Context-dependent Incremental Intention Recognition through Bayesian Network Model Construction

Han The Anh and **Luís Moniz Pereira**
Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
$h.anh@fct.unl.pt, lmp@di.fct.unl.pt$

Abstract

We present a method for context-dependent and incremental intention recognition by means of incrementally constructing a Bayesian Network (BN) model as more actions are observed. It is achieved with the support of a knowledge base of readily maintained and constructed fragments of BNs. The simple structure of the fragments enables to easily and efficiently acquire the knowledge base, either from domain experts or automatically from a plan corpus. We exhibit experimental results improvement for the Linux Plan corpus. For additional experimentation, new plan corpora for the iterated Prisoner's Dilemma are created. We show that taking into account contextual information considerably increases intention recognition performance.

1 INTRODUCTION

We propose a method for intention recognition in a dynamic, real-world environment. An important aspect of intentions is *future-directedness*, i.e. if we intend something now, we mean to execute a course of actions to achieve something in the future [3]. Most actions may be executed only at a far distance in time. During that period, the world is changing, and the initial intention may be changed to a more appropriate one or even abandoned [3, 6]. An intention recognition method should take into account these changes, and may need to reevaluate the intention recognition model depending on some time limit; in addition, as new actions are observed, the model should be reconfigurable to incorporate them.

Generally, *intention recognition* (also called *goal recognition*) is defined as the process of becoming aware of the intention of another agent and, more technically, as the problem of inferring an agent's intention through its actions and their effects on the environment [10]. *Plan recognition* is closely related to intention recognition, extending it to also recognize the plan the observed agent is following in order to achieve his intention [20]. Intention recognition is performed in domains in which it is better to have a fast detection of just the user's goal/intention rather than a more precise but time consuming detection of the complete user's plan, e.g. in the interface agents domain [12].

In this work, we use Bayesian Networks (BN) as the intention recognition model. The flexibility of BNs for representing probabilistic dependencies and the efficiency of inference methods for BN have made them an extremely powerful and natural tool for problem solving under uncertainty [16, 17]. We present a knowledge representation method to support incremental BN construction for performing intention recognition during runtime, from an initially given domain knowledge base. As more actions are observed, a new BN is constructed reinforcing some intentions whilst ruling out others (Section 3). This incremental method allows domain experts to specify knowledge in terms of small and simple BN fragments, which can be easily maintained and changed. Alternatively, these fragments can be learned from data. Our intention recognition method is evaluated on the Linux Plan corpus [2] (Section 5) and on our new, so-called IPD plan corpora (Section 6). We also propose a method to represent relationship among intentions when considering the case that agents may pursue multiple intentions simultaneously (Section 4). It is an indispensable aspect, but mostly omitted in prior works, which also allows us to sometimes significantly decrease the complexity of the probability inference [7].

It is inspired in that knowledge experts often consider a related set of variables together, and organize domain knowledge in larger chunks. An ability to represent conceptually meaningful groupings of variables and their interrelationships facilitates both knowledge elicitation and knowledge base maintenance [14, 13]. To this end, there have been several methods proposed for BN construction from small and easily maintained network fragments [16, 19, 14, 15, 13]. In essence, a combination of BNs is a graph that includes all nodes and links of the networks,

where nodes with the same name are combined into a common node. The main issue for a combination method is how the influence of different parents of the common node can be combined in the new network, given the partial influence of each parent in the corresponding fragment. The most popular method is Noisy-Or, firstly proposed by [16] for BNs of Boolean variables, and generalized by [22] for the general case of arbitrary domains.

2 BAYESIAN NETWORKS

Definition 1 *A BN is a pair consisting of a directed acyclic graph (DAG) whose nodes represent variables and missing edges encode conditional independencies between the variables, and an associated probability distribution satisfying the Markov assumption of conditional independence, saying that variables are independent of non-descendants given their parents in the graph [16, 17].*

In a BN, associated with each node of its DAG is a specification of the distribution of its variable, say A, conditioned on its parents in the graph (denoted by $pa(A)$)—i.e., $P(A|pa(A))$ is specified. If $pa(A) = \emptyset$ (A is called *root* node), its unconditional probability distribution, $P(A)$, is specified. These distributions are called Conditional Probability Distribution (CPD) of the BN.

The joint distribution of all node values can be determined as the product of conditional probabilities of the value of each node on its parents $P(X_1, ..., X_N) = \prod_{i=1}^{N} P(X_i|pa(X_i))$, where $V = \{X_i|1 \leq i \leq N\}$ is the set of nodes of the DAG.

Suppose there is a set of evidence nodes (i.e. their values are observed) in the DAG, say $O = \{O_1, ..., O_m\} \subset V$. We can determine the conditional probability distribution of a variable X given the observed value of evidence nodes by using the conditional probability formula

$$P(X|O) = \frac{P(X,O)}{P(O)} = \frac{P(X,O_1,...,O_m)}{P(O_1,...,O_m)} \qquad (1)$$

where the numerator and denominator are computed by summing up the joint probabilities over all absent variables with respect to V.

3 INCREMENTAL INTENTION RECOGNITION

In [18], a general BN model for intention recognition is presented and justified based on Heinze's intentional model [10]. Basically, the BN consists of three layers: cause/reason nodes in the first layer (called *pre-intentional*), connecting to intention nodes in the second one (called *intentional*), in turn connecting to action nodes in the third (called *activity*) (Figure 1). In this work, we present a method for incrementally constructing such BN model for performing *incremental* intention recognition.

