Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-28 (2022) 73–78

# Parallel Motion Execution and Path Rerouting for a Team of Mobile Robots [★]

**Sofia Hustiu** [*,**] **Cristian Mahulea** [*] **Marius Kloetzer** [**]

[*] *Aragón Institute of Engineering Research (I3A), University of Zaragoza, Spain, (e-mail: cmahulea@unizar.es)*
[**] *Dept. of Automatic Control and Applied Informatics, Technical University "Gheorghe Asachi" of Iasi, Romania, (e-mail: sofia.hustiu@academic.tuiasi.ro, marius.kloetzer@academic.tuiasi.ro).*

**Abstract:** This work achieves parallel movements and collision free paths for a team of identical robots evolving in a known environment while satisfying a global Boolean-based formula over a set of regions of interest. The movement capabilities of the robots within the grid-based environment are modeled as a Petri net system. The solution starts from initially known robot paths given by the approach in (Mahulea et al., 2020b). The main advantages of this work are two-folded: (i) it reduces the number of waiting states of the robots throughout their paths, (ii) it considers path rerouting action solved by a MILP problem, without generating new observations along their paths. The rerouting is triggered when the number of robots that are waiting is greater than a threshold, or when some stopped robots inhibit the movement of others. Thus, the proposed method yields parallel movement of robots and path rerouting when needed. The algorithm is integrated in the open-source toolbox RMTool (González et al., 2015).

*Keywords:* Petri net, Boolean-based specification, Multi-robot system

## 1. INTRODUCTION

In the field of mobile planning, a large part of interest is directed towards ensuring collision free movements for a team of robots, whose members are generally denoted as Automated Guided Vehicles (AGV) (Reveliotis, 2000). This topic is also denoted as multi-agent path finding (MAPF) problem. Depending on the domain, a given mission needs to be satisfied, e.g., package delivery, rescue operation etc. The formalism used for mission varies from Boolean-based specification (Mahulea et al., 2020b) to high-level specification such as LTL (Schillinger et al., 2018; Lacerda and Lima, 2019; Hustiu et al., 2021). These approaches are included in symbolic planning for robot motion (Belta et al., 2007). An essential part in this field is expressed by the model used for the environment and for the team of robots, such as: global team automaton (Ding et al., 2011), local automaton assigned to each robot (Schillinger et al., 2018), Petri net (PN) system (Mahulea et al., 2021) or graph representation (Hönig et al., 2018).

Depending on the given mission for multi-robot systems, the motion of robots can be coordinated (Mahulea et al., 2020b), or it can be individual, based on independent tasks (Yu et al., 2021). Our previous work (Mahulea et al., 2020b) captures the first aspect, by providing movement plans for a team of robots to ensure a global Boolean-based specification denoted $\varphi$. The motion capabilities of the team are captured by a Petri net system called Robot Motion Petri net (RMPN). The algorithm divides the solution in two steps, as the specification $\varphi$ imposes Boolean requirements along robot paths and at their final destinations. The implementation dwells in solving two Mixed Linear Integer Programming (MILP) problems, one for each step. Although the paths are collision free and the global mission is fulfilled, the movement of the robots is quite conservative, i.e., the robots reach their final destination sequentially in scenarios where the paths share a common cell. Therefore, it is compulsory to improve this strategy through parallel motion, such that the time to fulfill the mission is reduced.

Moreover, the robot motions should be collision- and deadlock- free and for this reason, the problem can be included in class of Resource Allocation Systems (RAS). One solution that prevails is based on Banker's Algorithm (BA), which was originally designed to avoid deadlocks. In short, this algorithm simulates the maximum resource allocation for one process at each step, before taking a decision in regards to the actual sharing of resources. Thus, the processes finish one at a time, but an iterative implementation can be done by considering only one operation/movement in a process per step. It is required to know beforehand the number of resources in the system, and the system needs to be in a *safe state* at each step (meaning that all processes can finish in a finite time) (Lawley et al., 1998). Although BA has several benefits for deadlock avoidance, only a few works use this method for mobile planning a team of robots, considering each process as the path of one robot. The papers (Kalinovcic et al., 2011; Bobanac and Bogdan, 2008) propose improvements of Banker's Algorithm, based on graph representation, by allowing the robots to be in *unsafe states* for a reduced

[★] This work has been partially supported by CICYT-FEDER PID2021-125514NB-I00 and by UNIZAR UZ2021-TEC-02 projects in Spain and by PN-III-P2-2.1-PTE-2019-0731 project in Romania.

time and only in some particular conditions to reduce the redundant waiting time of the robots. A different approach is captured in (Song et al., 2021) considering dynamical release of resources throughout the running processes in the system. The advantages of a PN system while the BA is implemented are included in (Tricas et al., 2000), emphasized in flexible manufacturing systems.

