tau_1.o_1$, such that τ is a trace of the system that does not contain f , τ_1 is a finite continuation of $\tau.f$ that is unobservable and o_1 is observable and there is a finite continuation $\tau_2.o_2\tau_3.o_3 \dots \tau_k.o_k$ with $P_{\Sigma_o}(\tau_i) = \varepsilon$ such that for any $i \in \{1, \dots, k-1\}$, $P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_i.o_i) \in \text{Sig}(\neg f) \cap \text{Sig}(f)$ $P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_k.o_k) \in \text{Sig}(f) \setminus \text{Sig}(\neg f)$ which implies the condition 2 with $\sigma = P_{\Sigma_o}(\tau.f.\tau_1.o_1 \dots \tau_{k-1}.o_{k-1})$.

(\Leftarrow) Suppose now that conditions 1 and 2 hold. Consider an observable trace σ that is ambiguous ($\sigma \in \text{Sig}(f) \cap \text{Sig}(\neg f)$). Condition 1 states that there exists $n < k$ such that $\text{LARGESTPREFIXES}(\sigma, n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$ and $\text{LARGESTPREFIXES}(\sigma, n+1) \not\subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$. Consider now any largest observable trace σ' such that $|\sigma'| - |\sigma| = m$ and $\sigma \in \text{LARGESTPREFIXES}(\sigma', m) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$, it follows that $\text{LARGESTPREFIXES}(\sigma', m+n) \subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$ and $\text{LARGESTPREFIXES}(\sigma', m+n+1) \not\subseteq \text{Sig}(\neg f) \cap \text{Sig}(f)$. As σ' is one of the largest observable trace holding this condition, any observable trace $\sigma'o, o \in \Sigma_o$, is either in $\text{Sig}(f)$ or in $\text{Sig}(\neg f)$ but not in both of them. Condition 1 states that $m+n < k$, so k observations at least are required to solve the ambiguity. Condition 2 states that there exists at least such an observable trace σ' with $m+n = k-1$ and an observation o so that $\sigma'o$ is definitively in $\text{Sig}(f)$ so f can be diagnosed with certainty in this case with exactly k observations. Hence the result. \square

3.2 Algorithm

The principle of the random generator is depicted in Algorithm 1. Given a parameter k and a fault event f , the algorithm randomly generates a system S where the event f is k -diagnosable by construction. We also provide another parameter deg which is the maximal number of output transitions that is allowed per state during the generation of the system. Parameter deg is important for the creation of benchmarks as the output degree has a strong influence on the diagnosis/diagnosability computations.

Algorithm 1 General algorithm for the random generation of k -diagnosable systems.

Input: $k \in \mathbb{N}, k \geq 1$

Input: f an event

Input: deg maximal output degree

$(\Sigma_o, \Sigma) \leftarrow \text{GENERATEEVENTS}()$

$S \leftarrow \emptyset$

$AmbSig(f) \leftarrow \text{GENERATEAMBSIGNATURE}(k, \Sigma_o, deg)$

/ AmbSig(f) = (Q, \Sigma_o, T, q_0, A) a deterministic automaton */*

$MF[q_0] \leftarrow \text{GENERATESTATES}()$

$MNF[q_0] \leftarrow \text{GENERATESTATES}()$

for all $q \in Q$ **in Breadth-First Order from** q_0 **do**

$(\Sigma_o^f, \Sigma_o^{\neg f}) \leftarrow \text{RANDOMSPLIT}(\Sigma_o, q)$

$S \leftarrow S \cup \text{GENFAULTEXTS}^*(MF[q], \Sigma_o^f, deg)$

$S \leftarrow S \cup \text{GENNOMEXTS}^*(MNF[q], \Sigma_o^{\neg f}, deg)$

for all $q \xrightarrow{o} q' \in T$ **do**

if $MF[q'] = \emptyset$ **then**

$MF[q'] \leftarrow \text{GENERATESTATES}()$

$MNF[q'] \leftarrow \text{GENERATESTATES}()$

end if

$S \leftarrow S \cup$

$\text{GENNOMEXTS}(MNF[q], MNF[q'], o, deg)$

if $q' \notin A$ **then**

$S \leftarrow S \cup \text{GENNOMEXTS}(MF[q], MF[q'], o, deg)$

else

if $q \in A$ **then**

$S \leftarrow S \cup \text{GENEXTS}(MF[q], MF[q'], o, deg)$

else

$S \leftarrow S \cup$

$\text{GENFAULTEXTS}(MF[q], MF[q'], o, deg)$

end if

end if

end for

end for

Output: S where f is k -diagnosable.

