Multi-robot Path Planning with Boolean Specifications and Collision Avoidance

Cristian Mahulea, Marius Kloetzer, Jean-Jacques Lesage

Multi-robot Path Planning with Boolean Specifications and Collision Avoidance ⋆

Cristian Mahulea [*], **Marius Kloetzer** [**],
Jean-Jacques Lesage [***]

[*] *Aragón Institute of Engineering Research (I3A), University of Zaragoza, Zaragoza, Spain (e-mail: cmahulea@ unizar.es)*
[**] *"Gheorghe Asachi" Technical University of Iasi, Iasi, Romania (e-mail: kmarius@ac.tuiasi.ro)*
[***] *LURPA, ENS Paris-Saclay, Université Paris-Saclay, 94235 Cachan, France (e-mail: jean-jacques.lesage@ens-paris-saclay.fr)*

Abstract: In this paper we consider the path planning problem for a team of identical mobile robots that should fulfill a given global specification. This specification is given as a Boolean formula over some regions of interest and should be satisfied on the final state (when the robots stop) and on the trajectories. The main novelty of this paper is the automatic computation of collision-free trajectories. The approach is based on a Petri net model and on solving two Mixed Integer Linear Programming problems. Based on the solutions of these problems, intermediate synchronization points are introduced in order to avoid possible collisions. Additionally, the algorithm in this paper is implemented in an open-source Matlab toolbox, called RMTool.

Keywords: discrete event systems, path planning, multi-robot systems

1. INTRODUCTION

This paper deals with path planning problem for teams of identical robots that consists in computing trajectories for the robots to achieve a given specification expressed as a Boolean formula over some regions of the environment. The specification is global and the robots should cooperate in order to fulfill it.

Path-planning with high-level specifications is a problem extensively studied in literature, both for single robot Belta et al. (2007); Kress-Gazit et al. (2009); Fainekos et al. (2009); Ding et al. (2014); Kloetzer and Mahulea (2015) or for multi-robot systems Schillinger et al. (2018); Lacerda and Lima (2019); Kloetzer and Mahulea (2020). In the case of multi-robot systems, one of the main problems is to obtain a compact model for the team of robots. One possibility to tackle this problem is to use Petri net models Lacerda and Lima (2019); Mahulea et al. (2020) that are scalable with respect to the number of robots (assuming identical robots), i.e., by adding a new robot to the team, the structure of the model is not changing but only the initial marking (state).

In Mahulea and Kloetzer (2018) a solution for the path-planning problem in multi-robot systems with Boolean specifications is proposed. In particular, two optimization problems are presented. The first one permits to obtain trajectories when the specification contains only constraints on the final state while the second problem considers also constraints on the trajectories. The computational complexity of the second problem is very big since the number of variables in the Mixed Integer Linear Programming (MILP) problem is depending on the number of intermediate markings (states). These markings are used to avoid solutions containing *spurious firing vectors* (firing vectors not corresponding to any firing sequence). The number of intermediate markings should be big enough to allow the robots to reach the required *cells* (or regions in which the environment has been partitioned), being upper-bounded by the number of transitions of the Petri net model (equal to twice the number of common edges). Therein, the number of possible collisions is reduced, but collision-free trajectories were not guaranteed.

This paper proposes a method to compute collision-free trajectories by using a fixed number of intermediate markings, equal to the number of robots. Furthermore, assuming that the formula can be divided into two independent parts, one for the trajectory and one corresponding to the final state, we show that the solution for the planning problem can be obtained by solving two independent MILPs. The first one is used to compute an intermediate state at which the formula on the trajectory is fulfilled, while the second one is used to obtain a final marking to fulfill the formula on the final state. In both cases, we introduce a number of intermediate markings corresponding to the synchronization points, ensuring that the collisions cannot appear. Under the assumptions on the formula, the proposed MILPs return firing vectors that are easy to transform into firing sequences, hence to robot movements. Spurious firing vectors will not appear, meaning that under these assumptions it is not necessary to introduce intermediate markings to ensure the fireability of the firing vectors as in Mahulea and Kloetzer (2018). Furthermore, the proposed algorithms are implemented in

⋆ The work of C. Mahulea has been partially supported by the MINECO "Salvador de Madariaga" mobility program. M. Kloetzer acknowledges the grant PN-III-P1-1.1-TE-2016-0737.