The current work overcomes the conservatism of our previous approach (Mahulea et al., 2020b), by returning high-level path planning with parallel movement of the robots. In this sense, we intend to maximize the number of robots which move in one global *step*, based on their known paths. The main idea of the overall solution is to iteratively check what robots can safely advance in their next position. The robots must ensure a global Boolean-based specification, while exploiting the advantages of a PN model in a partitioned environment. An important novelty of our approach lies in considering path rerouting action without generating new observations along the paths, action triggered by the number of robots waiting in the same step. In addition, the rerouting of paths is also triggered when a blocking along the paths may occur due to the parallel movement and waiting of the robots. For example, this situation appears when some robots reach their final states, but block the path of other robots. The rerouting is made for the entire team, by solving one MILP problem which has as inputs the current and the final markings of the PN system. This will shorten the time (counted in global *steps*) necessary to fulfill the mission.

## 2. PRELIMINARIES

**Environment and motion planning.** Let us consider a known 2D rectangular environment, denoted $E$, in which a team $R = \{r_1, r_2, \ldots, r_{|R|}\}$ of identical and omnidirectional mobile robots evolves. The notation used for the number of robots throughout the paper is denoted as: $r = |R|$. The environment includes several disjoint regions of interest (ROIs) denoted $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$. To ease the movement of the robots based on the model of the environment, $E$ is partitioned into regions (or cells). For simplicity of exposition, we here assume a grid decomposition of the environment, where each ROI is equal with one or with a union of multiple cells. The set of cells is denoted by $P = \{p_1, p_2, \ldots, p_{|P|}\}$. The classification of cells is given by the function $h : P \to Y \cup \emptyset$, with $h(p_i) = \emptyset$ for the cells which do not belong to any ROI, and $h(p_i) = y_j$ for the cells which are included in the $y_j$ region. Since regions in $Y$ are disjoint, each cell can correlate to maximum one ROI.

*Example 1.* Let us consider the environment $E$ from Fig. 1. $E$ is divided into 6 identical cells, with 3 ROIs $y_1, y_2, y_3$. Two robots $r_1, r_2$ are initially located in $p_1, p_2$. For example, the cell $p_6$ corresponds to the third ROI, therefore it is labeled as: $h(p_6) = y_3$. On the other hand, cell $p_3$ does not belong to any region, hence $h(p_3) = \emptyset$. ■

*Definition 2.* A Robot Motion Petri Net system (RMPN) model is represented by the 4-tuple $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, Y, h \rangle$ (Mahulea et al., 2020a), where:

- $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ represents a Petri net model.
- $\boldsymbol{m}_0$ represents the initial marking, where $\boldsymbol{m}_0[p]$ captures the initial number of tokens in place $p$, $\forall p \in P$ (in this model, one token corresponds to one robot).

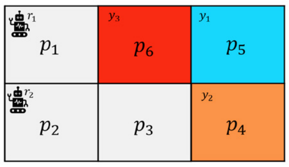

Fig. 1. Environment with 2 robots and 3 regions of interest

- $Y \cup \emptyset$ represents the set of output symbols, where $\emptyset$ denotes the free space.
- $h : P \to Y \cup \emptyset$ represents the observation map. □

The RMPN considered in Fig. 1 has $\boldsymbol{m}_0 = [1, 1, 0, 0, 0, 0]^T$. By increasing or decreasing the number of robots in team, only $\boldsymbol{m}_0$ is modified. More details about RMPN systems are described in (Mahulea et al., 2020a), here being mentioned only some essential features. Thus, the RMPN has fixed topology with respect to the number of robots, while capturing team evolution based on various mathematical properties of PNs (Ezpeleta et al., 1998). By construction, an RMPN is a *state machine* Petri net.