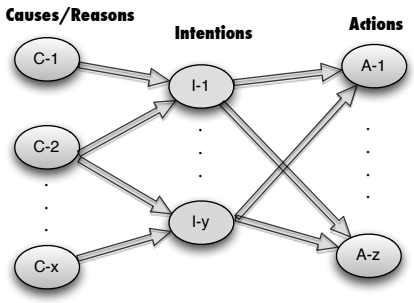

Figure 1: Bayesian Network for Intention Recognition.

Definition 2 (Intention Recognition BN – IRBN) *A BN for intention recognition (IRBN) W is a triple* $\langle \{Cs, Is, As\}, pa, P_W \rangle$ *where*

- Cs, Is *and* As *are the sets of cause/reason nodes, intention nodes and action nodes, respectively. They stand for binary random variables (i.e. their value is either* true (T) *or* false (F)*).*
- pa *is a mapping which maps a node to the set of its parent nodes such that:* $pa(C) = \emptyset \; \forall C \in Cs$; $pa(I) \subseteq Cs \; \forall I \in Is$; *and* $\emptyset \neq pa(A) \subseteq Is \; \forall A \in As$.
- *CPD tables are given by the probability distribution* P_W, *i.e.* $P_W(X|pa(X))$ *defines the probability of X conditional on* $pa(X)$ *in W,* $\forall X \in Cs \cup Is \cup As$.

The intention recognition method will be performed by incrementally constructing an IRBN as more actions are observed. The construction is based on a prior knowledge base consisting of Unit BN Fragments.

Definition 3 (Unit Fragments) *There are two types of unit fragments used for IRBN model construction:*

1. *A unit fragment* for an action A *consists of an intention I connecting to (i.e. causally affecting) A, and is denoted by* $UF_{\mathfrak{A}}(I, A)$.
2. *A unit fragment* for an intention I *consists of a context-independent and fixed over time set of causes/reasons* Cs *connecting to (i.e. causally affecting) I, and is denoted by* $UF_{\mathfrak{I}}(Cs, I)$.

Definition 4 (Knowledge Base) *The domain knowledge base* KB *consists of a set of actions* Δ, *a set of intentions* Υ, *a set of unit fragments for each action in* Δ *and a single unit fragment for each intention in* Υ, *satisfying that*

- *An intention I has a unique unit fragment in KB. The set of its parents (causes) and the CPD table associated with it are fixed. Let* $\mathfrak{C}(I)$ *denote the set of parents of I and* $P_{KB}(I|\mathfrak{C}(I))$ *define its CPD table.*

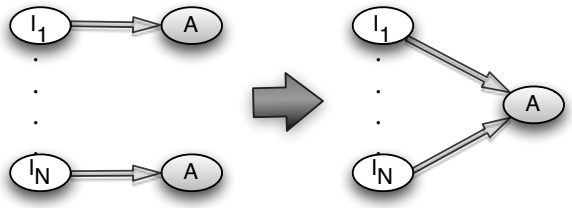

Figure 2: Noisy-OR Combination Method

- *A cause C has the same prior probability distribution in all the unit fragments (for intentions) that it belongs to, denoted by* $P_{KB}(C)$.

The simple structures of unit fragments enable domain experts to easily construct and maintain the knowledge base. The fragments also can be learnt from appropriate datasets, as we shall see later with the Linux and IPD corpora. Before presenting the intention recognition algorithm, let us define some (original) operators for handling CPD tables and IRBNs.

3.1 OPERATORS FOR CONSTRUCTING IRBN

As a new action A is observed, we need to incorporate it into the current IRBN. First, appropriate unit fragments for A are selected from KB. Let *select(A)* denote the set of *all* unit fragments for A from KB [1]. They are then combined using the Noisy-OR method [16, 22], thereby obtaining a BN with a single action A (Figure 2).

Definition 5 (Unit IRBN via Noisy-OR) *The* Unit IRBN *for action* A *is an IRBN with a single action, denoted by* $irBN(A) = \langle\{Cs, Is, \{A\}\}, pa, P_W\rangle$. *It is obtained via* Noisy-OR *method as follows. Let* $select(A) = \{UF_{\mathfrak{A}}(I_1, A), ..., UF_{\mathfrak{A}}(I_N, A)\}$ *and for* $1 \leq i \leq N$, $P(A = T|I_i = T) = q_i$ *(defined in fragment* $UF_{\mathfrak{A}}(I_i, A)$*). Then,*

- $Is = \{I_1, ..., I_N\}$; $Cs = \bigcup_{I \in Is} \mathfrak{C}(I)$;
- $pa(I) = \mathfrak{C}(I)\ \forall I \in Is$; $pa(A) = Is$;
- $P_W(C) = P_{KB}(C)\ \forall C \in Cs$; $P_W(I) = P_{KB}(I)\ \forall I \in Is$; *and, according to the* Noisy-OR *method,* $P_W(A = T|pa(A)) = 1 - \prod_{i:I_i=T}(1 - q_i)$.

The rationale and appropriateness of the application of the Noisy-OR method here for combining unit fragments is based on the intuition that each intention can be interpreted as a "*cause*" of action A; and action A occurs when one or more of the intentions are active. Detailed arguments for this can be found in [5, 16].

[1] The selection can be done in a context-dependent manner, but it is beyond the scope of this paper.

