

PNRD and iPNRD Integration Assisting Adaptive Control in Block World Domain

José Jean Paul Zanlucchi de Souza Tavares and Gabriel de Almeida Souza

Universidade Federal de Uberlândia, Av. João Naves de Ávila, 2121, 38400902
Uberlândia Brazil

{jean.tavares,gabrielsouzaworking}@ufu.br

Abstract. Too much effort is being invested in the automated planning field, but this area has been focusing more on improving the performance of search algorithms in an abstract space than on applying automated planning in practice. Among several challenges, online replanning is an open issue. This paper presents *PNRD/iPNRD* integration in Blocks World Domain in order to create an adaptive control of a robotic arm and passive agents. The procedure is composed of building a Petri space, *PNRD*, and *iPNRD* model for each agent, and by integrating their information so that the robotic arm may autonomously order the blocks to the desired final global state, stored previously in each block's RFID tag; comparing logical and physical information, generating a sequence of movements, and executing related actions. Some issues arise because of the dispersed information, physical-logical interface and its constraints, requiring the detection of non-conformant results, optimally solve and realize the required sorting given its partial observability, and sequential movement of one block each time. The feedback from *PNRD/iPNRD* models with physical positioning can certify the required initial, intermediate and final condition and also check for exceptions, resulting in an adaptive robotic control. This didact example points out where this approach can work with automated planning for a more complex system.

Keywords: PNRD/iPNRD · Blocks World · Adaptive Control

1 Introduction

It is a well known fact that the AI planning community is very committed to apply the developments already achieved in this area to real complex applications. However realistic planning problems bring challenges not only during design processes but also for the automated planners during the planning itself [6].

According to [1] there are many reasons for the low adoption of automated planning in the industry. First, they rely on the widely used 'preconditions-and-effects' representations that provide predictive models of actions often too abstract to be of use by actors. The automated planning field has been focusing more on improving the performance of search algorithms in an abstract space

than on improving its models' predictive capabilities. Planning is easily formalized, whereas acting is not. There is a clear frontier between planning and performing, but the borderline between acting and performing is more blurred. In addition, automated plans are usually a simple packaged input–output function, although actor's deliberation functions are difficult to integrate to open and dynamic environments. Such changes on focus entail two interconnected principles: a hierarchical structure to integrate the actor's deliberation function, and the continual online planning and reasoning throughout the acting process. [9] also questioned when automatic planners would be integrated with industrial systems. However, online process redesign and replanning is still an open issue. In order to investigate a way to increase automated planning in the industry, this paper presents how *PNRD* integrated with *iPNRD* can assist an adaptive control in blocks world domain through exception identification. Thus a *PNRD/iPNRD* didactic example is presented with three block and one robotic arm. The next section presents *PNRD* and *iPNRD* fundamentals and block worlds in practice. Section 3 shows *PNRD* and *iPNRD* integration case study; followed by the robotic adaptive control. Some conclusions and further works are presented.

2 *PNRD/iPNRD* Approach

2.1 *PNRD*

Each RFID tag can be abstractly identified by a single token on *PNRD* and each reader can represent a Petri net (PN) transition. Originally, *PNRD* was developed for the identification and monitoring of passive agents, such as commercial items, parts, logistics units, and physical products. The process that an object must follow is associated with its behavioral model through Petri net data structure. Thus, *PNRD* is a formal data structure grounded in elementary Petri net which stores Petri net equation parts in RFID components: tag and reader [10].

Operationally, along the physical distributed RFID readers, tag data can represent a single marking, and this marking is automated updated during the tag data capture following the matrix equation.

It is possible to evaluate in real time whether the next calculated marking is suitable or not. In this sense, a suitable marking should respect the tag-token bi-univocal unique relationship, it means, each tagged object cannot have any negative component in its marking. Thus, the *PNRD* exception can assist Edge Computing to solve some issues locally.

A conflict in the *PNRD* approach arises when the same reader/antenna is related to more than one transition for the same tag identification and previous marking. In this case, it is necessary to apply a decision algorithm to define which transition should be chosen based on additional data or software based on specific cases [10]. This conflict can be also represented as a stochastic choice.

To exemplify the practical application of the *PNRD*, Fig. 1 presents a didactic example where a raw material is machined an pass through a test. The machined part, if approved, goes to the stock. If it is rejected it is disposed. Otherwise it

is remanufactured. Fig. 2 is the corresponded *PNRD* presenting activities of the process that holds RFID readers.

In the process in question, raw material (p_0) is sent to a workstation (p_1), and,

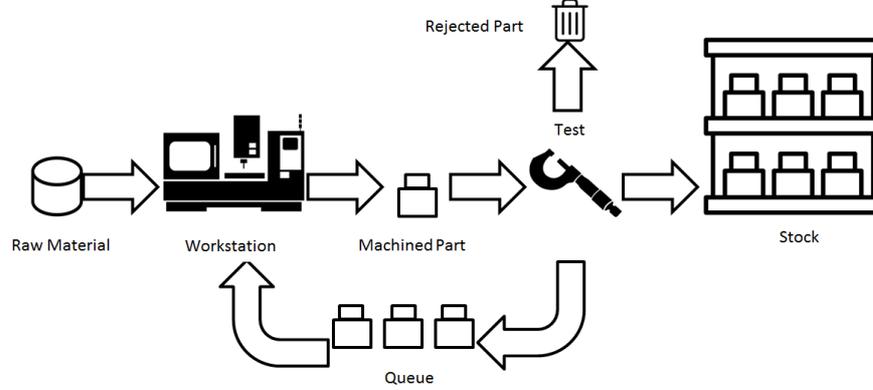


Fig. 1. Machined part test schema.

after it, the machined part p_2 pass through a specific test p_3 . If the test approves the part, it goes to the stock p_4 . If it is rejected, the piece goes for recycling p_5 . If the part does not meet the test, it goes to the rework queue p_6 , and through a new machining process and it returns to the beginning of the machined system. Although t_0 and t_6 are related to Reader1/Antenna1, both have distinct previous marking (p_1 and p_6 , respectively), what doesn't generate a conflict. The Fig. 2 correspondently adjacent matrix A is presented as following.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