Robot Motion Toolbox (RMTool), an open-source MATLAB toolbox Parrilla et al. (2017).

2. PRELIMINARIES AND PROBLEM DEFINITION

Robot workspace and team model. We consider an environment where a number of N_r identical robots evolve, the robots being labeled with $r_1, r_2, \ldots, r_{N_r}$. In the environment there exist some disjoint regions of interest, labeled with elements from the set $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$. Common to multiple planning scenarios, the robots are reduced to points and the environment is assumed to be partitioned in a set of regions (cells), e.g., by an existing cell decomposition method Choset et al. (2005); Mahulea et al. (2020). Thus, the motion of a robot is basically a movement over a discrete event system, i.e., moving from one cell to an adjacent one means taking a transition in the discrete abstraction.

The set of cells is denoted by $P = \{p_1, p_2, \ldots, p_{|P|}\}$. Since the regions of interest $\mathcal{Y}$ are disjoint and their boundaries are not crossed by partition regions, each cell from P corresponds to either a region of interest or to the free space. The observation of each cell is given by function $h : P \to \mathcal{Y} \cup \{\emptyset\}$, with $h(p_i) = y_j$ if cell p_i is included in or equal to region y_j, and $h(p_i) = \emptyset$ if p_i does not belong to any region from $\mathcal{Y}$.

Example 1. Fig. 1 shows an environment constructed in RMTool that has been partitioned in 200 grid-based cells denoted by p_1 (bottom-left) to p_{200} (top-right). There are 20 regions of interest (the ones filled with different colors), 10 of them placed in the middle of the environment and 10 in the right. For example, cell p_{10} corresponds to the first region of interest and it is labeled by y_1, p_{30} is the second region of interest labeled by y_2, etc. Hence, $h(p_{10}) = y_1$, $h(p_{30}) = y_2$, while $h(p_2) = \emptyset$. ■

Under the above, we abstract the evolution of the robotic team to a Robot Motion Petri Net (RMPN) Mahulea and Kloetzer (2018); Kloetzer and Mahulea (2020) $\mathcal{Q} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post}, \boldsymbol{m}_0, \mathcal{Y}, h\rangle$, where: P is the set of places (one place for each cell); T is the set of transitions, each transition corresponding to a robot movement between adjacent cells; $\boldsymbol{Post} \in \{0,1\}^{|P|\times|T|}$ is the post-incidence matrix, defining the arcs from transitions to places; $\boldsymbol{Pre} \in \{0,1\}^{|P|\times|T|}$ is the pre-incidence matrix defining the arcs from places to transitions; $\boldsymbol{m}_0$ is the initial marking, where $\boldsymbol{m}_0[p]$ gives the number of robots initially deployed in cell $p \in P$; $\mathcal{Y} \cup \{\emptyset\}$ is the set containing the output symbols, $\emptyset$ being the empty symbol; and $h : P \to \mathcal{Y} \cup \{\emptyset\}$ is the observation map, defined above. Thus, if p_i has at least one token (i.e., at least one robot is currently in cell p_i), then region of interest $h(p_i)$ is visited.

Additional details on RMPN can be found in Mahulea et al. (2020), and here we briefly recall some important aspects. RMPN $\mathcal{Q}$ has N_r tokens (each one corresponding to a robot), and it models the evolution of the entire team by maintaining a fixed topology, since its places and transitions do not change when adding or removing robots. Moreover, each transition from T has only one input and one output place, and therefore $\mathcal{Q}$ is a *state machine*.