The generation is composed of two steps. The first one is the generation of the ambiguous signature with $\text{GENERATEAMBSIGNATURE}$. The result of this function is a deterministic automaton $AmbSig(f) = (Q, \Sigma_o, T, q_0, A)$ that actually generates the language $\text{Sig}(f) \cap \text{Sig}(\neg f)$ (any transition path from state q_0 to an accepting state of A represents a sequence of $\text{Sig}(f) \cap \text{Sig}(\neg f)$). The automaton $AmbSig(f)$ is generated with respect to the conditions 1 and 2 that are defined in Theorem 1 to ensure the k -diagnosability of the resulting system S . The second step of the generation is the effective generation of S based on the ambiguous signature $AmbSig(f)$. The idea is to map every state q of $AmbSig(f)$ with two sets of states in S denoted $MF[q]$ and $MNF[q]$. Given any path σ of $AmbSig(f)$ that leads to state q with, as a last observation, the event o , any state of $MF[q]$ (resp. $MNF[q]$) will be reached by at least one transition path τ of S starting from a state of $MF[q_0]$ (resp. $MNF[q_0]$) that ends with a transition labelled with o and the observable projection of τ is exactly σ . The difference between MF and MNF is that any underlying path of S leading to a state of $MF[q]$ (resp. $MNF[q]$) has an observable projection which is a prefix of $\text{Sig}(f)$ (resp. a prefix of $\text{Sig}(\neg f)$). To generate S we explore $AmbSig(f)$ from its initial state in a breadth-first search manner. For a given state q , we have to consider three types of transition

generations going out of any state of $MF[q]$, $MNF[q]$. The first ones are the transition paths that will lead to an observation o that belongs to $AmbSig(f)$, the second one is the set of transition paths that do not lead to an observation o that belongs to $AmbSig(f)$ but lead to an observation o' that belongs to $Sig(f)$ only and the third one is the set of transition paths that do not lead to an observation o that belongs to $AmbSig(f)$ but lead to an observation o'' that belongs to $Sig(\neg f)$ only.

The second and third cases are handled by randomly splitting Σ_o into two subsets ($\Sigma_o^f, \Sigma_o^{\neg f}$) each of them only containing observable events that are not output event of q in $AmbSig(f)$ (RANDOMSPPLIT(Σ_o, q)). Then given Σ_o^f , we randomly generate faulty extensions for a subset of Σ_o^f (the selection of the subset is also random and might even be empty if q has no output events in $AmbSig(f)$, indeed if q has no output events, it must be extended to ensure that the observability of the system is live). An extension is a set of acyclic and unobservable transition paths that lead to a transition labeled with an observable event from Σ_o^f . A faulty extension ensures that an event f has at least occurred on any generated transition path before the observable transition (GENFAULTEXTS). Given $\Sigma_o^{\neg f}$, we proceed the same way to generate non-faulty extensions (GENNOMEXTS). As, in these two cases, the traces generated by these extensions are not associated with observable traces involved in $AmbSig(f)$ any more, it is sufficient to generate further extensions on these traces and guarantee that the observable language associated with these further extensions is live (this procedure is denoted by the $*$ in GENNOMEXTS* and GENFAULTEXTS*).

The last case to handle now is the case where the observable event o is an output event of q in $AmbSig(f)$, which means that there exists one and only one transition $q \xrightarrow{o} q'$ in $AmbSig(f)$. If q' has never been visited, the set of states $MF[q']$ and $MNF[q']$ are generated first. A nominal extension is generated from $MNF[q]$ to $MNF[q']$. Depending on the status of q' , the extension between $MF[q]$ and $MF[q']$ is different. If $q' \notin A$, it means that any prefix generated by $AmbSig(f)$ with paths from q_0 to q' are prefixes of sequences in $Sig(f) \cap Sig(\neg f)$ but they are not in $Sig(f) \cap Sig(\neg f)$, they can therefore be only in $Sig(\neg f)$: extensions between $MF[q]$ and $MF[q']$ are then nominal extensions. Now, if $q' \in A$, there are two cases. If $q \notin A$, it means the system must become faulty between the states of $MF[q]$ and $MF[q']$ so that paths of the system that reach any state of $MF[q']$ is associated with an observable trace that belongs to $Sig(f)$ (GENFAULTEXTS). If $q \in A$, any path that reaches a state of $MF[q]$ is already faulty (its observable trace is already in $Sig(f)$), any type of extension from $MF[q]$ to $MF[q']$ is therefore possible (faulty or not), hence the use of GENEXTS.