Definition 6 (Project of CPD Table) *Let Tb be a CPD table defining* $P(X|V)$*, the probability of a random variable X conditional on a set of random* binary *variables V, and* $V' \subsetneq V$. *The project of Tb on* V'*, denoted by* $\texttt{proj}(\texttt{Tb}, \texttt{V}')$*, is the part of Tb corresponding to all variables in* $V \setminus V'$ *being false.*

Now we need to combine the obtained unit IRBN, $\texttt{irBN}(\texttt{A})$, with the current IRBN. For that, in the sequel we define how to combine two IRBNs. Intuitively, we simply add up all the new nodes and links of the new IRBN to the current IRBN, keeping the CPD tables from the original IRBNs.

Definition 7 (Combination of IRBNs) *Let* $W_1 = \langle\{Cs_1, Is_1, As_1\}, pa_1, P_1\rangle$ *and* $W_2 = \langle\{Cs_2, Is_2, As_2\}, pa_2, P_2\rangle$ *be two IRBNs, such that* $As_1 \cap As_2 = \emptyset$ *(the actions in* As_2 *which are already in* As_1 *are renamed). The combination of these two IRBNs is an IRBN, denoted by* $\texttt{comb}(\texttt{W}_1, \texttt{W}_2) = \langle\{Cs, Is, As\}, pa, P_W\rangle$*, where*

- $As = As_1 \cup As_2$; $Is = Is_1 \cup Is_2$; $Cs = Cs_1 \cup Cs_2$;
- $pa(I) = \mathfrak{C}(I)\ \forall I \in Is$; $pa(A) = pa_1(A) \cup pa_2(A)$;
- $P_W(C) = P_{KB}(C)\ \forall C \in Cs$; $P_W(I|pa(I)) = P_{KB}(I|\mathfrak{C}(I))\ \forall I \in Is$; $P_W(A|pa(A)) = P_{W_k}(A|pa_k(A))$ *if* $A \in As_k$ *(with* $k = 1, 2$*).*

Note that here it is allowed the possibility that the observed agent follows multiple intentions simultaneously. When some intentions are found irrelevant—e.g. because they are much unlikely[2]—those intentions should be removed from the IRBN. This is enacted by considering them as completely false and employing the *project* operator.

Definition 8 (Remove Intentions from IRBN) *Let* $W = \langle\{Cs, Is, As\}, pa, P_W\rangle$ *be an IRBN and* $R \subset Is$ *a strict subset of* Is. *The result of removing the set of intentions R from W is an IRBN, denoted by* $\texttt{remove}(\texttt{W}, \texttt{R}) = \langle\{Cs_R, Is_R, As_R\}, pa_R, P_R\rangle$*, where*

- $As_R = As$; $Is_R = Is \setminus R$; $Cs_R = \bigcup_{I \in Is_R} \mathfrak{C}(I)$;
- $pa_R(I) = \mathfrak{C}(I)\ \forall I \in Is_R$; $pa_R(A) = pa(A) \setminus R\ \forall A \in As_R$;
- $P_R(C) = P_{KB}(C)\ \forall C \in Cs_R$; $P_R(I|pa_R(I)) = P_{KB}(I|\mathfrak{C}(I))\ \forall I \in Is_R$; *and for each* $A \in As_R$, $P_R(A|pa_R(A))$ *is defined by the CPD table* $\texttt{proj}(\texttt{Tb}, \texttt{pa}_\texttt{R}(\texttt{A}))$ *where Tb is the CPD table for A in W, i.e. defined by* $P_W(A|pa(A))$.

Based on these operators, we now describe an algorithm for incremental intention recognition in a real-time manner.

Incremental Intention Recognition Algorithm. Repeat the following steps until some given time limit is reached; the most likely intention in previous cycle is the final result.

[2] One intention is much less likely than the other if the fraction of its likelihood and that of the most likely intention is less than some small threshold. It is up to the KB designer to provide it.

- Let A be a new observed action. Combine the current IRBN W with $\texttt{irBN(A)}$ to obtain $\texttt{W}' = \texttt{comb(W, irBN(A))}$. If A is the initially observed action, let $\texttt{W}' = \texttt{irBN(A)}$.
- Compute the probability of each intention in W', conditional on the set of current observations in W'. Remove the intentions which are much less likely than the others (following Definition 8).

4 RELATION AMONG INTENTIONS

When considering the case in which the observed agent may pursue multiple intentions simultaneously, it is undoubtedly indispensable to take into account and express the relations amongst the intentions in the model. Pursuing one intention may exclude some other intention to be pursued. We introduce a so-called *exclusive relation e*—a binary relation on the set of intention nodes—representing that if one intention is pursued, then the other intention cannot be pursued. It is usually, although perhaps not always, the case that intentions exclusiveness is symmetric. Here we assume that e is symmetric; it can be renamed *mutually exclusive relation*.

Intentions I_1 and I_2 are mutually exclusive iff they cannot be pursued simultaneously, i.e. $P(I_1 = T, I_2 = T) = 0$. Thus, for any action A, if $I_1, I_2 \in pa(A)$ then the CPD table for A is undefined. Hence, the BN needs to be restructured. The mutually exclusive intentions must be combined into a single node since they cannot co-exist as parents of a node. Each intention represents a possible value of the new combined node. Namely, let $I_1, ..., I_t$ be such that $e(I_i, I_j),\ \forall i, j : 1 \leq i < j \leq t$. The new combined node, I, stands for a random variable whose possible outcomes are either $I_i,\ 1 \leq i \leq t$, or $\tilde{I}$—the outcome corresponding to the state that none of $I_i = T$. Note that if the intentions are exhaustive, $\tilde{I}$ can be omitted. Next, I is linked to all the action nodes that has a link from one of $I_i,\ 1 \leq i \leq t$.