For a generic transition $t_j \in T$, ${}^\bullet t$ denotes its input place, while $t^\bullet$ denotes its output place. Same notations are used for places, when denoting their input and output transitions. Formally, ${}^\bullet t_j = \{p_i \in P | Pre[p_i, t_j] = 1\}$ and $t_j{}^\bullet = \{p_i \in P | Post[p_i, t_j] = 1\}$. Transition $t_j \in T$ is enabled at marking $\boldsymbol{m}$ if its input place contains at least one token [1], i.e., $\boldsymbol{m}[p_i] \geq 1$, where $p_i = {}^\bullet t_j$. An enabled transition t_j can fire, and the RMPN reaches a new marking $\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C}[\cdot, t_j]$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token flow matrix and $\boldsymbol{C}[\cdot, t_j]$ is its column corresponding to t_j.

According to RMPN structure, the firing of a transition t corresponds to the movement of a robot from cell p_i to cell p_j, where $\boldsymbol{Pre}[p_i, t] = 1$ and $\boldsymbol{Post}[p_j, t] = 1$. For the moving robot, transition t thus means to apply a control law that drives the robot from cell p_i to p_j, and there exist approaches for designing such continuous laws Habets et al. (2006); Belta and Habets (2006).

We will be interested in finding sequences of transitions to be fired such that the team fulfils a given specification. If a RMPN marking $\tilde{\boldsymbol{m}}$ can be reached from $\boldsymbol{m}$ through a finite sequence of transition firings, we denote with $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ the firing count vector, i.e., its j^{th} element is the cumulative amount of firings of t_j. In this case, the state (or fundamental) equation (1) is satisfied.

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

For a live [2] state machine Petri net system as is our model $\mathcal{Q}$, the solutions of the fundamental equation (1) give the set of reachable markings Silva et al. (1998). Moreover, for this class of Petri nets, if we find a firing vector $\boldsymbol{\sigma}$ that drives the RMPN to a desired marking by firing the minimum number of transitions (i.e., $\boldsymbol{\sigma}$ is solution of the following optimization problem: $\min \mathbf{1}^T \cdot \boldsymbol{\sigma}$ subject to (1)), it can be transformed into the corresponding sequence of robot movements Mahulea et al. (2020). However, if constraints are imposed on the intermediate markings (or equivalently on $\boldsymbol{\sigma}$) this is not true in general because of the empty cycles (subtours disjoint from the main trajectory) Silva et al. (1998). These cycles could be included in the firing count vector $\boldsymbol{\sigma}$ in order to satisfy the constraints on the intermediate markings but the resulted $\boldsymbol{\sigma}$ is not corresponding to any firing sequence (see Example 2).

For each region of interest y_i, we denote by $\boldsymbol{v}_i \in \{0,1\}^{1\times|P|}$ its characteristic vector, where $\boldsymbol{v}_i[p_k] = 1$ if $h(p_k) = y_i$ and $\boldsymbol{v}_i[p_k] = 0$ otherwise, $\forall p_k \in P$. In words, at any given marking $\boldsymbol{m}$, the region y_i is visited by at least one robot if we have $\boldsymbol{v}_i \cdot \boldsymbol{m} > 0$.

Example 2. Let us consider again the environment in Example 1. The Petri net model of the part of the environment delimited by the red rectangle in Fig. 1 is given in Fig. 2. This RMPN part is composed of four places, $\{p_9, p_{10}, p_{29}, p_{30}\}$ corresponding to the cells with the same name. Assuming omnidirectional robots, transitions are corresponding to the adjacency relation. For example, because cells p_9 and p_{10} are adjacent, two transitions are added to the model, t_{13} modeling the movement of a robot from p_9 to p_{10} and t_{14} for the movement from p_{10} to p_9.

[1] Petri net systems considered in this paper are ordinary.