PNRD exception can be shown as follow. Initially a part which is a raw with a given RFID tag. This tag is at p_0 , corresponds to the initial marking $m_0 = [1, 0, 0, 0, 0, 0, 0]^t$. Now Reader1/Antenna1 (t_0) reads this tag. Next state calculus result is stored in the tag as $m_1 = p_0 + A^t \cdot [1, 0, 0, 0, 0, 0, 0]^t = [1, 0, 0, 0, 0, 0, 0]^t + [-1, 1, 0, 0, 0, 0, 0] = [0, 1, 0, 0, 0, 0, 0]^t = p_1$. Here, Reader1/ Antenna1, does this calculation, and store the new marking in tag's memory. Next, this tag is read by Reader1/Antenna2 (transition t_1), and it used next state calculus to compute marking $m_2 = [0, 0, 1, 0, 0, 0, 0]^t = p_2$, using $m_2 = p_1 + A^t \cdot [0, 1, 0, 0, 0, 0, 0]^t$. It stores this new marking on tag. At this stage, tag at m_2 can be perceived by Reader2/Antenna2 which computes the next marking by using next state

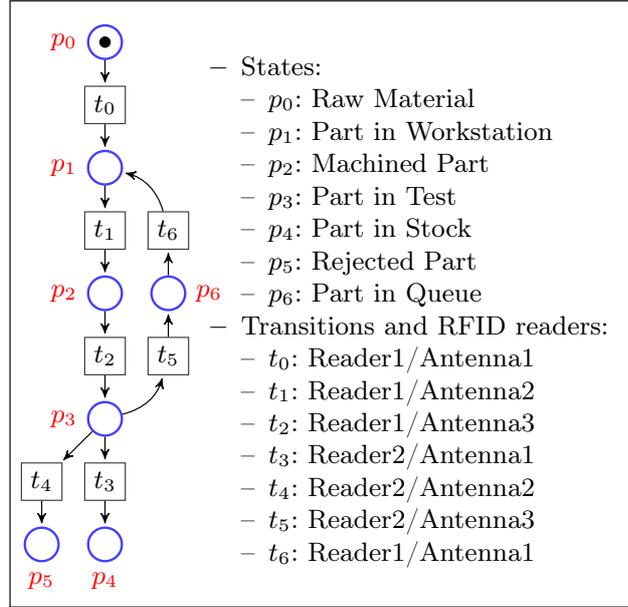


Fig. 2. *PNRD* machined part test.

calculus as follows: $m_3 = [0, 0, 1, -1, 0, 1, 0]^t = m_2 + A^t \cdot [0, 0, 0, 0, 1, 0, 0]^t$. This marking has a -1 in it, which is not possible at any reachable marking in the net. So the Reader2/Antenna2 senses that there is some anomaly in the object expected process and raises an exception.

2.2 *iPNRD*

As the active agents are normally embed with a micro controller, it becomes impracticable to associate their identification with only one RFID tag, as it is more reasonable to place an RFID reader in agents similar to mobile robots. The use of tags to identify strategic points and places of interest within the course of agents is capable of changing the behavior of these agents. In addition, the use of passive tags in remote locations to identify locations along tracks without network and electric infrastructure, and in environments with sparse power points proves to be adequate, and technically and economically feasible. Thus, to fit the *PNRD* approach in the control and monitoring of active agents, it is necessary to inverse the storage locations of the components of the Petri net equation. This approach was called inverted *PNRD* or *iPNRD* and it was presented by [7] [8]. In the *iPNRD*, the tag stores information about the firing vector u_k , while the adjacent matrix A^t and the current marking M_k are associated with the RFID Reader/Antenna.

To illustrate the *iPNRD*, a search and rescue problem, which has been simplified and dealt with on mobile robot stands, is presented in Fig. 3 here are three distinct agents or mobile robots, namely AerialBot, Groundbot and Walker.

These refer to aerial robots (a platform capable of filming the workbench), search and rescue robots, and robots emulating users, respectively.

Each agent has different inputs and outputs. In the case of AerialBot, it has access to images from a camera (view) and the exchange of TCP / IP messages. The outputs correspond to the movement of motors and the TCP/IP messages. For Groundbot the inputs refer to contact sensor, ultrasonic sensor and RFID reader with iPNRD software, in addition to receiving TCP / IP type messages. The outputs are TPCP / IP messages, motors and data recording in RFID tags arranged along the workbench. Walker has contact sensors, ultrasonic sensors and RFID reader with *iPNRD* as input, as well as motors, access to benchtop RFID tags and an emergency light to indicate any type of problem (physical failure or when it is lost). Corresponded Walker *iPNRD* is shown at Fig. 4.

The adjacency matrix A of Petri net of Figure 4 is as follows:

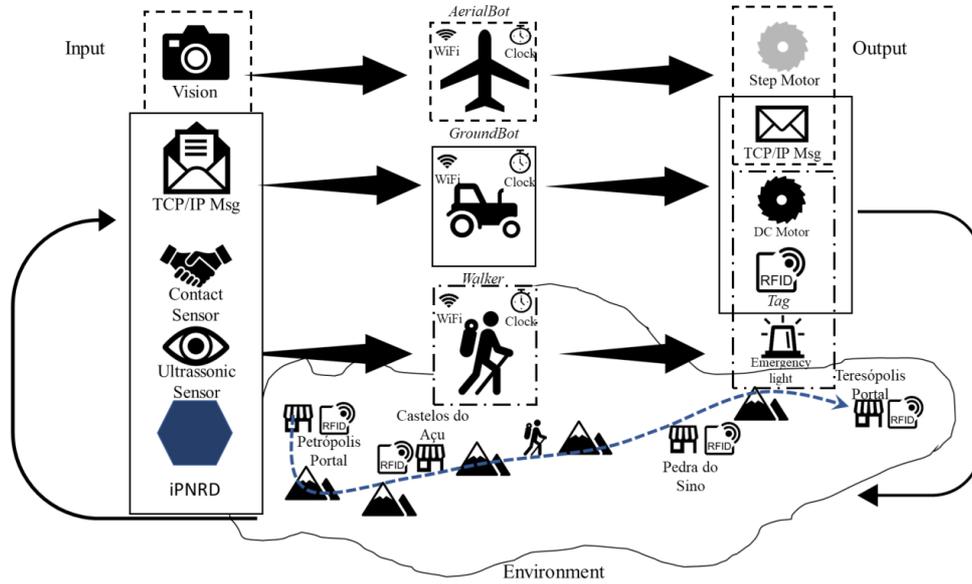


Fig. 3. Search and rescue schema.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

iPNRD data storages firing vector in tag can also hold the history composed by the agents Id, timestamps and additional information. This historical data facilitates the tracking and tracing of Walkers by the Groundbots. This information is helpful in order to directed search to the region most likely to find the victim,