3.3 Implementation

Algorithm 1 is implemented with the help of the Diades library package [5]. Diades is a set of C++ libraries that implement discrete event systems in a component-based way, different diagnosis algorithms as defined in the spectrum of [6] (from component-based algorithms to diagnoser-based algorithms). DIADES also implements a diagnosability checker as well as an accuracy checker. The generator results in a Linux terminal command `dd-diagnosable-des-generate` with a set of pa-

rameters like the number of (un)observable events the output degree of transitions, the parameter k , the minimal number of observable events involved in the ambiguous signature, the number of states (still experimental). One particular parameter is the seed parameter that allows the generation of the same system (the seed ensures the same generation of random numbers). By construction, the algorithm is linear in the number of states. A set of pre-computed benchmarks as well as the implemented generator are available at the following url:

<http://homepages.laas.fr/ypencole/benchmarks>

4 Conclusions

To test and compare diagnosis and/or diagnosability algorithms, fully detailed and available benchmarks are a necessity. In order to test how generic is an algorithm, we propose here an algorithm that randomly generates systems where a fault f is k -diagnosable. We also propose an implementation within the DIADES framework. Extension to generate systems with n k -diagnosable faults is easy, it requires to repeat the generation of the ambiguous signatures for the n faults and explore them in parallel to generate the k -diagnosable system. Our short-term perspective is to improve the generator to allow a better control of the number of generated states. A fixed number of generated states requires to add new constraints in the generation that propagate during the generation process. Without any control about this propagation, the generation may just fail as it could become an over-constrained problem. Our perspective is to also go one step further by generating diagnosable systems that are component-based in order to scale up the size of the generated system. The DIADES framework already has a tool to generate component-based systems [5] which ensures that any component is globally consistent, but adding the constraint of diagnosability makes the generation far more complex to implement.

References

- [1] Gianfranco Lamperti and Marina Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, 137:91–163, 2002.
- [2] Yannick Pencolé and Marie-Odile Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(2):121–170, 2005.
- [3] Janan Zaytoon and Stéphane Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37:308–320, 2013.
- [4] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *Transactions on Automatic Control*, 40(9):1555–1575, 9 1995.
- [5] Yannick Pencolé. Fault diagnosis in discrete-event systems: How to analyse algorithm performance? In *Diagnostic reasoning: Model Analysis and Performance*, pages 19–25, Montpellier, France, 2012.
- [6] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In *17th International Workshop on Principles of Diagnosis*, pages 194–201, Nashville, TN USA, 2007.

HYDIAG: extended diagnosis and prognosis for hybrid systems

Elodie Chanthery^{1,2}, Yannick Pencolé¹, Pauline Ribot^{1,3}, Louise Travé-Massuyès¹

¹CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

e-mail: [firstname.name]@laas.fr

²Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

³Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France

Abstract

HYDIAG is a software developed in Matlab by the DISCO team at LAAS-CNRS. It is currently a software designed to simulate, diagnose and prognose hybrid systems using model-based techniques. An extension to active diagnosis is also provided. This paper aims at presenting the native HYDIAG tool, and its different extensions to prognosis and active diagnosis. Some results on an academic example are given.

1 Introduction

HYDIAG is a software developed in Matlab, with Simulink. The development of this software was initiated in the DISCO team with contributions about diagnosis on hybrid systems [1]. It has undergone many changes and is currently a software designed to simulate, diagnose and prognose hybrid systems using model-based techniques [2; 3; 4]. An extension to active diagnosis has been also realized [5; 6]. This article aims at presenting the native HyDiag tool and its different extensions to prognosis and active diagnosis.

Section 2 recalls the hybrid formalism used by HYDIAG. Section 3 presents the native HYDIAG tool that simulates and diagnoses hybrid systems. Section 4 explains how HYDIAG has been extended in HYDIAGPRO to prognose and diagnose hybrid systems. Section 5 presents the extension to active diagnosis. Experimental results of HYDIAG and its extension HYDIAGPRO are finally presented in Section 6.

2 Hybrid Model for Diagnosis

HYDIAG deals with hybrid systems defined in a monolithic way. Such a system must be modeled by a hybrid automaton [7]. Formally, a hybrid automaton is defined as a tuple $S = (\zeta, Q, \Sigma, T, C, (q_0, \zeta_0))$ where:

- ζ is a finite set of continuous variables that comprises input variables $u(t) \in \mathbb{R}^{n_u}$, state variables $x(t) \in \mathbb{R}^{n_x}$, and output variables $y(t) \in \mathbb{R}^{n_y}$.
- Q is a finite set of discrete system states.
- Σ is a finite set of events.
- $T \subseteq Q \times \Sigma \rightarrow Q$ is the partial transition function between states.
- $C = \bigcup_{q \in Q} C_q$ is the set of system constraints linking continuous variables.

- $(\zeta_0, q_0) \in \zeta \times Q$, is the initial condition.