The movement of robots is given by the sequence of adjacent cells that they need to follow throughout their paths. Two cells are adjacent if they have a common edge, e.g., $p_1, p_2$ from Fig. 1. In this sense, we are interested in finding a sequence of places towards team destination. This can be accomplished by searching a set of fired transitions which are enabled. A transition $t$ is enabled if the markings of all its input places are greater or equal with the weight of the arc entering to $t$. If $t$ is enabled, it may fire such that a number of $\boldsymbol{Pre}[p_i, t]$ tokens are consumed from all input places $p_i$ and a number of $\boldsymbol{Post}[p_j, t]$ tokens are produced in all output places $p_j$. If the sequence of fired transitions outputs a new marking $\tilde{\boldsymbol{m}}$ reached from the marking $\boldsymbol{m}$, then the fundamental (state) equation (1) is satisfied:

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \tag{1}$$

where, $\boldsymbol{C}$ is the token flow matrix, $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$, and $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ is the firing count vector, containing on its $j^{th}$ element the cumulative amount of firings of $t_j$ in $\sigma$. In the field of mobile planning, the firing of a transition is translated into a control law applied for a robot to move from one place to another by designing suitable vector fields (Belta and Habets, 2006). Sec. 4 exploits the use of firing count vector by minimizing the number of fired transitions, subject to the optimization problem: min $\mathbf{1}^T \cdot \boldsymbol{\sigma}$ (Silva et al., 1996). Based on $\boldsymbol{\sigma}$ and on the properties of the state machine, the robot paths are returned.

**Boolean-based specifications.** The team of robots is required to satisfy a given global formula over set $Y$. The formula specifies the visit or the avoidance of several regions of interest (ROIs). We consider the next notations:

- Set $Y_{im}$ for expressing the intermediate requirements, where ROIs are denoted by *capital* letters. Elements of $Y_{im}$ specify the ROIs to be visited/avoided *along trajectories*, excluding robots' final positions.
- Set $Y_f$ for illustrating the final requirements, ROIs being denoted by *lowercase* letters. Elements of $Y_f$ specify the ROIs to be visited/avoided at team's *final position*.

We will express the global Boolean-based formula as a Conjunctive Normal Form (CNF) (King et al., 2003), by $\varphi = \varphi_i \wedge \cdots \wedge \varphi_n$, where each term $\varphi_i$ is a disjunction of terms from $Y_{im}$ or $Y_f$. For example, formula $\varphi = y_1 \wedge Y_2$ indicates the visit of region $y_2$ along the trajectory and the stop of at least one robot in region $y_1$.

**Previous approach**. In our previous work (Mahulea et al., 2020b), we proposed a solution to the next problem: having a team of robots evolving in the environment $E$, and a given global Boolean-based formula $\varphi$, find collision free paths for the robots to fulfill the mission. The solution of this approach lies in solving two MILP problems based on the team model RMPN: for intermediate, respectively for final requirements within formula $\varphi$. In addition, both MILPs consider a fixed number of intermediate markings (equal to the number of robots) to avoid collisions in the robot paths. The conservatism of this approach lies in sequential reaching of the goal cells when the paths share *common resources* (defined below). The results display the unnecessary waiting of some robots until one team member reaches its destination. Therefore, the current work proposes a solution in terms of parallel motion of the robots.

**Assumptions and additional definitions.** We consider the assumptions from (Mahulea et al., 2020b).

(a) The restriction $\boldsymbol{m}_0[p] \leq 1, \forall p \in P$ is satisfied, meaning that all the robots are initially deployed in different partition cells (places of RMPN).

(b) Each disjunction $\varphi_i$ from the Boolean specification can contain only terms from $Y_{im}$ or $Y_f$.

(c) Each disjunction $\varphi_i$ for the intermediate, respectively final requirements, can be satisfied by a single deployment of the $r$ robots. In other words, there exist at least one marking of RMPN to fulfill the requirements for set $Y_{im}$, respectively $Y_f$.

(d) Each disjunction for the intermediate requirements can either include the visit of several ROIs, or it can specify the avoidance of some ROIs.

Alongside with these assumptions, we use the following terminology in this paper:

(i) An *obstacle place* denotes a place in RMPN corresponding to a ROI which should be avoided, as imposed by the intermediate or final requirements in $\varphi$.

(ii) A *step* is defined as a global period of time in which at least one robot moves to an adjacent cell.

(iii) A *resource* is defined by a cell included in the path, while a *common resource* is represented by a cell that is crossed by more than one robot along the paths.

(iv) The *collision of robots* is defined by the presence of at least two robots in the same cell at the same time (step). The situation of two robots swapping places is eliminated, as it will be observed in Section 4.

(v) A *process* is represented by the path of one robot expressed as the sequence of cells, which is later denoted with $Traj$. It is assumed that these (initial) paths are known, being returned by the solution proposed in (Mahulea et al., 2020b).