There remains to re-define CPD tables in the new BN. They are kept the same for action A where $I \notin pa(A)$. For A such that $I \in pa(A)$, the new CPD table at $I = I_i$ corresponds to the CPD table in the original BN at $I_i = T$ and $I_j = F\ \forall j \neq i$, i.e. $P(A|I = I_i, ...) = P(A|I_0 = F, ..., I_{i-1} = F, \mathbf{I_i} = \mathbf{T}, I_{i+1} = F, ..., I_t = F, ...)$. Note that the left hand side is defined in the new BN, and the right hand side is defined in the original BN. Similarly, the new CPD table at $I = \tilde{I}$ corresponds to $I_i = F\ \forall 1 \leq i \leq t$. We now specify the CPD table of I. In the new BN, the causes/reasons of each intention are connected to the combined node, i.e. $pa(I) = \bigcup_{i=1}^{t} \mathfrak{C}(I_i)$. Applying the Markov assumption (Def.1) we have $P(I = I_i|pa(I)) = P_i(I_i = T|\mathfrak{C}(I_i))$ and $P(I = \tilde{I}|pa(I)) = \prod_{i=1}^{t} P_i(I_i = F|\mathfrak{C}(I_i))$, where P_i is the probability distribution of the unit fragment for I_i.

In the next section we focus on the single intention recognition case, showing how the approach to representing relationships amongst intentions can significantly decrease the complexity of the probability inference therein. We then present experimental results on the Linux Plan corpus. After that, in Section 6, we provide further experimentation on our novel, so-called IPD plan corpora.

5 SINGLE INTENTION RECOGNITION

5.1 THE MODEL

Suppose the observed agent pursues a single intention. In this case, all intentions are mutually exclusive, and they can be combined into a single node. The IRBN then has a single intention node, linking to all action nodes. All cause/reason nodes are connected to the intention node.

Let $I_1, ..., I_n$ be the intentions in the original IRBN. As usual, they are assumed to be exhaustive, i.e. the observed agent is assigned an intention from them. The combined node I thus has n possible outcomes $I_i,\ 1 \leq i \leq n$. Let $As = \{A_1, ..., A_m\}$ be the set of current observed actions. The set of all cause/reason nodes are $Cs = \cup_{i=1}^{n} \mathfrak{C}(I_i)$. Suppose $C_e \subseteq Cs$ is the set of cause/reason nodes which are observed (evidence nodes). Let $C_{ne} = Cs \setminus C_e$.

Applying Eq. 1, we obtain the probability of each intention conditional on the current observations, for $1 \leq j \leq n$,

$$P(I = I_j|C_e, As) = \frac{P(I_j, C_e, As)}{\sum_{i=1}^{n} P(I_i, C_e, As)}, \text{ where}$$

$$P(I_j, C_e, As) = \prod_{i=1}^{m} P(A_i|I_j) \left(\sum_{C_{ne}} P(I_j|Cs) \prod_{C \in Cs} P(C) \right)$$

This implies that, when not including causes/reasons of intentions ($Cs = \emptyset$) as in case of Linux Plan corpus below, our intention recognizer has a linear complexity $O(|n|)$.

If all the cause/reason nodes are not observable, i.e. $C_{ne} = Cs$ (as in the case of the Linux Plan we examine in the next subsection), it is easily seen that: $P(I_j, C_e, As) = P(I_j) \prod_{i=1}^{m} P(A_i|I_j)$. If all of them are observed ($C_{ne} = \emptyset$) (as we shall see in the IPD Plan corpora), the term $\prod_{C \in Cs} P(C)$ is simplified in the fraction. Thus, in these two cases, we do not need to define prior probabilities distribution of the root nodes in Cs. Note that in the latter case we still need to compute the conditional probabilities $P(I_j|Cs)$.

5.2 EXPERIMENTAL EVALUATION

The Linux Plan Corpus. Plan corpus is the term used to describe a set of plan sessions and consists of a list of goals/intentions and the actions a user executed to achieve them [1]. Although there are many corpora available for testing machine learning algorithms in other domains, just

a few are available for training and testing plan/intention recognizers; furthermore, each of the recognizers using plan corpora usually has its own datasets, leading to a difficult comparison among them. For that important reason, we chose Linux Plan corpus [2]—one of the rare regularly used plan corpora—which was kindly made publicly available by Nate Blaylock—to test our system. It enables a better comparison with other systems using this corpus [2, 1].

The Linux plan corpus was gathered from 56 human users. The users have different levels of expertise in the use of Linux, and they were allowed to perform as many times as they wished, in order to contribute more plan sessions. The sessions, consisting in sequences of commands performed by the users to achieve a given goal/intention, were automatically recorded. At the end of each session, the users were asked to indicate whether they succeeded in achieving their goal. In total, there are 547 sessions, 457 of which were indicated as successfully completing the goal, 19 goals and 43 actions.

The Linux Plan corpus is an important (especially in the interface-agents domain [12]) and hard benchmark for intention/goal recognition. First, data is collected from real humans and thus noisy. Second, involved humans expertise is varied, and they sometimes used wrong commands due to limited knowledge about the domain [2]. Furthermore, we observe that plan sessions' lengths in the corpus are quite varied. The minimum, maximum, and mean number of actions of a plan session are 1, 60, and 6.124, respectively.