[2] A Petri net is live if independently by the actual reachable marking, all transitions can fire in the future.

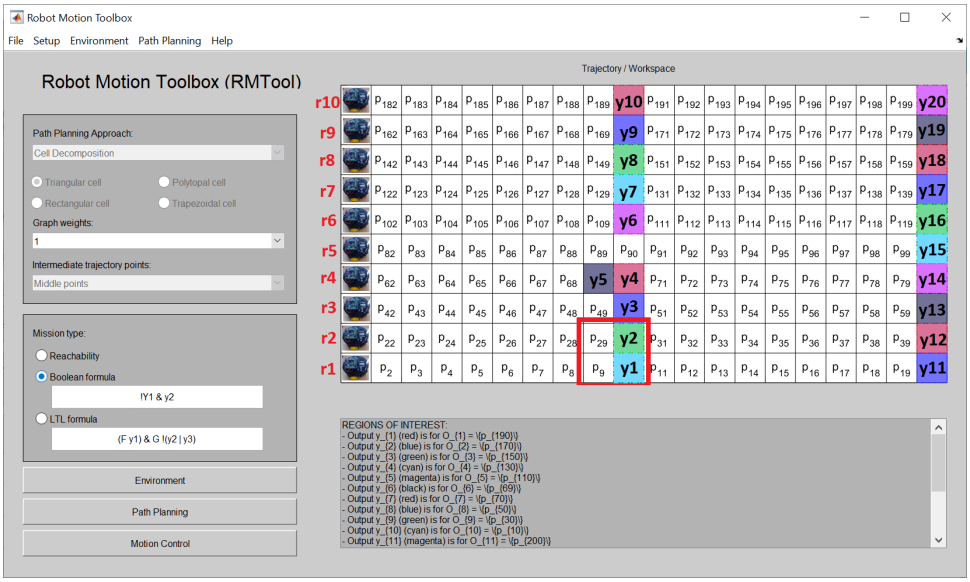


Fig. 1. Example of an environment composed by 200 grid cells and 20 regions of interest, constructed with RMTool.

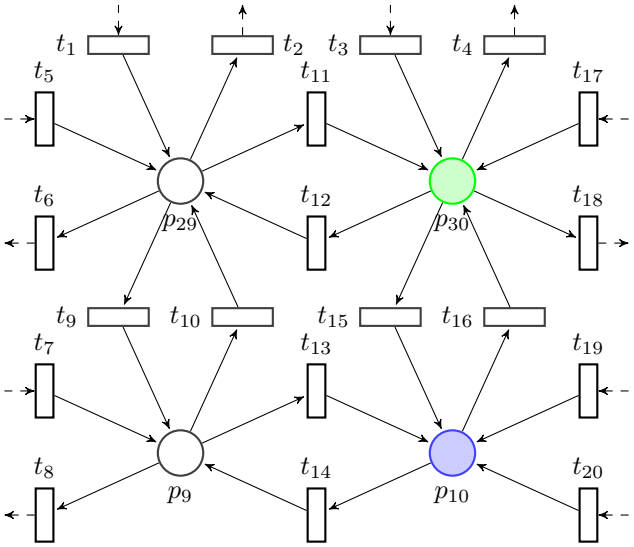


Fig. 2. Part of the RMPN modeling the environment in Fig. 1 (corresponding to the red rectangle).

Since p_{10} corresponds to the first region of interest labeled y_1, the characteristic vector of y_1, denoted $\boldsymbol{v}_1$, is such that $\boldsymbol{v}_1[p_{10}] = 1$. Assume now that there is only one robot initially located in p_9 and it should arrive at the final state in p_{10}. A firing vector can be obtained by solving the following MILP:

$$\min \ \mathbf{1}^T \cdot \boldsymbol{\sigma} \quad \text{s.t.} \begin{cases} \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \\ \boldsymbol{v}_1 \cdot \boldsymbol{m} \geq 1, \end{cases} \tag{2}$$

second constraint being equivalent to $m[p_{10}] \geq 1$. Obviously, the solution of this MILP is a firing vector $\boldsymbol{\sigma}$ having all elements equal to zero except $\boldsymbol{\sigma}[t_{13}] = 1$. In this case, the firing sequence is composed by only one transition, t_{13}.