Each state $q \in Q$ represents a behavioural mode that is characterized by a set of constraints C_q that model the linear continuous dynamics (defined by their representations in the state space as a set of differential and algebraic equations). A behavioural mode can be nominal or faulty (anticipated faults). The unknown mode can be added to model all the non anticipated faulty situations. The discrete part of the hybrid automaton is given by $M = (Q, \Sigma, T, q_0)$, which is called the *underlying discrete event system (DES)*. Σ is the set of events that correspond to discrete control inputs, autonomous mode changes and fault occurrences. The occurrence of an anticipated fault is modelled by a discrete event $f_i \in \Sigma_f \subseteq \Sigma_{uo}$, where $\Sigma_{uo} \subseteq \Sigma$ is the set of unobservable events. $\Sigma_o \subseteq \Sigma$ is the set of observable events. Transitions of T model the instantaneous changes of behavioural modes. The continuous behaviour of the hybrid system is modelled by the so called *underlying multimode system* $\Xi = (\zeta, Q, C, \zeta_0)$. The set of directly measured variables is denoted by $\zeta_{OBS} \subseteq \zeta$.

An example of a hybrid system modeled by a hybrid automaton is shown in Figure 1. Each mode q_i is characterized by state matrices A_i, B_i, C_i and D_i .

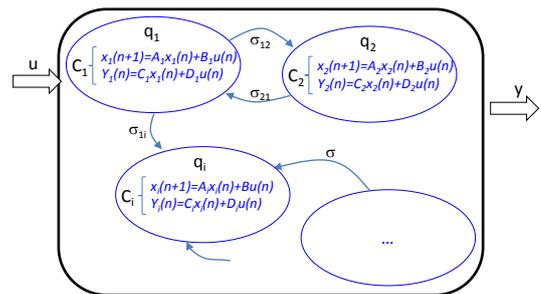


Figure 1: Example of an hybrid system

3 Overview of the native HYDIAG diagnoser

The method developed in [1] for diagnosing faults on-line in hybrid systems can be seen as interlinking a standard diagnosis method for continuous systems, namely the parity space method, and a standard diagnosis method for DES, namely the diagnoser method [8].

3.1 How to use HYDIAG ?

Step 1: hybrid model edition

HYDIAG allows the user to edit the modes of a hybrid automaton S as illustrated in Figure 1. To model the system, the user must first provide in the Graphical User Interface of the HYDIAG software the following information: the number of modes, the number of discrete events that can be observable or unobservable, and the sampling period used for the underlying multimode system (defined by the set of state matrices of the state space representation of each mode).

There are optional parameters that are helpful to initialize the mode matrices automatically before editing them: the number of entries for the continuous dynamics, the number of outputs for continuous dynamics, the dimensions of each matrix A . The number of entries (resp. outputs) must be the same for all the modes.

The simulator of the edited model has no restrictions on the number of modes or the order of the continuous dynamics, it is generically designed. Online computations are performed using Matlab / Simulink. Results provided by Matlab can be reused if a special need arises. Figure 2 shows an overview of the software interface.

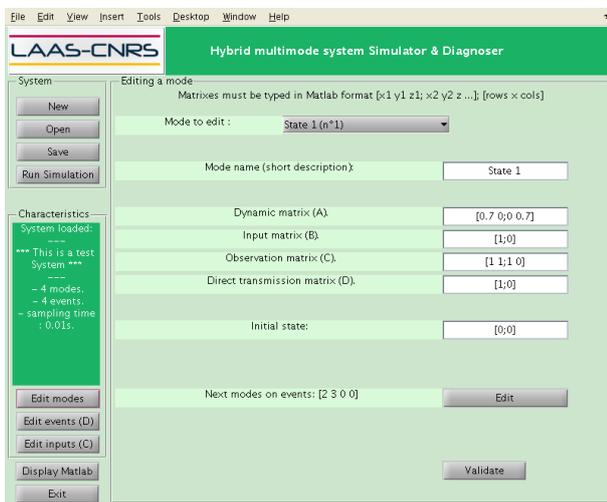


Figure 2: HYDIAG Graphical User Interface

Step 2: building the diagnoser

HYDIAG automatically computes the analytical redundancy relations (ARRs) by using the parity space approach [9]. Details of this computation can be found in [10].

The idea of HYDIAG is to capture both the continuous dynamics and the discrete dynamics within the same mathematical object. To do so, the discrete part of the hybrid system $M = (Q, \Sigma, T, q_0)$ is enriched with specific observable events that are generated from continuous information. The resulting automaton is called the *Behaviour Automaton* (BA) of the hybrid system. HYDIAG then builds the diagnoser of the Behaviour Automaton (see [8]) by using the DIADES¹ software also developed within the DISCO team at LAAS-CNRS (see an example of diagnoser in Figure 7).