Learning Unit Fragments from Data. For unit fragment $UF_{\mathfrak{A}}(I, A)$, the conditional probability of A given I is defined by the frequency of A in a plan session for achieving the goal/intention I divided by the frequency of any action for achieving I: $P(A = T|I = T) = \frac{freq(A_I)}{freq(I)}$. For better understanding, in the plan corpus each action is marked with the intention which the action is aiming at. Then, $freq(A_I)$ is the frequency of A being marked by I, and $freq(I)$ is the frequency of seeing the mark I.

Prior probabilities of all the intentions in the corpus are given initially, and used for generating tasks for users [2].

Making Predictions. Similar to [2], instead of letting the recognizer make a prediction after each observed action, we set a *confidence* threshold τ ($0 \leq \tau \leq 1$) , which allows the recognizer to decide whether or not it is confident enough to make a prediction; the recognizer only makes a prediction if the likelihood of the most likely intention in the model is greater than τ. Otherwise, it predicts "don't know". In addition, instead of only predicting the most likely intention, the recognizer provides a set of N most likely ones (*N-best prediction*).

Evaluation Metrics. For evaluating our system and comparing with the previous ones [2, 1], we use three different metrics. *Precision* and *recall* report the number of correct predictions divided by total predictions and total prediction opportunities, respectively. More formally (also see [1]), let $Seq = a_1, ..., a_n$ be a sequence of actions (plan session) achieving intention I. Considering N-best prediction case, let $correct(A) = 1$ if I is one of N most likely intentions, and 0 otherwise. Then, precision and recall for *Seq* are defined as: $precision(Seq) = (\sum_{i=1}^{n} correct(a_i))/z$; $recall(Seq) = (\sum_{i=1}^{n} correct(a_i))/Z$, where z and Z are the number of predictions made (when the recognizer is confident enough) and the total number of prediction opportunities (i.e. when $\tau = 0$), respectively.

Table 1: Intention Recognition Results on the Linux Plan Corpus

N-best	1-best	2-best	3-best	4-best
τ	0.95	0.5	0.45	0.42
Precision	0.786	0.847	0.870	0.883
Recall	0.308	0.469	0.518	0.612
Converg.	0.722	0.799	0.822	0.824

On the other hand, *convergence* is a metric that indicates how much time the recognizer took to converge on what the current user goal/intention was. Let t be such that $correct_i = 0$ for $0 \leq i \leq t-1$ and 1 for $t \leq i \leq n$ (i.e. t is the first time point which from there on the system always correctly predicts), *convergence* for *Seq* is defined as: $convergence(Seq) = (z - t + 1)/z$.

Finally, the *overall* precision, recall and convergence are obtained by taking averages over all testing sessions.

Experiments and Results. Because of the small size of the Linux corpus, similar to previous works, we ran experiments using the one-out cross validation method [1].

Table 1 shows the results for different values of N (and the corresponding value of τ). Similar to the previous works [2, 1], we keep the best results for each value of N w.r.t. τ. For example, we obtained a precision of 78.6% for 1-best that is increased to 87.0% for 3-best prediction and 88.3% for 4-best one. Convergence is increased from 72.2% for 1-best to 82.2% for 3-best and 82.4% 4-best prediction.

The best performance on the Linux corpus (namely, in terms of precision and convergence) so far was reported in [1], where the authors use variable Markov model with exponential moving average. Here we got an increment of 14% better precision and 13.3% better convergence for 1-best prediction, 8.2% better precision and 9.3% better convergence for 2-best prediction, and 7.5% better precision and 7.7% better convergence for 3-best prediction. We also obtained better recalls comparing with [2] in all cases.

The Linux corpus allows an appropriate comparison with existent works. However, it does not include contextual information (reasons/causes of intentions), and there is no intention change/abandonment occurrences (users follow a

single intention throughout entire plan sessions). To evaluate the context-dependent aspect as well as the capability of dealing with intention change/abandonment, we next present new plan corpora.

6 IPD PLAN CORPORA

We present new plan corpora in the context of iterated Prisoner's Dilemma (IPD) [21] and provide experimental results for them. The intentions/goals to be recognized are the (known) strategies in IPD (see below). Plan sessions are sequences of moves played by such strategies.

6.1 ITERATED PRISONER'S DILEMMA

Prisoner's Dilemma (PD) is a symmetric two-player non-zero game defined by the payoff matrix

$$\begin{array}{c} \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cc} C & D \end{array} \\ \begin{pmatrix} R,R & S,T \\ T,S & P,P \end{pmatrix} \end{array}$$

Each player has two options in each round, cooperates (C) or defects (D). A player who chooses to cooperate with someone who defects receives the sucker's payoff S, whereas the defecting player gains the temptation to defect, T. Mutual cooperation (resp., defection) yields the reward R (resp., punishment P) for both players. PD is characterized by the payoff ranking $T > R > P > S$ (and, in addition, $2R > S + T$ for IPD). Thus, in a single round, it is always best to defect, but cooperation may be rewarded if the game is iterated. Let r denote the (average) number of rounds the game is iterated.