Assume now that the mission of the robot in p_9 is again to finally reach p_{10}, but during the trajectory the robot should pass through p_{30} (corresponding to the second region of interest, y_2). A first idea is to add to MILP (2) a new constraint forcing the firing of one transition of ${}^\bullet p_{30}$.

$$\min \ \mathbf{1}^T \cdot \boldsymbol{\sigma} \quad \text{s.t.} \begin{cases} \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}, \\ \boldsymbol{v}_1 \cdot \boldsymbol{m} \geq 1, \\ \sum_{t \in {}^\bullet p_{30}} \boldsymbol{\sigma}[t] \geq 1, \end{cases} \tag{3}$$

with a possible solution $\boldsymbol{\sigma}$ with all elements equal to zero except, $\boldsymbol{\sigma}[t_{13}] = \boldsymbol{\sigma}[t_{11}] = \boldsymbol{\sigma}[t_{12}] = 1$. Obviously, this firing vector cannot be transformed into a fireable firing sequence. This happens because the T-semiflow $t_{11} + t_{12}$ is added to the solution ensuring that a token will be created in p_{30}, but it cannot be fired if no token is in p_{29}.

In order to solve the above problem of *spurious firing vectors*, in Mahulea and Kloetzer (2018) a number of intermediate markings are added and at each step a robot can advance only to an adjacent cell. In this paper, less expressive Boolean formulas are considered, which will allow us to reach one intermediate marking at which the part of formula on trajectory is satisfied. The only constraints that we will introduce on the firing vectors is to avoid some regions (and not to reach them), and this guarantees that the solution is not spurious. ■

Boolean-based specifications. The team of robots is required to move so that it fulfils a Boolean formula over the set of regions of interest. The formula can include requirements on both the robot trajectories and on their final (stopping) positions. Formally, the formula is given over set $\mathcal{Y}_i \cup \mathcal{Y}_f$, where $\mathcal{Y}_i = \{Y_1, Y_2, \ldots, Y_{|\mathcal{Y}|}\}$ and $\mathcal{Y}_f = \mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$, with the following meaning:

- Set $\mathcal{Y}_i$ refers to *intermediate requirements* on robot trajectories, i.e., it is used to specify which regions of interest are to be visited or avoided during the robot motion, excluding their stopping positions;
- Set $\mathcal{Y}_f$ refers to *final requirements* and it is used for indicating regions in which the robots should or should not remain at the end of their movement.

For example, formula $\varphi = Y_1 \wedge \neg Y_2 \wedge y_2$ requires that the team should visit along trajectory the region y_1, it should

avoid along trajectory the region y_2 and at least one robot must stop (remain for all future times) in region y_2.

We assume that any given Boolean formula is expressed in Conjunctive Normal Form (CNF), mentioning that any Boolean expression can be transformed in CNF King et al. (2003). Thus, we denote the formula that should be satisfied by the team movement by $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$, where each term φ_i is a disjunction of terms from $\mathcal{Y}_i \cup \mathcal{Y}_f$.

Problem statement. Given (1) a RMPN corresponding to the movement of a robotic team in their workspace and (2) a Boolean formula φ expressing the regions of interest that should be visited or avoided along trajectories and in final positions, find a collision-free sequence of robotic movements such that the specification is accomplished.

In Mahulea and Kloetzer (2018) we have proposed a solution for this problem, based on solving some MILP optimizations. Here, under some additional assumptions, we will propose a solution that guarantees collision avoidance.