Step 3: system simulation and diagnosis

Given the built hybrid diagnoser, HYDIAG then loads a set of timed observations produced by the system and it provides at each observation time an update of the diagnosis

¹<http://homepages.laas.fr/ypencole/DiaDes/>

of the system by triggering the current transition of the hybrid diagnoser that matches the current observation. It is possible to define in HYDIAG a simulation scenario for the modeled system with a duration and a time sample defined by the user.

3.2 Software architecture with extensions

The general architecture of HYDIAG and its two extensions (see the next sections for their description) is presented on Figure 3. Ellipses represent the objects handled by the software, rectangles with rounded edges depict HYDIAG functions and rectangles with straight edges correspond to external DIADES packages. The behaviour automaton is at the heart of the architecture as HYDIAG and both its extensions rely on it to perform diagnosis, active diagnosis and prognosis.

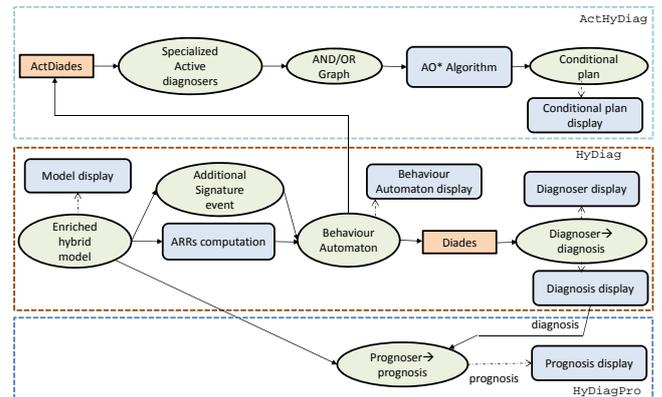


Figure 3: HYDIAG architecture with its extensions HYDIAGPRO and ACTHYDIAG.

4 HYDIAGPRO : an extension for Prognosis

HYDIAG has been extended in order to provide a prognosis functionality to the software [4]. The prognosis function computes (1) the fault probability of the system in each behavioural mode, (2) the future fault sequence that will lead to the system failure, (3) the Remaining Useful Life (RUL) of the system.

In HYDIAGPRO, the initial hybrid model is enriched by adding for each behavioural mode a set of aging laws: $S^+ = (\zeta, Q, \Sigma, T, C, \mathcal{F}, (q_0, \zeta_0))$ where $\mathcal{F} = \{F^q, q \in Q\}$ and F^q is a set of aging laws one for each anticipated fault $f \in \Sigma_f$ in mode q . The aging modeling framework that is adopted in HYDIAGPRO is based on the Weibull probabilistic model [11] (see more details in [4]). The Weibull fault probability density function $W(t, \beta_j^q, \eta_j^q, \gamma_j^q)$ gives at any time the probability that the fault f_j occurs in the system mode q . Weibull parameters β_j^q and η_j^q are fixed by the system mode q and characterise the degradation in mode q that leads to the fault f_j . Parameter γ_j^q is set at runtime to memorize the overall degradation evolution of the system accumulated in the past modes [11].

The prognoser uses the aging laws in S^+ to predict fault occurrences (see Figure 3). The prognoser uses the current diagnosis result to update on-line these aging laws (the parameters γ_j^q) according to the operation time in each behavioural mode. For each new result of diagnosis, the prognosis function computes the most likely sequence of dated

faults that leads to the system failure. From this sequence is estimated the system RUL [4].

5 ACTHYDIAG: Active Diagnosis

The second extension of HYDIAG provides an active diagnosis functionality to the software (see Figure 3). The inputs are the same as for HYDIAG but an additional file indicates the events of S that are actions, as well as their respective cost. Based on the behaviour automaton, we compute a set of specialised active diagnosers (one per fault): such a diagnoser is able to predict, based on the behaviour automaton, whether a fault can be diagnosed with certainty by applying an action plan from a given ambiguous situation [6]. From these diagnosers, we also extract a planning domain as a AND/OR graph.

At runtime, when HYDIAG is diagnosing, the diagnosis might be ambiguous. An active diagnosis session can be launched as soon as a specialised active diagnoser can analyse that the current faulty situation is discriminable by applying some actions. If the active diagnosis session is launched, an AO* algorithm starts and computes a conditional plan from the AND-OR graph that optimises an action cost criterion. It is important to note that in the case of a system with continuous dynamics, only discrete actions are contained in the active diagnosis plan issued by ACTHYDIAG. In particular, it is assumed that if it is necessary to guide the system towards a value on continuous variables, the synthesis of control laws must be performed elsewhere.