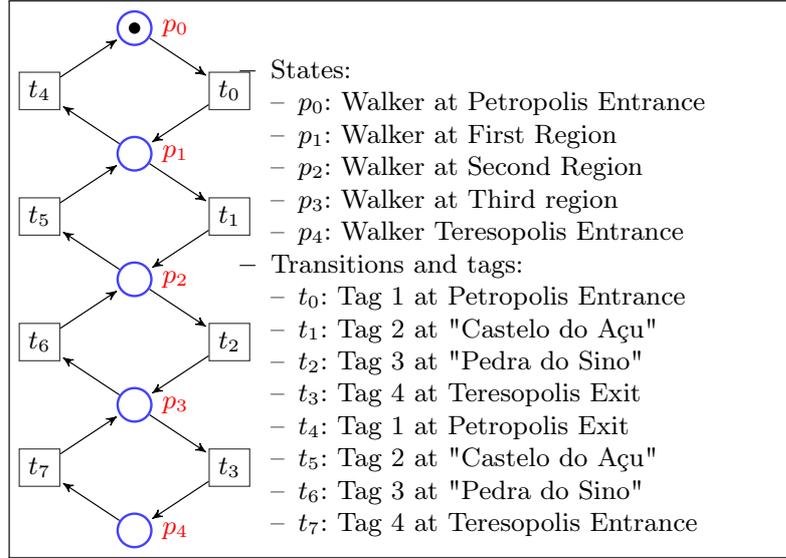


Fig. 4. Search and rescue Walker *iPNRD*.

reducing search time and, consequently, increasing the chances of rescue.

The route has 4 tags, the first one in the Petrópolis entrance/ exit, the second dividing the trail into first to second regions, the third splitting the second to third region, and the last one at the Teresópolis exit/ entrance of the trail. Note that each tag has a set of transition it is related to depending on what is the direction of the trail (from Petrópolis to Teresópolis or *vice-versa*). The Walker agent w^1 is at $w_0^1 = [1, 0, 0, 0]^t$. When Tag 1 is read, then t_0 fires, $w_1^1 = [1, 0, 0, 0]^t + [-1, 1, 0, 0]^t = [0, 1, 0, 0]^t$ which means p_1 , or First region. The Tag 1 receives additional information about the agent ID w^1 , its previous marking and the moment of registration in a database (timestamp). Other tags present similar behavior, but for different transitions. Similarly to *PNRD* exception, *iPNRD* also can identify automatically exception states, which generates an Edge Computing system.

Fig.5 summarizes how the terms of Petri net matrix equation are associated with the RFID components in both approaches.

2.3 *PNRD/iPNRD* Integration

Both *PNRD* and *iPNRD* are able to structure the communication between several agents of a system that makes use of RFID devices, however, until now only stand alone application was presented. Could their integration assist automated planning adoption in industry? To answer this question it is necessary to show how *PNRD* and *PNRD* can be integrated in an automated planning domain. In this direction, this paper presents the integration of *PNRD* and *iPNRD* related with Blocks Worlds of 3 components.

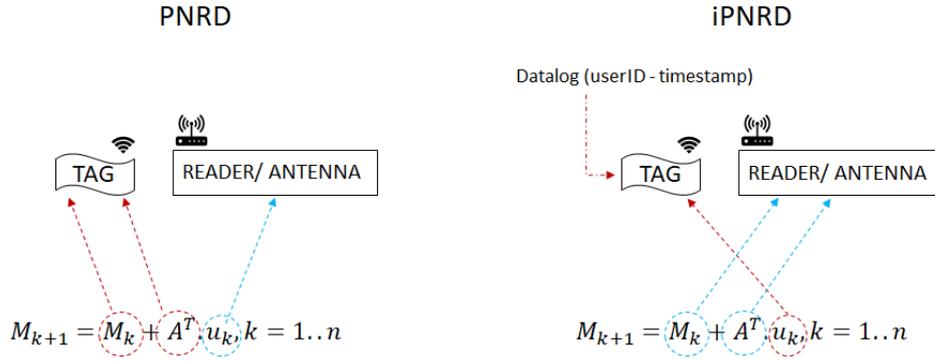


Fig. 5. Petri net matrix equation association with PNRD/iPNRD [8]

Block World in Practice The blocks world is one of the most famous planning domains in artificial intelligence [3]. The goal is to move the blocks from the initial state to the desired one. Only one block may be moved at a time, and it may either be placed on the table or placed atop another block. Because of this, any blocks that are at a given time under another block cannot be moved. A physical model is necessary to place the blocks and enable motion control, and, because of the material bounds, many logical configurations are impossible. The impossibility of some configurations is further discussed in section 3.3.

In this paper the Blocks World has a 3x3 discrete position grid with three different blocks endowed with unique identity. A notation system is defined where if a block X is over a block Y , this is represented by XY . If they are in distinct columns, they are represented by X_Y . For example AB_C means that block A is over B , and both are besides C . This notation is suitable logically, however, it generates ambiguity physically, it means, in a place with three predefined columns, AB_C can have 6 distinct physical configurations, depending on block's column positioning. In this paper, A_B_C representation (all blocks over the table) is applied for any displacement in the table (it means 3! distincts block's physical positioning). In the case of only one occupied column, the case of CBA , there are $3!/2!$ possible physical positioning. The total number of physical position is 60. To deal with the physical ambiguity, each block only must be placed on the table in a predetermined position, what reduces the total number of physical position to 13.

Blocks world planning has been shown to be a difficult problem for finding an optimal plan, as it is NP-hard in the worst case [3]. Finding an optimal route means to change the blocks state to the objective with minimal cost, where cost is generally the total summation of all transitions cost. In this work all transitions have unitary cost.

The Blocks World restrictions, actions and possible configurations can be modeled through different paradigms, such as propositional logic, full state space representation or tree expansion. As this domain is didactic in order to present physical informational integration using *PNRD/iPNRD*, these models have full

state representation due to the small quantity of elements, thus the optimal solution can be found using simple graph search algorithms, whose time complexity order is polynomial.