Assumptions. As in Mahulea and Kloetzer (2018), the robots are assumed to be able to synchronize if required, i.e., to leave some partition cells at the same time. Additionally to Mahulea and Kloetzer (2018), we assume:

(i) Each disjunction φ_i of formula contains only terms from either the set $\mathcal{Y}_i$ or $\mathcal{Y}_f$, but not from both.
(ii) The part of the formula referring to intermediate requirements can be satisfied by a single deployment of the N_r robots, i.e., there exists at least a marking of $\mathcal{Q}$ where all intermediate requirements are fulfilled. Of course, the same should hold for the final requirements, otherwise the satisfaction of the formula being impossible.
(iii) Each disjunction over $\mathcal{Y}_i$ (intermediate requirements) can contain either multiple non-negated elements, or it is solely formed by a negated region.
(iv) The robots are initially deployed in different cells, yielding that the initial RMPN marking satisfies $\boldsymbol{m}_0[p] \leq 1$, $\forall p \in P$.

Note that the above requirements (i)-(iii) yield less expressive requirements than in Mahulea and Kloetzer (2018). For example, due to assumption (i) we cannot have a specification containing $(Y_1 \vee y_2)$. Due to (ii), if there is only one robot, the specification $Y_1 \wedge Y_2$ will be impossible, while the approach from Mahulea and Kloetzer (2018) would return a solution that visits at different moments the two different regions. Finally, assumption (iii) forbids disjunctive terms as $(Y_1 \vee \neg Y_2)$. However, under these assumptions, the solution from Section 3 will guarantee that the robots cannot collide during their motion, even if their trajectories intersect.

3. COLLISION-FREE MOVEMENT PLANS

In this section we present a new approach for solving the collision-free path planning problem for specifications given as Boolean formulas on final state and on trajectory. First we recall some previous results used to manage the Boolean formulas and then we develop two new MILPs that guarantee collision-free trajectories.

Inequalities corresponding to φ. The Boolean formula φ can be transformed into a set of n linear inequalities, one inequality for each disjunctive term. These translation is detailed in Mahulea and Kloetzer (2018) and it is briefly recalled here for completeness of presentation.

Define a binary vector $\boldsymbol{x}$ with $2 \cdot |\mathcal{Y}|$ variables, denoted by $\boldsymbol{x} = [x_{Y_1}, x_{Y_2}, \ldots, x_{Y_{|\mathcal{Y}|}}, x_{y_1}, x_{y_2}, \ldots, x_{y_{|\mathcal{Y}|}}]^T \in \{0,1\}^{2\cdot|\mathcal{Y}|}$, as follows: $x_{Y_i} = 1$ (respectively $x_{y_i} = 1$) if region labeled with y_i is visited along trajectory (respectively in the final state) by at least one robot, and $x_{Y_i} = 0$ (respectively $x_{y_i} = 0$) otherwise.

For each disjunction φ_i, construct a function $\alpha_i : \mathcal{Y}_i \cup \mathcal{Y}_f \to \{-1, 0, 1\}$ as:

$$\alpha_i(\gamma) = \begin{cases} -1, & \text{if } \varphi_i = \neg\gamma \\ 0, & \text{if } \gamma \text{ does not appear in } \varphi_i \\ 1, & \text{if } \gamma \text{ appears in } \varphi_i \end{cases}, \forall \gamma \in \mathcal{Y}_i \cup \mathcal{Y}_f \tag{4}$$

Now, disjunction φ_i is equivalent to,

$$\sum_{\gamma \in \mathcal{Y}_i \cup \mathcal{Y}_f} (\alpha_i(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{Y}_i \cup \mathcal{Y}_f} min\left(\alpha_i(\gamma), 0\right). \tag{5}$$

Of course, due to assumption (i), the sums from (5) can be taken for set $\mathcal{Y}_i$ or for $\mathcal{Y}_f$, but not both.