6 HyDiag/HyDiagPro Demonstration

Water tank system model

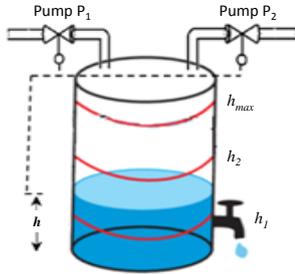


Figure 4: Water tank system

HYDIAGPRO has been tested on a water tank system (Figure 4) composed of one tank with two hydraulic pumps (P_1, P_2). Water flows through a valve at the bottom of the tank depending on the system control. Three sensors (h_1, h_2, h_{max}) detect the water level and allow to set the control of the pumps (on/off). It is assumed that the pumps may fail only if they are on. The discrete model of water tank and the controls of pumps are given in Figure 5. Discrete events in $\Sigma = \{h_1, h_{2s}, h_{2i}, h_{max}, f_1, f_2\}$ allow the system to switch into different modes. Observable events are $\Sigma_o = \{h_1, h_{2s}, h_{2i}, h_{max}\}$. Two faults that correspond to the pump failures are anticipated $\Sigma_f = \{f_1, f_2\}$ and are not observable. The Weibull parameter values of aging models $\mathcal{F} = \{F^{q_i}\}$ are reported in Table 1.

The underlying continuous behaviour of every discrete mode q_i for $i \in \{1..8\}$ is represented by the same state

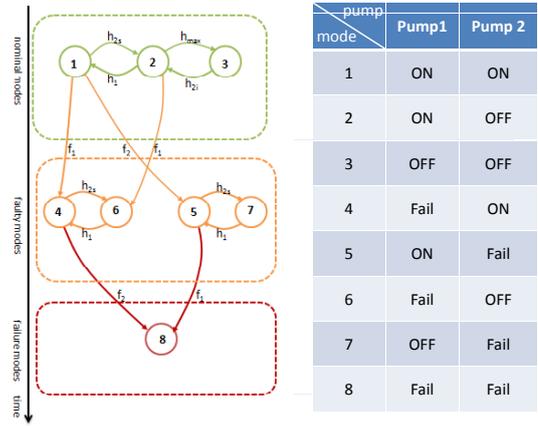


Figure 5: Water tank DES model

Table 1: Weibull parameters of aging models

Aging laws		β	η	Aging laws		β	η
F^{q1}	f_1^{q1}	1.5	3000	F^{q2}	f_1^{q2}	2	3000
	f_2^{q1}	1.5	4000		f_2^{q2}	1	7000
F^{q3}	f_1^{q3}	1	8000	F^{q4}	f_1^{q4}	NaN	NaN
	f_2^{q3}	1	7000		f_2^{q4}	2	4000
F^{q5}	f_1^{q5}	2	3000	F^{q6}	f_1^{q6}	NaN	NaN
	f_2^{q5}	NaN	NaN		f_2^{q6}	1	7000
F^{q7}	f_1^{q7}	1	8000	F^{q8}	f_1^{q8}	NaN	NaN
	f_2^{q7}	NaN	NaN		f_2^{q8}	NaN	NaN

space:

$$\begin{cases} X(k+1) = AX(k) + BU(k) \\ Y(k) = CX(k) + DU(k) \end{cases} \quad (1)$$

where the state variable X is the water level in the tank, continuous inputs U are the flows delivered by the pumps P_1, P_2 and the flow going through the valve, $A = (1)$, $B = \begin{pmatrix} eTe/S \\ eTe/S \\ eTe/S \end{pmatrix}$ with Te the sample time, S the tank base area and $e_i = 1$ (resp. 0) if the pump is turned on (resp. turned off), $C = (1)$ and $D = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

HYDIAG results

Figure 6 presents the set of results obtained by HYDIAG and HYDIAGPRO on the following scenario. The time horizon is fixed at $T_{sim} = 4000h$, the sampling period is $T_s = 36s$ and the filter sensitivity for the diagnosis is set as $T_{filter} = 3min$. The residual threshold is 10^{-12} . The scenario involves a variant use of water (max flow rate = 1200L/h) depending on user needs during 4000h. Pumps are automatically controlled to satisfy the specifications indicated above. Flow rate of P_1 and P_2 are respectively 750L/h and 500L/h.

The diagnoser computed by HYDIAG is given in Figure 7. Each state of the diagnoser indicates the belief state in the model enriched by the abstraction of the continuous part of the system, labelled with faults that have occurred on the system. This label is empty in case of nominal mode. In the scenario, fault f_1 was injected after 3500h and fault f_2 was not injected.