This means that planning for sorting execution is simply about searching through different PNs to generate the robot driving sequence. The search algorithm of choice is the BFS (breadth-first search), that will expand all places following a FIFO (First In First Out) list, this algorithm being optimal for uniform cost arcs [5]. The time and space complexity is $O(TP)$, where T is the total number of transitions and P the total places, because the PN is represented as a modified adjacency graph and BFS usage.

Using a heuristic search might give significantly better results for bigger state spaces. If a different paradigm was used, such as tree expansion, the BFS has completeness assurance, but this paradigm is better suited to deal with bigger state spaces, meaning that a heuristic search would outperform the non-informed BFS.

According to [2], in the physical domain there are some distinct dynamic treatments for fixed robots, like inverse kinematics [4] for pre-established spatial coordinates, which is very useful when dealing with very well managed object disposition, but it requires higher environment control in order to work well. In more flexible and robust applications it might require the use of sensors, planners and controllers for adaptive action, like vision systems and machine learning methods, that can be computationally expensive and require training data.

As this work deals with predefined discrete points similar to a Petri space [4] for the robot to move and act, then the course of action can be given by a planner based on PN matrix search algorithm. This method is simpler to implement and has inferior computational cost when compared to a machine learning high level sorter, but is more adjustable than a pick-and-place methodology that does not include superior level instruction planning.

Sorting can be very challenging depending on the level of automation [2], so constraints might be necessary to reduce the need for more sophisticated tools. As autonomous sorting is enabled by sensors and actuators, this system dynamics is constrained by the use of a single sensory system. The RFID components are the unique sensory system in this work which activates actuation in the robotic arm. Because the possible positions are modeled in Petri Space there is no need for a more complex motion planner, as the whole position state space is represented, where the transitions represent the modeled movements and the places position, again a graph search algorithm can be used to find the optimal logical path given the model restrictions. A robot can perform tasks at any point in its Petri space, so it is possible to flexibilize the discrete positions into a less constrained method in a case which requisites of using more complex motion planners and sensor systems. Fig. 6 represent the Petri state of 3 blocks. A logical solution to avoid collisions is adding a line atop the original one (specifically places $C1_4$, $C2_4$ and $C3_4$), so that the robot may move superior to the blocks when changing columns. Thus this Petri space represents the robot's physical-movement actions through its transitions between predefined positions

(places). For didactic reason this is a simplified Petri Space which could be modeled as a state machine, however, real applications requires a more complex Petri space.

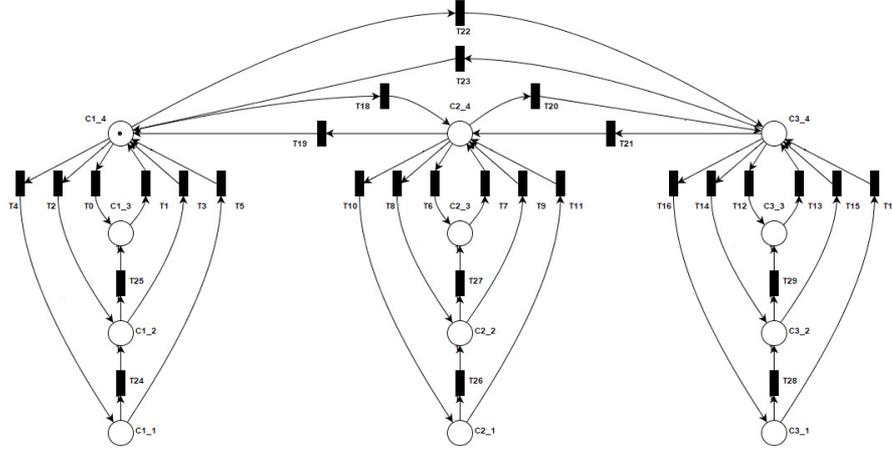


Fig. 6. Three blocks physical discrete positioning viewed as Petri Space

3 Case Study: PNRD/iPNRD for Three Blocks

3.1 Block PNRD model

The block *PNRD* models the relationship of itself with the environment. Depending on arm physical movement, it is possible to identify what block is under another (with a movement from the top to the bottom of a column), or, otherwise, what block is over another (with a movement from the bottom to the top), what is implemented in this work. So, their *PNRD* model represents which element the referred block stays atop. As an example, block A *PNRD* model can have three distinct states, A over B (*AB*), A over C (*AC*), or A on the table (*AT*) and it is presented in Fig. 7. The *PNRD* topology is composed of 3 places and 6 transitions for each one of the three blocks. An additional information is demanded, each block must inform its expected final state.

3.2 Robot iPNRD model

The *iPNRD* represents the active agent perspective on the world, and as there is only one robot arm, its *iPNRD* model also represents the whole Blocks World disposition state space. For three blocks, it is possible to represent the logical macro state as shown in Fig. 8, that is the blocks relative positions with 13 distinct states and 30 transitions. As informed in subsection 2.3, there is no logical

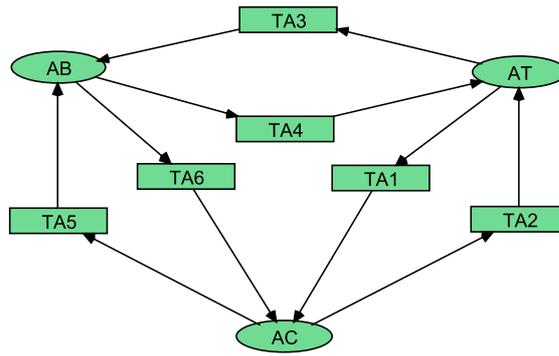


Fig. 7. Block A *PNRD* model.

distinction between A_B_C and C_A_B , but only the reference physical dispositions shown in Fig. 8 are treated in this paper. Each transition refers to a movement of a block, for instance, $miX2Y$ means movement number i of block X to position Y (on table T or another block). Place identification is following the notation described at subsection 2.2. Through this *iPNRD* model it is