## 3. PROBLEM DEFINITION

**Problem.** The current work exploits the motion planning problem as follows: given a set of paths denoted $Paths$ for a team of $r$ identical robots, their evolution being modeled as a RMPN system, compute collision free parallel movements of the robots to fulfill a global Boolean-based specification.

*Example 3.* Let us recall the scenario illustrated in Fig. 1, and consider the following Boolean-based mission for the team of two robots:

$$\varphi = y_1 \wedge y_2 \wedge \neg Y_3 \tag{2}$$

This specifications requires to avoid region $y_3$ during robots movement, while the regions $y_1$ and $y_2$ are required to be visited at final positions. The method captured in (Mahulea et al., 2020b) returns two paths, by assigning the goal region $y_1$ to $r_2$ and $y_2$ to $r_1$. The paths of robots are indicated next, with the notation $\langle (r_i, p_j), (r_{i+1}, p_k) \rangle$, meaning robot $r_i$ is in cell $p_j$ and $r_{i+1}$ is in cell $p_k$, both at the same step. In the initial step (*step 0*), the *assigned resources* for $r_1, r_2$ are represented by $p_1, p_2$.

$$\begin{aligned} &Paths = \{ \\ &\quad \text{step 0: } \langle (r_1,p_1),(r_2,p_2)\rangle \text{ step 1: } \langle (r_1,p_1),(r_2,p_3)\rangle \\ &\quad \text{step 2: } \langle (r_1,p_1),(r_2,p_4)\rangle \text{ step 3: } \langle (r_1,p_1),(r_2,p_5)\rangle \\ &\quad \text{step 4: } \langle (r_1,p_2),(r_2,p_5)\rangle \text{ step 5: } \langle (r_1,p_3),(r_2,p_5)\rangle \\ &\quad \text{step 6: } \langle (r_1,p_4),(r_2,p_5)\rangle \qquad\qquad \} \end{aligned} \tag{3}$$

As it can be seen, the robots move sequentially: $r_1$ starts to move after $r_2$ reaches its destination cell $p_5$. Due to this delay, the mission is accomplished in 6 steps (recall assumption (ii) for the meaning of *step*). We propose to overcome the conservatism of this approach, by minimizing the number of steps to fulfill the team's mission. In this sense, our solution captures the collision free parallel movement of the robots following their paths, while rerouting them when needed. The rerouting is acquired by solving a MILP, based on the known current ($\boldsymbol{m}_0$) and final ($\boldsymbol{m}_f$) markings in the RMPN model. By applying the strategy to be detailed in Alg. 1, the robots can reach their final destination in 4 steps. Their movements are described by:

$$\begin{aligned} &Paths' = \{ \\ &\quad \text{step 0: } \langle (r_1,p_1),(r_2,p_2)\rangle \text{ step 1: } \langle (r_1,p_1),(r_2,p_3)\rangle \\ &\quad \text{step 2: } \langle (r_1,p_2),(r_2,p_4)\rangle \text{ step 3: } \langle (r_1,p_3),(r_2,p_5)\rangle \\ &\quad \text{step 4: } \langle (r_1,p_4),(r_2,p_5)\rangle \qquad\qquad \} \end{aligned} \tag{4}$$

## 4. ALGORITHM FOR PARALLEL MOVEMENT

As mentioned before, the current work is engaged in reducing the number of steps to satisfy the team mission. This can be achieved by a parallel movement of the robots. Let us denote with $Traj = \{Traj_1, Traj_2, \ldots, Traj_r\}$ the set of robot paths, $Traj_i$ being the trajectory of the robot $r_i \in R$. The path $Traj_i$ is given as the sequence of cells that robot $r_i$ should follow to reach its goal cell. For example, the path of $r_1$ is expressed as: $Traj_1 = [p_1, p_2, p_3, p_4]$. Although the procedure in (Mahulea et al., 2020b) is effective in sense of collision-free paths, there are multiple waiting moments of robots in some cells when there are many common resources along the path, thus leading to a larger number of steps. To overcome this,