IPD is usually known as a story of tit-for-tat (TFT), which won both Axelrod's tournaments [21]. *TFT* starts by cooperating, and does whatever the opponent did in the previous round. It will cooperate if the opponent cooperated, and will defect if the opponent defected. But if there are erroneous moves due to noise (i.e. an intended move is wrongly performed with a given execution error), the performance of *TFT* declines: it cannot correct errors or mistakes. Tit-for-tat is then replaced by generous tit-for-tat (GTFT), a strategy that cooperates if the opponent cooperated in the previous round, but sometimes cooperates even if the opponent defected (with a fixed "forgiveness" probability $p > 0$) [21]. *GTFT* can correct mistakes. Subsequently, *TFT* and *GTFT* were replaced by win-stay-lose-shift (WSLS) as the winning strategy chosen by evolution [21]. *WSLS* repeats the previous move whenever it did well, but changes otherwise. Some other less famous strategies (which we are going to use later) are *GRIM* – a grim version of *TFT*, prescribing to defect except after a round of mutual cooperation, and Firm-But-Fair (FBF) – known as a tolerant brother of *TFT*, prescribing to defect only if getting a sucker's payoff S in previous round. Details of all strategies described above can be found in [21] (Chapter 3).

Next, we describe how training and testing plan corpora are created employing these strategies.

6.2 CORPUS DESCRIPTION

We made an assumption that all strategies to be recognized have the memory size bounded-up by M ($M \geq 0$)—i.e. their decision at the current round is independent of the past rounds that are at a time distance greater than M. The strategies described above have memory $M = 1$. Abusing notations, R, S, T and P are referred to as game states (in a single round or interaction). We too use E (standing for *empty*) to refer to the game state having had no interaction.

An action in the corpus is of the form $s_1...s_M\xi$, where $s_i \in \{E, R, T, S, P\}$, $1 \leq i \leq M$, are the states of the M last interactions, and $\xi \in \{C, D\}$ is the current move. We denote by Σ_M the set of all possible types of action. E.g, $\Sigma_1 = \{EC, RC, TC, SC, PC, ED, RD, TD, SD, PD\}$. This encoding method enables to save the game states without having to save the co-player's moves, thus simplifying the corpus representation, described below.

Suppose we have a set of strategies to be recognized. The plan corpus for this set consists of a set of plan sessions generated for each strategy in the set. A plan session of a strategy is a sequence of actions played by that strategy (more precisely, a player using that strategy) against an arbitrary player. As an example, let us consider *TFT* and the following sequence of its interactions with some other player (denoted by X), in the presence of noise

round :	0	1	2	3	4	5
TFT :	–	C	C	D	D	D
X :	–	C	D	D	C	D
TFT-states :	E	R	S	P	T	P

The corresponding plan session for *TFT* is $[EC, RC, SD, PD, TD]$. At 0-th round, there is no interaction, thus the state is E. *TFT* starts by cooperating (1-st round), hence the first action of the plan session is EC. Since player X also cooperates in the 1-st round, the game state at this round is R. *TFT* reciprocates in the 2-nd round by cooperating, hence the second action of the plan session is RC. Similarly for the third and the fourth actions. Now, at the 5-th round, *TFT* should cooperate since X cooperated in 4-th round, but because of noise, it makes an error to defect. Therefore, the 5-th action is TD.

6.3 PLAN CORPORA GENERATION

Let us start by generating a plan corpus for seven most popular strategies within the IPD framework: *AllC* (always cooperate), $AllD$ (always defect), *TFT*, *GTFT* (probability of forgiving a defect is $p = 0.5$), *WSLS*, *GRIM* and *FBF*.

We collect plan sessions of each strategy by playing a random move (C or D) in each round with it. To be more thor-

ough, we can also play all possible combinations for each given number of rounds r. E.g, if $r = 5$, there are 2^5 combinations: C or D in each round. When noise is present, each combination is played repeatedly several times.

The training corpus to be used here is generated by playing with each strategy all the possible combinations 10 times, for each number of rounds r from 5 to 10. The testing dataset is generated by playing a random move with each strategy in each round, also for r from 5 to 10. We continue until obtaining the same number of plan sessions as of the training dataset (corpus). Both datasets are generated in the presence of noise (namely, an intended move is wrongly performed with probability 0.05).

In this testing dataset, intention (strategy) changes/abandonment are not taken into account. The players use the same strategy in all the rounds. We refer to this testing dataset as `Testset-IRFIX`. For testing the context-dependent aspect of our intention recognizer, as well as taking into account intention changes/abandonment, we next introduce the concept of *social learning* within the framework of evolutionary game theory [11].

6.4 SOCIAL LEARNING

In social learning, individuals in a population can observe the behavior of others and the outcomes of those behaviors. They copy the behavior of others whenever these appear to be more successful [21]. The accumulated payoff from all interactions emulates the individual *fitness* or social *success* and the most successful individuals will tend to be imitated by others. There are many ways to model social learning [11, 21]. The most popular one is implemented using the so-called pairwise comparison rule [21]: an individual **A** with fitness f_A will adopt the strategy of a randomly chosen individual **B** with fitness f_B with a probability given by the Fermi function (from statistical physics): $p(f_A, f_B) = \left(1 + e^{-\beta[f_B - f_A]}\right)^{-1}$, where the quantity β controls the "imitation strength", i.e. how strongly the players are basing the decision to imitate on payoff comparisons. Henceforth, **A** and **B** are referred to as imitating and imitated individuals, respectively. For simplicity, we use $\beta = 1$ for the rest of this paper: the imitation depends on comparing the exact payoffs.

It is now allowed the possibility that a player can change his/her strategy (intention) by imitating the randomly met player's strategy (intention), depending on how the latter player is more successful. The two players' ongoing success difference (SD) causally affects the imitating player's current intention. In addition, this intention is causally affected by the so-called imitation event (IE), stating whether the player is meeting some other player for learning/imitating. Now we have an IRBN with two cause/reason nodes, a single intention node, and observed action nodes (Figure 3).