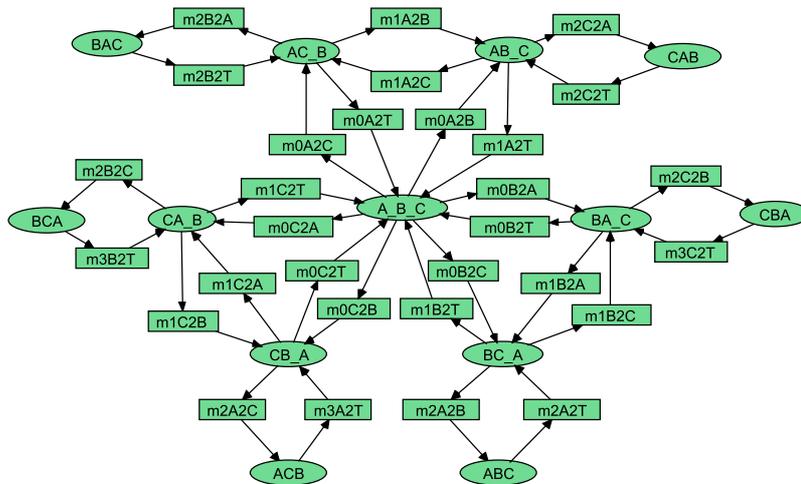


Fig. 8. Robot arm *iPNRD* model.

possible to navigate from a state to another, identifying the necessary intermediate states and transitions. Searching in the *iPNRD* model is similar to a macro state planning, or a discrete state control, which can reach the final state from the initial by firing a correct sequence of transitions. This planning is of higher level because it does not relate directly to the robot kinematic sequence. The

movement sequence is generated by searching on the physical level, the Petri Space.

3.3 *PNRD/iPNRD* integration with Petri Space

All blocks PNRD representation must be coherent with the arm *iPNRD* model and with the Petri Space from the physical system. Places and transitions are related to the other models directly and these relationships can be expressed through tables. It is mandatory to inspect whether the physical arrangement is equivalent to the *PNRD* and *iPNRD* representation regarding the initial state. As this case study has a simple model and the whole state space is modeled in a PN, the optimal solution can be found using graph search algorithms. For a more complex model this solution requires a different approach, as an example automated planner integration with *iPNRD* model.

The proposed method to identify the position of each block on the physical domain is by using the RFID reader placed in the robot grapppler. As all the blocks have RFID tags, which gives them an individual unique identity, the reconnaissance of each block and its corresponded position is feasible during a sweep operation.

All blocks identification from *PNRD* habilitates the identification of *iPNRD* initial and final state. This means that the blocks current position is identified by tag reading, and this information can be confronted with block's own state from its *PNRD*; and the final state by reading in each block additional data. Summarizing, Petri state (blocks position perceived by robot), and *PNRD* data are mapped to *iPNRD* state as presented by Table 1, after a robot sweep operation.

Table 1. Petri Space, *PNRD* and *iPNRD* State relationship

Petri Space Block A	Petri Space Block B	Petri Space Block C	<i>PNRD</i> A	<i>PNRD</i> B	<i>PNRD</i> C	<i>iPNRD</i> State
<i>C1_1</i>	<i>C2_1</i>	<i>C3_1</i>	<i>AT</i>	<i>BT</i>	<i>CT</i>	<i>A_B_C</i>
<i>C1_1</i>	<i>C1_2</i>	<i>C3_1</i>	<i>AT</i>	<i>BA</i>	<i>CT</i>	<i>BA_C</i>
<i>C1_1</i>	<i>C2_1</i>	<i>C1_2</i>	<i>AT</i>	<i>BT</i>	<i>CA</i>	<i>CA_B</i>
<i>C1_1</i>	<i>C1_2</i>	<i>C1_3</i>	<i>AT</i>	<i>BA</i>	<i>CB</i>	<i>CBA</i>
<i>C1_1</i>	<i>C1_3</i>	<i>C1_2</i>	<i>AT</i>	<i>BC</i>	<i>CA</i>	<i>BCA</i>
<i>C2_2</i>	<i>C2_1</i>	<i>C3_1</i>	<i>AB</i>	<i>BT</i>	<i>CT</i>	<i>AB_C</i>
<i>C1_1</i>	<i>C2_1</i>	<i>C2_2</i>	<i>AT</i>	<i>BT</i>	<i>CB</i>	<i>CB_A</i>
<i>C2_2</i>	<i>C2_1</i>	<i>C2_3</i>	<i>AB</i>	<i>BT</i>	<i>CA</i>	<i>CAB</i>
<i>C2_3</i>	<i>C2_1</i>	<i>C2_2</i>	<i>AC</i>	<i>BT</i>	<i>CB</i>	<i>ACB</i>
<i>C3_2</i>	<i>C2_1</i>	<i>C3_1</i>	<i>AC</i>	<i>BT</i>	<i>CT</i>	<i>AC_B</i>
<i>C1_1</i>	<i>C3_2</i>	<i>C3_1</i>	<i>AT</i>	<i>BC</i>	<i>CT</i>	<i>BC_A</i>
<i>C3_2</i>	<i>C3_3</i>	<i>C3_1</i>	<i>AC</i>	<i>BA</i>	<i>CT</i>	<i>BAC</i>
<i>C3_3</i>	<i>C3_2</i>	<i>C3_1</i>	<i>AB</i>	<i>BC</i>	<i>CT</i>	<i>ABC</i>

Exception can occur in two ways. In the *PNRD* level if and only if the Petri state set is distinct from *PNRD* state set, this means some block(s) was moved from their initial position, as Petri state set has priority over *PNRD* state set. Consequently the *PNRD* states must be updated following the physical disposing. Constraints in this exception arise from the restrictions of the physical world and the proposed Blocks World restrictions. There are thirteen different logical dispositions in the *iPNRD* for three distinct *PNRD* places for each block. As the mapping is a bijective function with three inputs, having three possible values each, most permutations are invalid having no real transfer to an *iPNRD* state. The same occurs on the physical layer to *iPNRD* or *PNRD*, mathematically many configurations exist, but the physical model constraints the possible configurations.

Another exception can be raised from *PNRD* desirable state set. There is 13 *iPNRD* states with specific *PNRD* state set, for instance, $A_B_C = AT, BT, CT$. If the desirable final state set does not belongs to Table 1 *PNRD* set, it means that the final state set is unreachable and inconsistency, as example AB, BA, CA , where A is over B , B is over A and C is over A , what is physically unfeasible. This mapping is bijective, so it admits an inverse. Transitions in *iPNRD* convert to a sequence of Petri State transition.