**Algorithm 1:** Parallel movement of robots

```
   Input  : $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, Y, h\rangle, \boldsymbol{m}_f, Paths, N$
   Output: $Paths'$ /* Strategic robot's movement */
1  Compute $Traj$ based on $Paths$;
2  Compute the order of assigning resources $p_j \in P$ to processes in $Traj$;
3  Update $\mathcal{Q}$ by removing the obstacle places;
4  $Paths' =$ step 0 in $Paths$;
5  while ($\boldsymbol{m}_0 \neq \boldsymbol{m}_f$) do
6      $RobotsToMove = \emptyset$;
7      for $r_i \in R$ do
8          Let $p_j$ be the second place in $Traj_i$ /* it is $r_i$'s turn to enter $p_j$ */
9          if $p_j$ should be assigned next to $r_i$ AND no robot in $p_j$ then
10             $RobotsToMove = RobotsToMove \cup \{r_i\}$;
11         end
12     end
13     Add a new step in $Paths'$ moving $RobotsToMove$ to their next places while the other will keep their position;
14     Update $Traj$ by removing first places of all $r_i \in RobotsToMove$;
15     Update $\boldsymbol{m}_0$;
16     if ($|R \setminus RobotsToMove| \geq N$) OR ($|RobotsToMove| == 0$) then
17         Reroute the paths by solving MILP (5) for $\boldsymbol{m}_0$ and $\boldsymbol{m}_f$;
18         Compute $Traj$ based on rerouted paths;
19         Compute the order of assigning resources $p_j \in P$ to processes in $Traj$;
20     end
21 end
```

subsection 4.1 includes an algorithm for increasing parallel execution along the paths from $Traj$ and by rerouting the robots when there are multiple waitings at the same step.

### 4.1 Overall solution

The main idea of the solution is to change the initial movements captured by *Paths* by forcing the robots to move in parallel if possible. For this, we assume the system as a RAS, both terms *process* and *resource* being defined in the previous section. The problem can be seen as assigning resources to the processes in order to avoid deadlocks and collisions, while finishing all processes means to reach the final cells by all robots. The strategy relies on the notion of *safe state* from BA, which is transferred into the current approach. The proposed algorithm ensures a solution under the previously defined assumptions. First, the order of using each resource by the processes is computed based on *Paths*. This computation considers also back and forth scenarios for robot paths (at least one resource is utilized multiple times during the process). Notice that this order will ensure that all processes will finish as proved in (Mahulea et al., 2020b). Then, at any step, if a robot is not moving according to *Paths* but the next cell in which it should enter is free and this resource is assigned next to that robot, then this movement is shifted to the actual step in $Paths'$. Furthermore, if a number of $N$ or more robots are waiting for resources in one step, then the paths for all $r$ robots are recomputed.

Alg. 1 captures the overall steps for the robot motion, starting from *Paths* obtained by applying the approach in (Mahulea et al., 2020b). Beside this input, a threshold $N$ is defined by the user, to impose rerouting whenever at least $N$ robots are waiting in a step. Several actions are needed before computing the parallel execution of robots: compute the trajectories for the robots in $Traj$ (line 1); compute the order in which robots will cross through cells (line 2); remove the places that shouldn't be crossed along the trajectories (line 3), and initialize the starting cells of the robots in the initial step - step 0 (line 4).

Line 6 initializes the variable $RobotsToMove$ to empty set, containing the set of robots moving in the actual step. The loop in lines 7 to 12 checks for each robot if it can move in the actual step. This occurs if the next place is free and it is the robot's turn to cross that place. Once the set of moving robots is computed, the parallel motion of robots is captured by lines 13-15. The last part of the algorithm determines the necessity to reroute paths. The MILP responsible for rerouting (described in the subsection 4.2) is enforced, with the updated $\boldsymbol{m}_0$ based on robot's current position (lines 17-19). The robot paths are recomputed when the number of waiting robots reaches the threshold $N$, or when no robot is able to move in the current step. The latter examination avoids deadlocks, which would otherwise be possible if the order in which robots reach their final cells is modified (by a previous rerouting) and some robots block the paths of others.

### 4.2 Optimization problem to reroute paths

To accomplish parallel movement, the rerouting of paths is made for all robots. The MILP (5) ensures that the robots can finish their trajectories in a sequential order thus avoiding collisions and deadlocks. However, the implementation using the strategy in previous subsection ensures the parallel movement of the robots based on the solution of MILP. The unknown variables consist in the initial $\boldsymbol{m}_{0,i}$ and final $\boldsymbol{m}_i$ markings, joined by the firing count vector denoted as $\boldsymbol{\sigma}_i$, for each robot $r_i \in R$. The unknown vectors $\boldsymbol{\sigma}_i$ are also used in the cost function, to minimize the weighted sum of the $r$ firing count vectors. Compared with our previous approaches in designing MILPs for robot movement strategies - where $\boldsymbol{\sigma}$ gathers the firing sequence for the entire team -, herein the robots' executions are accounted individually, based on the mentioned unknowns. By minimizing this cost function, it is not possible for two robots to swap their places, hence assumption (iv) is fulfilled. The MILP (5) has the following constraints:

- Constraints (a) represent the state equation (1) for a sequence of markings $\boldsymbol{m}_i, i = 1, \ldots, r$. A sequence of markings is used in order to avoid the collisions.
- Constraints (b) impose the presence of maximum one robot in each place, considering the final positions of the previous $(i-1)$ robots and the initial positions of the next $(r-i)$ robots.
- Constraints (c) and (d) ensure the gathering of the initial $\boldsymbol{m}_{0,i}$ and final $\boldsymbol{m}_i$ markings for all robots

$r_i \in R$, in accordance with the initial $\boldsymbol{m}_0$ and final $\boldsymbol{m}_f$ markings of RMPN team model.
- Constraints (e) complement the previous constraints, by ensuring $\boldsymbol{m}_{0,i}$ is the marking for a single robot.
- Constraints (f) specify the type of unknown variables.

$$\begin{aligned}
&\min \mathbf{1}^T \cdot \sum_{i=1}^{r} i \cdot \boldsymbol{\sigma_i} \\
\text{s.t. } & \boldsymbol{m}_i = \boldsymbol{m}_{0,i} + C \cdot \boldsymbol{\sigma_i}, \quad {\scriptstyle i=1,2\ldots r} && (a)\\
& \boldsymbol{Post} \cdot \boldsymbol{\sigma}_i + \sum_{j=1}^{i-1} \boldsymbol{m}_i + \sum_{j=i+1}^{r} \boldsymbol{m}_{0,i} \leq \mathbf{1}, \quad {\scriptstyle i=1,2\ldots r} && (b)\\
& \sum_{i=1}^{r} \boldsymbol{m}_{0,i} = \boldsymbol{m}_0 && (c)\\
& \sum_{i=1}^{r} \boldsymbol{m}_i = \boldsymbol{m}_f && (d)\\
& \mathbf{1}^T \cdot \boldsymbol{m}_{0,i} = 1, \quad {\scriptstyle i=1,2\ldots r} && (e)\\
& \boldsymbol{m}_{0,i} \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{m}_i \in \mathbb{R}_{\geq 0}^{|P|}, \boldsymbol{\sigma}_i \in \mathbb{N}_{\geq 0}^{|T|}, \quad {\scriptstyle i=1,2\ldots r} && (f)
\end{aligned} \tag{5}$$

**Remark.** The limitations of our proposed algorithm are two-folded: the selected number of waiting robots $N$ does not provide any monotony with respect to the number of global steps in the team path planning (see Table 1); in some scenarios, our solution may return the same robot motion as in the previous work (Mahulea et al., 2020b).

## 5. NUMERICAL RESULTS

The proposed algorithm was implemented and integrated in the open-source toolbox RMTool - MATLAB (González et al., 2015). The simulation results are obtained on a computer with i7 - 8th gen. CPU @ 2.20GHz and 8GB RAM, and rerouting MILP (5) was solved with CPLEX.

*Example 4.* Let us recall the example considered throughout the previous sections, with the Boolean-based formula from (2). The mission enforces the robots to reach regions $y_1, y_2$, while avoiding $y_3$. The solution returned by the current method is given in eq. (4), for $N = 1$. Thus, the number of steps to fulfill the Boolean-based mission was reduced compared with the approach from (Mahulea et al., 2020b), the total steps herein being equal with 4, instead of 6. It can be observed that in the first step only $r_2$ is moving: its next cell ($p_3$) is free, so it can advance; at the same step, $r_1$'s next cell is $p_2$ which is still occupied by $r_2$. For this small example, the running time of Alg. 1 is insignificant, being close to 0 seconds. ■

*Example 5.* To better examine the quality of our solution, let us consider the relevant example in which all the robots should pass through a common free passage to reach their destination. Fig. 2 illustrates a grid-based environment with 5 × 5 cells, including 8 ROIs and 4 robots placed in cells $p_1$, $p_6$, $p_{16}$, $p_{21}$ (first column from the left). The Boolean-based specification is the following: $\varphi = \neg Y_1 \wedge \neg Y_2 \wedge \neg Y_3 \wedge \neg Y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$, which means the avoidance of the first 4 regions along trajectories and visiting the last 4 regions at destination cells.