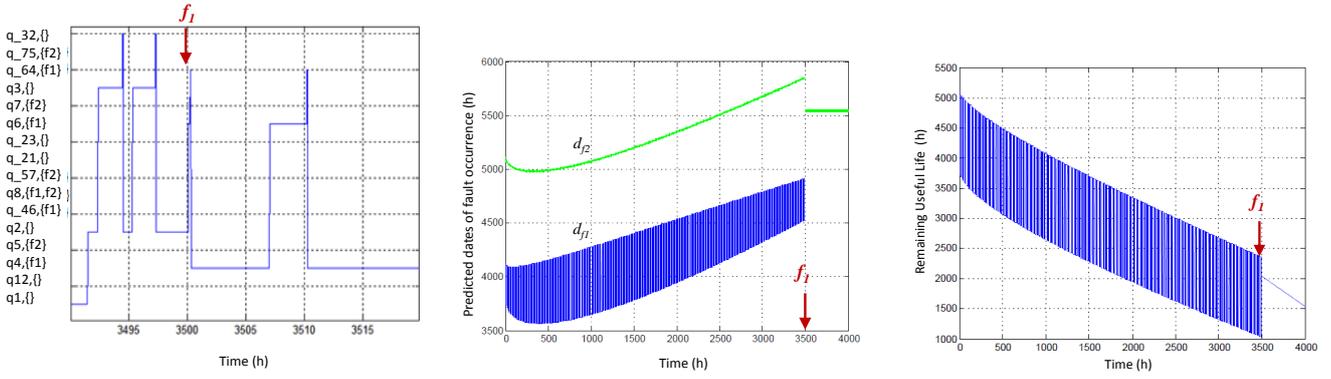


Figure 6: Scenario: Diagnoser belief state (left), Prognosis results of degradations d_{f_1} and d_{f_2} (middle), System RUL (right).

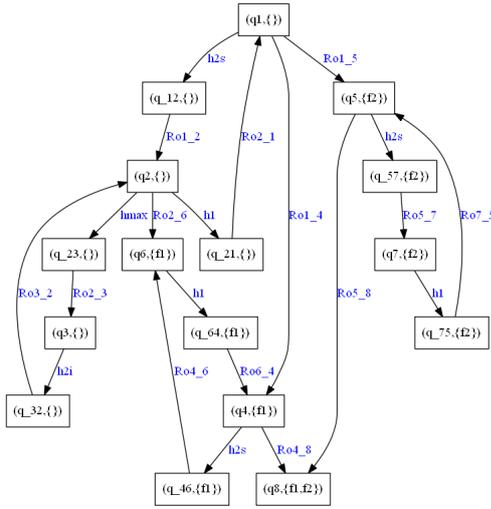


Figure 7: Diagnoser state tracker

Left hand side of Figure 6 shows the diagnoser belief state just before and after the fault f_1 occurrence. Results are consistent with the scenario: before 3500h, the belief states of the diagnoser are always tagged with a nominal diagnosis. After 3500h, all the states are tagged with f_1 .

Middle of Figure 6 illustrates the predicted date of fault occurrence (d_{f_1} and d_{f_2}). At the beginning of the process, the prognosis result is: $\Pi_0 = (\{f_1, 4120\}, \{f_2, 5105\})$. It can be noted that the predicted dates d_{f_1} and d_{f_2} of f_1 and f_2 globally increase. Indeed, the system oscillates between stressful modes and less stressful modes. To make it simple, we can consider that in some modes, the system does not degrade, so the predicted dates of f_1 and f_2 are postponed. Before 3500h, the predicted date of f_1 is lower than the one of f_2 . After 3500h, the predicted date of f_2 is updated, knowing that the system is in a degraded mode. Finally, the prognosis result is $\Pi_{3501} = (\{f_2, 5541\})$. Figure 6 shows the evolution of the RUL of the system. At $t = 3501$, as the fault f_2 is estimated to occur at $t = 5541$, the system RUL at $t = 3501$ is $5541 - 3501 = 2040$ h.

7 Conclusion

HYDIAG is a software developed in Matlab, with Simulink, by the DISCO team, at LAAS-CNRS. This tool has been extended into HYDIAGPRO to simulate, diagnose and prognose hybrid systems using model-based techniques. Some

results on an academic example are exposed in the paper. An extension to active diagnosis is also presented. The active diagnosis algorithm is currently tested on a concrete industrial case. HYDIAG and its user manual will be soon available on the LAAS website.