A movement command (transition in the Petri State) is converted into a command signal for the robot driver system. This relation is also bijective, however, for the physical robot, the mapping between the path and the static position has an $n : 1$ relationship, meaning that multiple commands can reach the final position. This paper deals with this issue storing in advance each movement sequence and relating them with a correspondent Petri state transition.

3.4 Block World solution deployment

The first process to solve the Block world consists on recognizing the initial and objective states. Then a sorting engine based on the *iPNRD* model uses the robotic arm to act, following the Petri state transition movement.

The sorting engine is a multi-process composed of solving for the optimal action plan given the original and final global states, respecting the model constraints. This process consists of searching in the *iPNRD* model for the least amount of transitions. Then it is necessary to map the physical level, in which each transition has an initial and final position. The next step is to search for the least amount of robot movements in the physical model. Afterwards it is necessary to concatenate the actions and to translate them into robot driver command, moving the blocks in the physical world. This multi-process sorting is described in the next section.

After the definition of *iPNRD* objective and its correspondent *iPNRD* transitions, it is necessary to generate Petri state sub-goals, meaning, the required initial and final physical place that each involved block must be moved in order to complete a *iPNRD* transition. Each transition in the *iPNRD* maps to an unique sequence of Petri Space transitions, as is in Fig. 9. These movements must be concatenated in order to reach the desirable final set, so it is necessary

to connect the final place of a sub-goal to the initial in the consequent one. The final step is to convert a Petri space transition to a robot command. The simplest way to perform this task is composed of mapping these transitions into a more easily readable order, converting the concatenated transitions into an driving order string, then it is processed by the robot driver. The robot driver converts the string orders into joint angles and grappler movement by commanding the servomotors.

iPNRD Transition	Petri space Transition set	iPNRD Transition	Petri space Transition set	iPNRD Transition	Petri space Transition set
<i>m0A2C</i>	<i>T5+T22+T14</i>	<i>m0C2B</i>	<i>T17+T21+T8</i>	<i>m2C2A</i>	<i>T17+T21+T6</i>
<i>m0A2T</i>	<i>T15+T23+T4</i>	<i>m1C2T</i>	<i>T9+T20+T16</i>	<i>m2C2T</i>	<i>T7+T20+T16</i>
<i>m0A2B</i>	<i>T5+T18+T8</i>	<i>m1A2C</i>	<i>T9+T20+T14</i>	<i>m2C2B</i>	<i>T17+T23+T0</i>
<i>m1A2T</i>	<i>T9+T19+T4</i>	<i>m1A2B</i>	<i>T15+T21+T8</i>	<i>m3C2T</i>	<i>T1+T22+T16</i>
<i>m0B2A</i>	<i>T11+T19+T2</i>	<i>m1B2A</i>	<i>T15+T23+T2</i>	<i>m2A2B</i>	<i>T5+T22+T12</i>
<i>m0B2T</i>	<i>T3+T18+T10</i>	<i>m1B2C</i>	<i>T3+T22+T14</i>	<i>m2A2T</i>	<i>T13+T23+T4</i>
<i>m0B2C</i>	<i>T11+T20+T14</i>	<i>m1C2B</i>	<i>T3+T18+T8</i>	<i>m2A2C</i>	<i>T5+T18+T6</i>
<i>m1B2T</i>	<i>T15+T21+T10</i>	<i>m1C2A</i>	<i>T9+T19+T2</i>	<i>m3A2T</i>	<i>T7+T19+T4</i>
<i>m0C2A</i>	<i>T17+T22+T2</i>	<i>m2B2A</i>	<i>T11+T20+T12</i>	<i>m2B2C</i>	<i>T11+T19+T0</i>
<i>m0C2T</i>	<i>T3+T22+T16</i>	<i>m2B2T</i>	<i>T13+T21+T10</i>	<i>m3B2T</i>	<i>T1+T18+T10</i>

Fig. 9. *iPNRD* and Petri State Relationship.

4 Robotic Adaptive Control Based on *PNRD* and *iPNRD* Integration for Three Blocks

The implemented solution involves a PC, an Arduino mega microcontroller, RFID components, robotic arm and the three tagged blocks. This interfacing can be done by a range of methods, like serial protocols or intranet. In this paper the means for transmission are the RFID sensor for interfacing the blocks to the microcontroller and serial bus to communicate the microcontroller to the computer. For motor command the microcontroller sends processed signals to move the robot. The system could be adapted to be processed in a monolithic manner or to be further distributed among other agents.

The main idea of the adaptive control is presented in Fig. 10, which shows an *PNRD/iPNRD* feedback depending on whether there is exception in the process. The sorting process finishes when all steps have been fulfilled correctly. This system can operate independent of supervision, unless an exception that disables the process achievability happens. An example of disabling occurrence is sensorial malfunctioning.

The adaptive control flow starts with an external command to start the operation, such as a supervisor sending a job request.

The system performs a sweep to locate the blocks using the arm and mounted

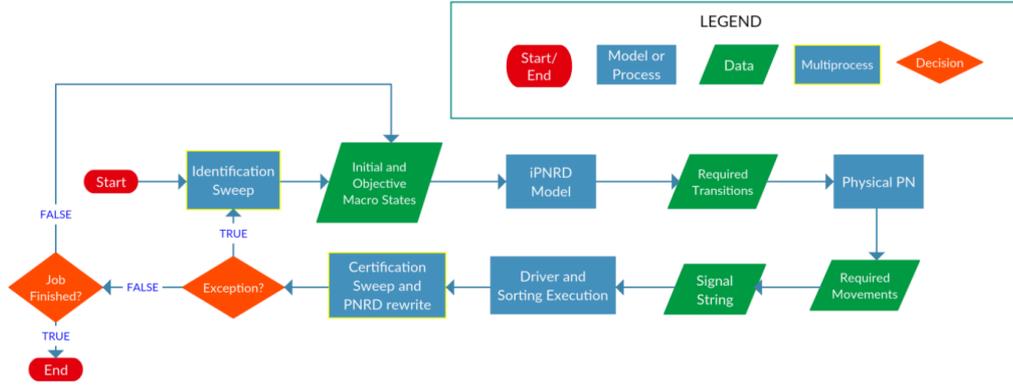


Fig. 10. Adaptive Sorting Process with *PNRD/iPNRD* feedback.