For this example, the running time to obtain the robot trajectories as in (Mahulea et al., 2020b) is 0.1 seconds. The mission is accomplished in 24 steps with the previous approach, in comparison with 11 steps given by the current approach. The parallel movement of the robots along the paths does not require a perceptible additional running time, especially if no rerouting is triggered. The order in which the robots reach their final cell is captured in Table 1. For the imposed threshold $N = 2$ (computing new paths whenever at least 2 robots are waiting in their current cells), the paths are rerouted 2 times based on MILP (5), yielding a total number of steps equal with 10. Fig. 2 captures the paths of robots with dashed lines ($r_1$ - red, $r_2$ - blue, $r_3$ - pink, $r_4$ - green). In addition, the figure illustrates the current positions of the robots and their movement intention (colored arrows) immediately before the path rerouting actions were triggered. The performances of running time for path rerouting are captured in mean $\mu$ and standard deviation *std* given in Table 1. ■

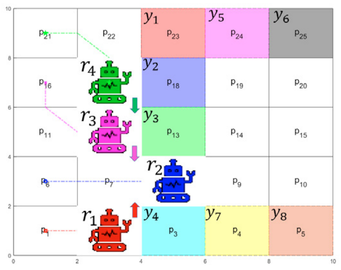

Fig. 2. Grid environment with 4 robots and 8 ROIs

Table 1 captures quantitative results obtained for scenarios with common resources and increased complexity in terms of number of cells and team size. The running time seized in the table is computed for solving the considered optimization problem for each approach, as follows: the MILPs in the previous work (Mahulea et al., 2020b) computes robot paths for satisfying the Boolean-based formula based on a global model of the team. The MILP in the current work reroutes the robots when needed: *Case I* rerouting only when a final state is occupied but should be crossed by another robot and *Case II* rerouting when $N$ robots are waiting. In both cases the final cells are maintained for the team, while collision-free paths are ensured.

Note that path rerouting is *Not Applicable* (NA) for the previous work, since this action is part of current contributions. The first set of lines shows the results for Example 5. The second set of lines gives simulation results for the scenario included in (Mahulea et al., 2020b) (grid with 20 × 10 cells). A movie depicting the robot movements in these scenarios can be inspected in the animated simulation from this link. The third set of lines from Table 1 considers a grid-based environment with 20 × 20 cells and $r = 20$. In this scenario, the parallel movement for $N = r$ includes two rerouting actions triggered by the inability of some robots to move due to other robots that already reached their final destinations and blocked some passages. When a trigger of $N = 10$ is imposed for the parallel movement, the robots reach their destination in 77 steps. All these results demonstrate the benefits of having a parallel robot motion with the possibility of rerouting, especially for large teams of mobile robots.

## 6. CONCLUSION

The current paper focused on reducing the conservativeness of our previous work (Mahulea et al., 2020b) by imposing a collision free parallel movement of the robots.

Table 1. Numerical data comparison: previous approach vs. current approach

| **Environment scenario** | **Numerical performances** | **Previous approach** | ***Case I*** | ***Case I* or *Case II* (here $N = \lfloor r/2 \rfloor$)** |
|---|---|---|---|---|
| **Grid-based 5 × 5, with $r = 4$ robots** | Number of steps | 24 | 11 | 10 |
| | Number of path rerouting | NA | 0 | 2 times |
| | Running time [sec] | 0.1 | 0.1 | $0.1 + (\mu = 7 \cdot 10^{-3}$, std $= 1 \cdot 10^{-2})$ |
| | Order of robots to reach their final cell | $r_3, r_1, r_4, r_2$ | $r_2, r_1, r_3, r_4$ | $r_2, r_1, r_3, r_4$ |
| | Assigned final regions for tupla $(r_1, r_2, r_3, r_4)$ | $(y_5, y_7, y_6, y_8)$ | $(y_5, y_7, y_6, y_8)$ | $(y_6, y_5, y_8, y_7)$ |
| **Grid-based 10 × 20, with $r = 10$ robots** | Number of steps | 155 | 39 | 43 |
| | Number of path rerouting | NA | 0 | 6 times |
| | Running time [sec] | 3.2 | 3.2 | $3.2 + (\mu = 1 \cdot 10^{-2}$, std $= 1 \cdot 10^{-2})$ |
| **Grid-based 20 × 20, with $r = 20$ robots** | Number of steps | 352 | 78 | 7 |
| | Number of path rerouting | NA | 2 times | 14 times |
| | Running time [sec] | 12 | 12 0 | $12 + (\mu = 9 \cdot 10^{-3}$, std $= 3 \cdot 10^{-3})$ |

The proposed algorithm releases dynamically the common resources visualized as common cells along the paths. This decreases the number of steps in which the team fulfills its mission. In addition, an improvement is made by applying a path rerouting approach whenever the number of robots waiting in the current cells reaches a given threshold or when no robot can move in the current step. The path rerouting is solved based on a MILP formulation, which requires the current and final markings of RMPN model. Future work envisions algorithms suitable for incomplete inputs, such as partially known paths.