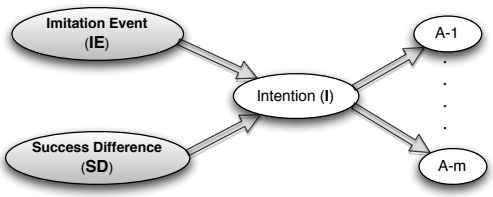

Figure 3: IRBN in IPD Context

We define the conditional probability distribution $P(I_i|IE, SD)$. If the player does not meet any other player for imitation (i.e. $IE = F$), I_i is independent of the success difference *SD*: $P(I_i|IE = F, SD) = P(I_i|IE = F)$. Now, let us consider the case $IE = T$. If the successes are also observable (thus, *SD* is observed, say, equal χ)[3], but the strategy of the imitated player is not, we have

$$P(I_i|IE = T, SD = \chi) = (1-u)p_i + \frac{u}{S-1}\sum_{j\neq i} p_j \quad (2)$$

where $u = (1+e^{-\chi})^{-1}$; p_i is the probability that I_i was the player's intention in the last prediction; and S is the number of strategies in the corpus. The formula is explained as follows. With probability $(1-u)p_i$ the imitating player's strategy remains I_i. Moreover, not being observed, the probability that I_i was the imitated player's strategy is (assumed) equal $1/(S-1)$. The second term expresses the probability that the player adopts the new strategy I_i by imitation.

Now, in case the imitated player's strategy is also observable, denoted by $I_{i^\star}$, similarly we have

$$\begin{aligned} P(I_{i^\star}|IE = T, SD = \chi) &= (1-u)p_i + u\sum_{j\neq i^\star} p_j \\ P(I_i|IE = T, SD = \chi) &= (1-u)p_i \quad \forall\, i \neq i^\star \end{aligned} \quad (3)$$

Testing Dataset. The testing dataset in this setting is generated by using a simplified evolutionary simulation as follows. We play a random choice with each of the seven above mentioned strategies for 10 rounds. The payoff of each strategy is accumulated over all the rounds. Then, for each strategy, another strategy is randomly chosen from the other six for imitation using the pairwise comparison rule. After all the seven strategies are given the chance to change their strategy (imitate another), the interactions are repeated for 10 more rounds. At the 10-th round, we save the accumulated payoff values of the imitating and imitated strategies. We experiment until obtaining the same number of plan sessions as in the training dataset. The PD payoff

[3]There may be noise in the evaluation of the successes. The observed value χ of SD is randomly taken in the range $((1-\epsilon)\chi_1, (1+\epsilon)\chi_1)$, where ϵ is a small positive number (here we use $\epsilon = 0.01$) and χ_1 is the exact value of the difference.

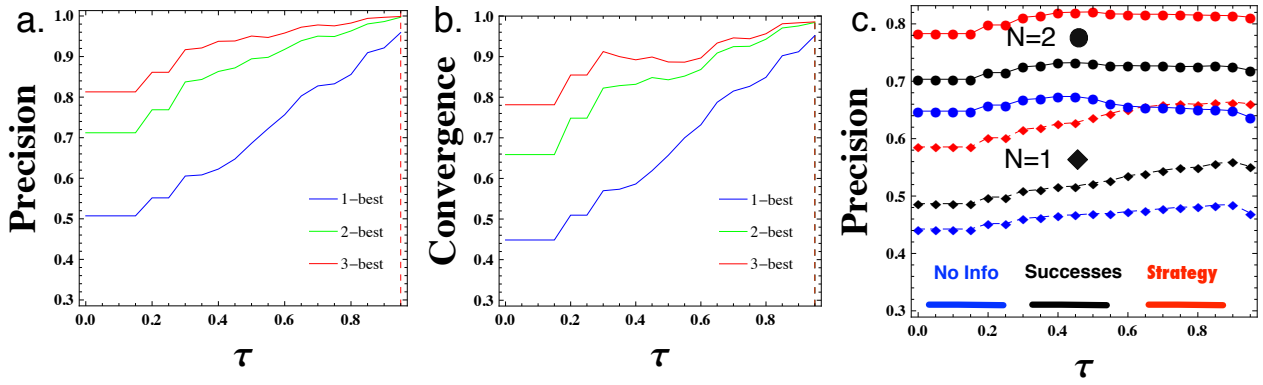

Figure 4: Panels (a) and (b): Precision and convergence for $\tau \in [0, 1]$ and for different values of N (N = 1,2,3) with respect to `Testset-IRFIX` dataset. Panel (c): Precision for different levels of contextual information, for $\tau \in [0, 1]$, with respect to `Testset-IRCHANGE` dataset. We consider $N = 1$ (dashed diamond) and $N = 2$ (circle).

matrix being used: $T = 20$, $R = 15$, $P = 10$, $S = 5$; and $noise = 0.05$. This testing dataset is referred to as `Testset-IRCHANGE`.

6.5 RESULTS

The intention recognition model is acquired using the training corpus. Figures 4a and 4b show the precision and convergence of the model with respect to the `Testset-IRFIX`. Given that the training as well as the testing datasets are generated in presence of noise, the achieved performance is quite good. Namely, for big enough τ, both precision and convergence scores are greater than 0.9, even for the 1-best case.