The intuition behind (4) and (5) is the following: if φ_i does not include the region corresponding to a symbol γ, then the binary variable for γ can have any value, without affecting φ_i's truth value. If φ_i includes more non-negated regions, the sum of their binary variables should be greater or equal than 1 in order to visit at least one of these regions. If φ_i equals a negated region, its corresponding binary variable x_γ should be 0, equivalent with $-x_\gamma \geq 0$ that would result from (5).

Solution for intermediate requirements. Let $\bar{\mathcal{Y}} \subseteq \mathcal{Y}_i$ be the set of regions that appear negated on the trajectory requirements, i.e., those mentioned at the end of assumption (iii). Formally, $\bar{\mathcal{Y}} = \{Y_j \in \mathcal{Y}_i | \exists \varphi_i \text{ s.t. } \varphi_i = \neg Y_j\}$. Let $\boldsymbol{\eta} \in \{0,1\}^{1\times|T|}$ be a row vector with $\boldsymbol{\eta}[^\bullet p_k] = 1$ for any p_k for which $h(p_k) \cap \bar{\mathcal{Y}} \neq \emptyset$, and $\boldsymbol{\eta}[^\bullet p_k] = 0$ otherwise. Basically, vector $\boldsymbol{\eta}$ indicates the transitions that should not fire in order to avoid the regions negated in the intermediate requirements.

$$\begin{aligned}
&\min \mathbf{1}^T \cdot \sum_{j=1}^{N_r+1} j \cdot \boldsymbol{\sigma}_j && (a)\\
\text{s.t. } &\boldsymbol{m}_j = \boldsymbol{m}_{j-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_j,\ j = 1, 2, \ldots, N_r + 1 && (b)\\
&\sum_{\gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}} (\alpha_j(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}} min\left(\alpha_j(\gamma), 0\right), \\
&\qquad\qquad \forall \varphi_j && (c)\\
&N_r \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}, \forall \gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}} && (d)\\
&x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}, \forall \gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}} && (e)\\
&\boldsymbol{\eta} \cdot \boldsymbol{\sigma}_j = 0,\ j = 1, 2, \ldots, N_r + 1 && (f)\\
&\boldsymbol{Post} \cdot \boldsymbol{\sigma}_j + \boldsymbol{m}_{j-1} \leq \mathbf{1},\ j = 1, 2, \ldots, N_r + 1 && (g)\\
&\boldsymbol{m}_j \in \mathbb{R}^{|P|}_{\geq 0},\ j = 1, 2, \ldots, N_r + 1, \\
&\boldsymbol{\sigma}_j \in \mathbb{N}^{|T|}_{\geq 0},\ j = 1, 2, \ldots, N_r + 1, \boldsymbol{x} \in \{0,1\}^{|\mathcal{Y}_i \setminus \bar{\mathcal{Y}}|}.
\end{aligned} \tag{6}$$

The MILP (6) finds a sequence of firing count vectors $\boldsymbol{\sigma}_j$, such that: **(A)** RMPN satisfies the part of formula φ containing regions from $\mathcal{Y}_i$, and **(B)** the robots cannot collide during movement.

The main idea for ensuring **(A)** is to drive the RMPN from $\boldsymbol{m}_0$ to a marking $\boldsymbol{m}_{N_r+1}$ where the team satisfies the part of φ requiring trajectory visits. Later, $\boldsymbol{m}_{N_r+1}$ will be left for satisfying the final requirements from φ. Due to assumption (ii), a marking as $\boldsymbol{m}_{N_r+1}$ exists. During the robot movements, the team should avoid any intermediate region that is negated in φ - as in assumption (iii), there is no choice (disjunction) between avoiding such a region or visiting some other region.