## REFERENCES

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G.J. (2007). Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1), 61–70.

Belta, C. and Habets, L.C. (2006). Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11), 1749–1759.

Bobanac, V. and Bogdan, S. (2008). Routing and scheduling in multi-AGV systems based on dynamic banker algorithm. In *16th Mediterranean Conference on Control and Automation*, 1168–1173. IEEE.

Ding, X.C., Kloetzer, M., Chen, Y., and Belta, C. (2011). Automatic deployment of robotic teams. *IEEE Robotics & Automation Magazine*, 18(3), 75–86.

Ezpeleta, J., García-Vallés, F., and Colom, J.M. (1998). A class of well structured Petri nets for flexible manufacturing systems. In *International Conference on Application and Theory of Petri Nets*, 64–83. Springer.

González, R., Mahulea, C., and Kloetzer, M. (2015). A matlab-based interactive simulator for mobile robotics. In *CASE'2015: Int. Conf. on Autom. Science and Eng.*

Hönig, W., Kiesel, S., Tinka, A., Durham, J., and Ayanian, N. (2018). Conflict-based search with optimal task assignment. In *Int. Joint Conf. on Autonomous Agents and Multiagent Systems.*

Hustiu, S., Hustiu, I., Kloetzer, M., and Mahulea, C. (2021). LTL task decomposition for 3D high-level path planning. *Journal of Control Engineering and Applied Informatics*, 23(3), 76–87.

Kalinovcic, L., Petrovic, T., Bogdan, S., and Bobanac, V. (2011). Modified Banker's algorithm for scheduling in multi-AGV systems. In *International Conference on Automation Science and Engineering*, 351–356. IEEE.

King, J., Pretty, R.K., and Gosine, R.G. (2003). Coordinated execution of tasks in a multiagent environment. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(5), 615–619.

Lacerda, B. and Lima, P.U. (2019). Petri net based multi-robot task coordination from temporal logic specifications. *Robotics and Autonomous Systems*, 122, 103289.

Lawley, M., Reveliotis, S., and Ferreira, P. (1998). The application and evaluation of banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 10(1), 73–100.

Mahulea, C., González, R., Montijano, E., and Silva, M. (2021). Path planning of multirobot systems using Petri net models. results and open problems. *Rev. Iberoamericana de Autom. e Infor. Ind.*, 18(1), 19–31.

Mahulea, C., Kloetzer, M., and González, R. (2020a). *Path Planning of Cooperative Mobile Robots Using Discrete Event Models.* John Wiley & Sons.

Mahulea, C., Kloetzer, M., and Lesage, J.J. (2020b). Multi-robot path planning with boolean specifications and collision avoidance. *IFAC-PapersOnLine*, 53(4), 101–108.

Reveliotis, S.A. (2000). Conflict resolution in AGV systems. *IIE Transactions*, 32(7), 647–659.

Schillinger, P., Bürger, M., and Dimarogonas, D.V. (2018). Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The Int. Journal of Robotics Research*, 37(7), 818–838.

Silva, M., Teruel, E., and Colom, J.M. (1996). Linear algebraic and linear programming techniques for the analysis of place/transition net systems. In *Advanced Course on Petri Nets*, 309–373. Springer.

Song, D., Li, Y., and Song, T. (2021). Modified Banker's algorithm with dynamically release resources. In *International Conference on Communications, Information System and Computer Engineering*, 566–569. IEEE.

Tricas, F., Colom, J.M., and Ezpeleta, J. (2000). Some improvements to the Banker's algorithm based on the process structure. In *Int. Conf. on Robotics and Automation*, volume 3, 2853–2858.

Yu, D., Hu, X., Liang, K., and Ying, J. (2021). A parallel algorithm for multi-AGV systems. *Journal of Ambient Intelligence and Humanized Computing*, 1–15.