RFID reader. This sweep starts at $C1_4$ Petri state and always follows a fixed transition sequence $\{T4 \rightarrow T24 \rightarrow T25 \rightarrow T1 \rightarrow T18 \rightarrow T10 \rightarrow T26 \rightarrow T27 \rightarrow T7 \text{ to } T20 \rightarrow T16 \rightarrow T28 \rightarrow T29 \rightarrow T13 \rightarrow T23\}$. The adaptive control algorithm performs the physical inspection reading each discrete position in a column from below to the top, as informed in section 3.1. When the first column is done the next occupied position will be the highest in the second column, then the robot will descend to the lowest point and will perform the scan upward. When all three columns are done the system will process the *PNRD* data.

Then, through the use of Table 1, initial and objective *iPNRD* states (macro states) are generated by *PNRD* and Petri space information. A BFS search is performed to find the optimal route in the *iPNRD* model. In each *iPNRD* transition a sub-goal search is done to find the set of movements in the Petri space transitions. The movements are concatenated and converted into a string to be fed to the microcontroller robot arm driver. The robotic arm motors respond and the system actuates the blocks until the final command.

A simplified sweep is done to verify if the current state is equal to the desired one through Petri state position, *PNRD* updates the block next state, as *iPNRD* changes the system state. In this step, it is verified if the final movement was possible to be executed. If not, it means that another block was moved without robotic assistance to that position, and it raises another exception type. The system verifies whether the process was finished. If not, it goes to the next sub-goal.

This adaptive control works by searching in the *iPNRD* model, called macro states, to generate a global plan. Each *iPNRD* transition maps to an initial and final state in the Petri Space for movement, defining a sequence of sub-goal action.

In order to concatenate the generated movements and act correctly, some special cares are necessary. The first step is to have a reference position to begin and end the actuation. The next required step is to concatenate the final state to the initial state for the different sub-goal. This is necessary because the robot

needs to materially move the block to perform the tasks. It is also fundamental to open and close the robot grappler at the right moments, so that instruction must be included in the command string.

The feedback of final condition is necessary to perceive intermediate exceptions. There can be many different types of exceptions through the process of solving the system, assuming every model and table was created correctly. Exceptions in the system identification might mean that a block is missing or an RFID element is malfunctioning. This return permits that the system self regulates and accuse anomalies. It is fundamental to emphasize that the system is partially observable, meaning that some external actions and singularities cannot be perceived immediately, sometimes only after a sorting requisition is finished, other during the beginning of another job. One methodology to analyze irregularities, their causes and origin, is viable through the circumstances of exception. This can be accomplished using methods that involve failure mode analysis and statistics to infer about possible faults.

In the case that the process is successfully finished it is necessary to update *PNRD* data. This is done by moving into the blocks position and sending new data to be stored in the tag according to the job last performed.

In order to actuate correctly the system needs geometrical references so that the arm and blocks can be positioned right. These markings are absolutely necessary if the system has a pick-and-place approach, doing work without trajectory feedback. In a model that includes more sensors and auxiliary systems for dynamics the layout can be more flexible.

Other very important aspects are the mechanical and electronic projects. The system has to be designed to handle the mechanical load and operate in the specified work space. In the block world explored here, it was simple to build an adequate robot for the task, so it was not necessary to make studies and formulate deeper about dimensions, stress, controls, power and their material tools and needs. A small arm was projected and built using additive manufacturing and common servomotors, and a simple microcontroller is used to operate the arm and manage data coming from the RFID reader and computer. In a more demanding situation studying those requirements is fundamental to use a suitable robot and components.

4.1 Adaptive control example

This example has *BA_C* initial state (Fig. 11) and *BAC* as final state (Fig. 12). First, the system must be booted by a supervisor, then it will proceed to sweep through the discrete positions identifying the blocks by their unique ids and get their desired *PNRD* state. The robot will execute this process as described in section 4.1. The initial identified Petri Space is $\{C1_1, C1_2, C3_1\}$. Through Table 1 it is possible to find the current *iPNRD* state as *BA_C*. The current *PNRD* state can be checked to conclude about changes in the expected state. The data about the blocks desired position is previously added in the *PNRD*, and it is collected during the initial sweep. Thus desired *PNRD* set is $\{AC, BA, CT\}$, Table 1 correlates this with *BAC iPNRD* state.

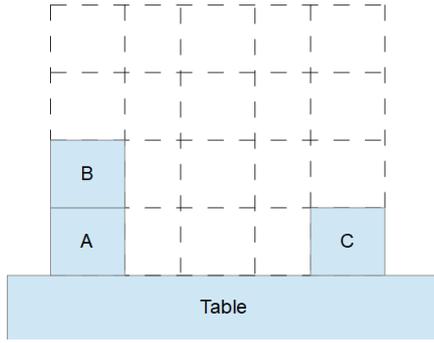


Fig. 11. Initial state representation.

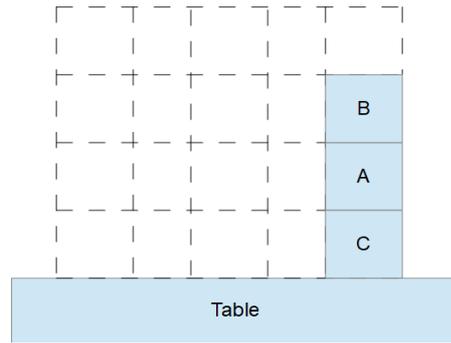


Fig. 12. Final state representation.

The next step is to feed the initial and objective macro states to the sorting engine as $\{BA_C, BAC\}$, in order to determine the course of action in the *iPNRD* state space.

The result of a BFS search algorithm (planner) is the subobjective progression and transition firing sequence. In this case, the best path to reach the objective state is, in order, through A_B_C , AC_B and finally to BAC . These macrotransitions must relate to Petri State transitions in order to actuate on the material world.

To move, the robot must leave a reference initial position, defined as $C2_4$, to the initial position of the first subobjective with the gripper open. The first subobjective is to change from *iPNRD* state BA_C to A_B_C , which translates into moving block B from $C1_2$ to $C2_1$. To move from $C2_4$ to $C1_2$ Petri Space transitions $T19$ and $T2$ must be fired in sequence. The next step is to grip the block. This action is provided in the database, because logically the block has to be caught and moved. In sequence, transitions $T3$, $T18$ and $T10$ must occur. Reaching its final position the arm can open. This ends the movement sequence to reach the first subobjective.