References

- [1] M. Bayouhd, L. Travé-Massuyès, and X. Olive. Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *IFAC World Congress*, 2008.
- [2] P. Ribot, Y. Pencolé, and M. Combacau. Diagnosis and prognosis for the maintenance of complex systems. In *IEEE International Conf. on Systems, Man, and Cybernetics*, 2009.
- [3] Elodie Chanthery and Pauline Ribot. An integrated framework for diagnosis and prognosis of hybrid systems. In *the 3rd Workshop on Hybrid Autonomous System (HAS)*, 2013.
- [4] S. Zabi, P. Ribot, and E. Chanthery. Health Monitoring and Prognosis of Hybrid Systems. In *Annual Conference of the Prognostics and Health Management Society*, 2013.
- [5] M. Bayouhd, L. Travé-Massuyès, and Xavier Olive. Active diagnosis of hybrid systems guided by diagnosability properties. In *the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, 2009.
- [6] E. Chanthery, Y. Pencolé, and N. Bussac. An AO*-like algorithm implementation for active diagnosis. In *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS*, 2010.
- [7] T. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.
- [8] M. Sampath, R. Sengputa, S. Lafortune, K. Sinnamohideen, and D. Teneketsis. Diagnosability of discrete-event systems. *IEEE Trans. on Automatic Control*, 40:1555–1575, 1995.
- [9] M Staroswiecki and G Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37(5):687–699, 2001.
- [10] M. Maiga, E. Chanthery, and L. Travé Massuyès. Hybrid system diagnosis : Test of the diagnoser hydiag on a benchmark of the international diagnostic competition DXC 2011. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, 2012.
- [11] P. Ribot and E. Bensana. A generic adaptative prognostic function for heterogeneous multi-component systems: application to helicopters. In *European Safety & Reliability Conference*, Troyes, France, September 18-22 2011.

Author index

A

Abdelkrim Mohamed Naceur	261
Abreu Rui	193, 209
Agudelo Carlos	241
Alonso-Gonzalez Carlos	59
Archer Dave	193
Aubrun Christophe	261

B

Berdjag Denis	159
Biswas Gautam	27, 75, 185
Blesa Joaquim	67
Bobrow Daniel	193
Boussaid Boumedyen	261
Bregon Anibal	59, 201
Bunte Andreas	185
Burke David	193

C

Cabassud Michel	253
Carrera Rolando	145
Cayetano Raúl	145
Chanthery Elodie	281
Chouiref Houda	261
Christopher Cody	119
Corruble Vincent	35
Cossé Ronan	159

D

Dague Philippe	51
Dahhou Boutaïeb	253
Daigle Matthew	201
De Kleer Johan	193, 225
Duvivier David	159

E

Eickmeyer Jens	43, 217
El Fallah Seghrouchni Amal	35
Eldardiry Hoda	193, 209

F

Feldman Alexander	127, 193
Feng Wenquan	27
Filasova Anna	235
Fourlas George	177

G

Ganguli Anurag	225
Gaurel Christian	159
Givehchi Omid	43
Gomez Pablo	11
Grastien Alban	105, 119
Grigoleit Florian	91
Grill Tanja	11
Guan Xiumei	27

H

Hanley John	193
He Zhangming	137
Herpson Cédric	35
Honda Tomonori	193, 209, 225

I

Ibrahim Hassan	51
Iverson Jonathan	209

J

Jannach Dietmar	3
Jauberthie Carine	67, 83
Jimenez Fernando	241
Jung Daniel	75

K

Kalech Meir	113, 247
Karras George	177
Khorasgani Hamed	75, 185
Kinnebrew John	185
Koitz Roxane	167
Krokavec Dusan	235
Kyriakopoulos Kostas	177

L

Le Gall Françoise	67
Le Lann Marie-Véronique	19, 269
Li Peng	43
Li Shuxing	137
Li Ze-Tao	253
Liao Linxia	209
Liscinsky Pavol	235

M

Maier Alexander	217
Matei Ion	225
Mishali Amir	247
Monrousseau Thomas	269
Mühlbacher Clemens	153

N

Niggemann Oliver	43, 185, 217
------------------	--------------

O

Ocampo-Martinez Carlos	99
------------------------	----

P

Pavel Radu	209
Pencolé Yannick	83, 277, 281
Perez Alexandre	193
Pethig Florian	43
Piechowiak Sylvain	159
Pons Renaud	83
Provan Gregory	127
Puig Vicenç	99, 261
Pulido Belarmino	59

R

Ribot Pauline	83, 281
Roux Elisa	19
Roychoudhury Indranil	201

S

Saha Bhaskar	209
Santos Simón Jorge	153
Schmitz Thomas	3
Serbak Vladimir	235
Shchekotykhin Kostyantyn	3
Shinitzky Hilla	113
Simon Laurent	51
Steinbauer Gerald	153
Stern Roni	113, 247
Struss Peter	91
Subias Audine	241

T

Travé-Massuyès Louise	19, 67, 83, 241, 269, 281
Traxler Patrick	11

V

Vasquez John William	241
Verde Cristina	145
Volgmann Sören	185

W

Wang Jiongqi	137
Wotawa Franz	167

Z

Zhang Mei	253
Zhao Hongbo	27
Zhou Gan	27
Zhou Haiyin	137