In Figure 4c we show the effects of having different levels of contextual information on the intention recognition performance, using `Testset-IRCHANGE` dataset. Namely, in the first setting (blue curves), there is no information about the imitation event (IE) – it is not known if the recognized player may imitate and adopt another strategy. In the second setting (black curves), IE and the successes are observable. In the third setting (red curves), the strategy of the imitated player is also observable. It is clearly shown that the performance is considerably increased as more contextual information is available. Namely, comparing with the first setting where no contextual information is taken into account, an increase of about 5% and 15% precision is achieved in the second and third settings, respectively.

7 RELATED WORK

Bayesian Networks have been one of the most successful models applied for the intention/plan recognition problem, e.g. in [4, 6]. Depending on the structure of plan libraries, a knowledge-based model construction is employed to build BNs from the library—which is then used to infer the posterior probability of explanations (for the set of observed actions). These works address a number of important issues in intention/plan recognition (see [6] for details), but they made several assumptions for the sake of computational efficiency. First, prior probabilities of intentions are assumed to be fixed. This assumption is not reasonable because those prior probabilities should depend on the situation at hand [3], and can be captured by causes/reasons of the intentions as in our work. Second, intentions are assumed to be independent of each other. This is not generally the case since the intentions may support or exclude one another. Those works hence do not appropriately address multiple intention recognition. This latter assumption must always, explicitly or implicitly, be made by the approaches based on (Hidden) Markov Models, e.g. [1], or statistical corpus-based machine learning [2]. Generally, in those approaches, a separate model is built for each intention; thus no relations amongst the intentions are expressed or can be expressed. These works were restricted to the single intention case.

Different from all above mentioned works, our model is *context-dependent*, which is achieved by including in it causes/reasons of intentions. This way, our model can appropriately deal with the abandonment/changes of intentions—when the causes/reasons do not support or force the intending agent to hold those intentions anymore.

8 CONCLUDING REMARKS AND FUTURE WORKS

We have presented a novel method for incremental and context-dependent intention recognition. The method is performed by dynamically constructing a BN model for intention recognition from a prior knowledge base consisting of easily maintained fragments of BN. We have evaluated

the method on the Linux Plan corpus and compared with previous works. In general, our performance is better than all existent ones that make use of the corpus.

For further experimentation, we have created the so-called IPD plan corpora for the famous strategies in the context of the iterated Prisoner's Dilemma. We employed the famous model of (human) behaviors by means of social learning and evolutionary game theory to simulate intention changes/abandonment—enabling us to evaluate the context-dependent aspect of our intention recognizer and as well as its capability of dealing with intention changes/abandonment. Our experimental results show that taking into account contextual information is crucial, enabling to achieve significant recognition improvements.

The good performance of our method with respect to the Linux corpus shows its applicability to the important interface-agents domain [12]. In addition, given that PD and other social dilemmas [21] are regularly found in real life [11, 21], its good performance for the IPD corpora makes it highly applicable for a wide range of application domains, as diverse as Economics (e.g. recognizing companies policies), Psychology and Biology (e.g. the role of intention recognition in the evolution of cooperation, as our recent works exhibit in [8, 9], using the intention recognition methods described in this paper).

In Section 4 we made an implicit assumption that the intentions to be combined are perfectly mutually exclusive. This assumption can be relaxed by utilizing a latent variable for any subset of perfectly mutually exclusive intention nodes. We are exploring this direction to provide a more general method for representing relationships amongst intention nodes.

9 Acknowledgments

We thank the reviewers for useful comments. HTA acknowledges the support from FCT-Portugal, grant SFRH/BD/62373/2009.

References

[1] M.G. Armentano and A. Amandi. Goal recognition with variable-order markov models. In *IJCAI'09*.

[2] N. Blaylock and J. Allen. Statistical goal parameter recognition. In *ICAPS04*, pages 297–304.

[3] M. E. Bratman. *Intention, Plans, and Practical Reason*. The David Hume Series, CSLI, 1987.

[4] E. Charniak and R.P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1), 1993.

[5] F. G. Cozman. Axiomatizing noisy-or. In *ECAI'04*.

[6] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173:1101–1132, 2009.

[7] V. Gogate and R. Dechter. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

[8] T. A. Han, L. M. Pereira, and F. C. Santos. The role of intention recognition in the evolution of cooperative behavior. In *IJCAI'2011*.

[9] T. A. Han, L. M. Pereira, and F. C. Santos. Intention recognition promotes the emergence of cooperation. *Adaptive Behavior*, 2011.

[10] C. Heinze. *Modeling Intention Recognition for Intelligent Agent Systems*. PhD thesis, The University of Melbourne, Australia, 2003.

[11] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge U. P., 1998.

[12] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *UAI'98*, pages 256–265, 1998.

[13] K. B. Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140 – 178, 2008.

[14] K.B. Laskey and S.M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *UAI'97*, 1997.

[15] S. Natarajan, P. Tadepalli, T. G. Dietterich, and A. Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artificial Intelligence*, 54:223–256, 2008.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

[17] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge U.P, 2000.

[18] L. M. Pereira and T. A. Han. Intention recognition with evolution prospection and causal Bayesian networks. In *Computational Intelligence for Engineering Systems*, pages 1–33. Springer, 2011.

[19] A. Pfeffer, D. Koller, B. Milch, and Ken T. Takusagawa. Pook: A system for probabilistic object-oriented knowledge representation. In *UAI'99*, 1999.

[20] F. Sadri. Logic-based approaches to intention recognition. In *Handbook of Research on Ambient Intelligence: Trends and Perspectives*. 2010.

[21] Karl Sigmund. *The Calculus of Selfishness*. Princeton U. Press, 2010.

[22] S. Srinivas. A generalization of the noisy-or model. In *UAI'93*, 1993.