Now, to start the second subobjective, it is necessary to move from the current position to the next initial position. The next block that will be moved is block A . The planner identifies that it must concatenate the movement sequence and the subobjective now is to move from $C2_1$ to $C1_1$ with an open arm. Then another search is performed to change the *iPNRD* from A_B_C to AC_B . To perform this shift the planner must move block A from $C1_1$ to $C3_2$. In this example the last subobjective consists of moving AC_B to BAC . When sorting is complete the gripper must return to $C2_4$, ending the adaptive control.

The robot driver just interprets the received string and executes it. For each Petri Space transition there is a corresponding robot configuration. To reach the desired position each servomotor must be in a specified angle. If the position $C2_2$ is requested, a robot with 6 controlled joints (motors) must command each one to be in a pre specified angle. This can be represented as the sextuple where is the desired servomotor angle, as an example $\{90, 120, 60, 40, 170, 20\}$ in

degrees. Final steps are a certification sweep and *PNRD* rewrite. If the resulting *iPNRD* macro state is equal to the desired, the goal was performed successfully, else an exception occurred. If successful, each blocks *PNRD* must be updated by rewriting its tag's data.

5 Conclusions and Further Works

This paper presented how PNRD and iPNRD integration can be applied in an adaptive control. It also shows that it is feasible to integrate this solution with automated planners. However, for more complex system (as example 10 blocks) iPNRD model is not able to represent all blocks state space, and it must be simplified based on automated planning results. In each operation it is indispensable to rectify the model. In special on the physical level, the adequate fit for positions and trajectory, with uncertainty and precision levels inside a certain tolerance are essential. To build the models correctly and to realize the system it is fundamental to, in each step, certify that what was built is accurate and done right. But when dealing with automatic model generators and bigger systems this problem gets harder, and it must be investigated.

Partial observability is another issue, because it makes exception detection much harder and limits the sweeps average speed. By including more sensors partial observability issues could be greatly reduced. Building auxiliary systems that make exceptions harder to happen or faster to detect could increase system performance.

There are dynamical and geometrical issues regarding the blocks disposition and manipulation. In order to successfully realize the actuation and sensing the system must be precise or robust, meaning that motors torque, potency and control, vibration analysis, sensor signal studies and physical references to assure the fidelity of the driver system motion might be required. Using any form of position sensors to fully monitor the grid would make the moving processes faster. Adding other sensors the data could be compared to whatever additional information lies inside the tags. Vision systems could make the methods more flexible and be auxiliary in the movement planner. A more profound study and building improved hardware could make the system faster and more reliable.

The RFID system generates a complementary data structure locally, and, in case of network issues, it could assist robot movement independently, acting as Edge Computing application. This aspect must be studied straight forward.

Other means for improvement are making the machines and objectives more flexible and having multiple agents. The system could support many different objective states or perform partial satisfaction on conflicting jobs. The system could work on parallel or concurrent robots that share a common work space, having distributed tasks and logic but keeping consistent. To work based on a reference clock would be great for timing activities. These changes could be further improved by using model paradigms better suited for state space expansion but it would require more computational power and the use of more sophisticated techniques. A very important addition to the system is FMEA (Failure

Mode and Effect Analysis) for determining exception cause. A more powerful sensory system coupled with tools for inference and statistical analysis could provide useful data for concluding about the original problems.

PNRD/iPNRD must be formally defined for concepts, operations and analysis.

6 Acknowledgement

This work is supported by CNPq, CAPES, FAPEMIG, UFU. This paper is dedicated to prof. Dr. José Reinaldo Silva (PMR/USP).

References

1. Ghallab, M., Nau, D.S., Traverso, P.: The actor's view of automated planning and acting: A position paper. *Artificial Intelligence* **208**, 1–17 (2014). <https://doi.org/https://doi.org/10.1016/j.artint.2013.11.002>, <http://www.sciencedirect.com/science/article/pii/S0004370213001173>
2. Guerin, J., Stephane, T., Nyiri, E., Gibaru, O.: Unsupervised robotic sorting: Towards autonomous decision making robots. *International Journal of Artificial Intelligence and Applications (IJAIA)* **9**(2), 81–98 (03 2018). <https://doi.org/10.5121/ijaia.2018.9207>
3. Gupta, N., Nau, D.S.: On the complexity of blocks-world planning. *Artificial Intelligence* **56**(2), 223–254 (1992). [https://doi.org/https://doi.org/10.1016/0004-3702\(92\)90028-V](https://doi.org/https://doi.org/10.1016/0004-3702(92)90028-V), <http://www.sciencedirect.com/science/article/pii/000437029290028V>
4. Murray, R.M., Sastry, S.S., Zexiang, L.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edn. (1994)
5. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edn. (2009)
6. Sette, F.M., Vaquero, T.S., Park, S.W., Silva, J.R.: Are automated planners up to solve real problems? *IFAC Proceedings Volumes* **41**(2), 15817–15824 (2008). <https://doi.org/https://doi.org/10.3182/20080706-5-KR-1001.02674>, <http://www.sciencedirect.com/science/article/pii/S1474667016415387>, 17th IFAC World Congress
7. da Silva, C.E.A., de Souza Tavares, J.J.P.Z., Ferreira, M.V.M.: Arduino library developed for petri net inserted into rfid database and variants. In: Khomenko, V., Roux, O.H. (eds.) *Application and Theory of Petri Nets and Concurrency*. pp. 396–405. Springer International Publishing, Cham (2018)
8. da Silva Fonseca, J.P.: High-level Petri nets and inverted PNRD associated to mobile robot control: an approach to search and rescue on trails and crossings. Ph.D. thesis, Universidade Federal de Uberlândia (2018)
9. da Silva Fonseca, J.P., de Sousa, A.R., Ferreira, M.V.M., de Souza Tavares, J.J.P.Z.: Planpas: Plc and automated planning integration. *International Journal of Computer Integrated Manufacturing* **29**(11), 1200–1217 (2016). <https://doi.org/10.1080/0951192X.2015.1067909>, <https://doi.org/10.1080/0951192X.2015.1067909>
10. Tavares, J.J.P.Z.D.S., Saraiva, T.A.: Elementary petri net inside rfid distributed database (pnrd). *International Journal of Production Research* **48**(9), 2563–2582 (2010). <https://doi.org/10.1080/00207540903564934>, <https://doi.org/10.1080/00207540903564934>