For ensuring **(B)** (no collisions), MILP (6) uses a number of N_r intermediate markings between $\boldsymbol{m}_0$ and $\boldsymbol{m}_{N_r+1}$. The robots are required to synchronize in each intermediate marking, i.e., wait for whole team to reach that marking and continue the movement after that. A collision-free movement will be obtained, based on facts given in Remark 3, but for showing this we need to first explain MILP (6):

- The cost function (6a) minimizes a weighted sum of firing count vectors $\boldsymbol{\sigma}_j$. Due to the weights, we force that the last firing count vectors contain as few as possible transitions (even no transitions), the idea being to reduce the number of synchronizations. That is, if a vector $\boldsymbol{\sigma}_j$ results empty, then $\boldsymbol{m}_{j-1} = \boldsymbol{m}_j$, meaning that there will be less than N_r markings to synchronise in. Also, due to the cost function, there will be no unnecessary transitions fired.
- The set of constraints (6b) corresponds to eqn. (1).
- The constraints (6c) corresponds to the formula that should be satisfied along trajectory, by imposing conditions for variables $\boldsymbol{x}$ such that the intermediate team requirements are true. Note that we only need a number of $|\mathcal{Y}_i \setminus \bar{\mathcal{Y}}|$ variables in $\boldsymbol{x}$, because the regions to be avoided ($\bar{\mathcal{Y}}$) will be handled by constraint (6f).
- The constraints (6d) and (6e) impose the value for the reached marking $\boldsymbol{m}_{N_r+1}$, in accordance to value of $\boldsymbol{x}$ that satisfies the Boolean formula. Basically, if a region γ is visited in $\boldsymbol{m}_{N_r+1}$, than $\boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}$ can take any value from 1 to the number of robots N_r (depending on how many robots/tokens arrive in cells from region γ). Correspondingly, x_γ should be one due to visiting γ and this is imposed by inequality $N_r \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}$. Contrary, if γ is not visited in $\boldsymbol{m}_{N_r+1}$, then $\boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1} = 0$ and the value of x_γ is set to 0 by inequality $x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}$.
- The constraint (6f) imposes that the RMPN fires no transition that would lead to a place whose output is in $\bar{\mathcal{Y}}$ (regions that should be avoided). For this, vector $\boldsymbol{\eta}$ is necessary, since the product $\boldsymbol{\eta} \cdot \boldsymbol{\sigma}_j$ equals the total number of firings of those unpermitted transitions, during evolution from $\boldsymbol{m}_{j-1}$ to $\boldsymbol{m}_j$.
- The set of inequalities (6g) enforces that each place (cell) is crossed by at most one robot during successive markings $\boldsymbol{m}_{j-1}$ to $\boldsymbol{m}_j$, $\forall j = 1, \ldots, N_r + 1$. This is because the vector $\boldsymbol{Post} \cdot \boldsymbol{\sigma}_j + \boldsymbol{m}_{j-1}$ (with size $|P|$) contains on every position the number of total visits through each place of $\mathcal{Q}$, during movement from $\boldsymbol{m}_{j-1}$ to $\boldsymbol{m}_j$. Equivalently, these inequalities limit each cell as having a capacity of containing at most one robot during two successive markings. We use this requirement to avoid collisions, since the RMPN abstraction does not capture time intervals spent by each robot in each cell. Note that assumption (iv) from Section 2 is necessary because otherwise inequality (6g) would be violated by a $\boldsymbol{m}_0$ containing more tokens in the same place.
- The MILP variables are $\boldsymbol{m}_j$, $\boldsymbol{\sigma}_j$, and $\boldsymbol{x}$, with the indicated sizes. Once some values for these variables are found through optimization, we have the guarantee that $\boldsymbol{m}_{N_r+1}$ satisfies the intermediate requirements of formula, and each $\boldsymbol{\sigma}_j$ is a feasible firing count vector.

The sequence of firing count vectors $\boldsymbol{\sigma}_j$ obtained from MILP (6) is easily transformed to a sequence of firings, i.e., movements of each robot Mahulea et al. (2020).

Remark 3. Solution of MILP (6) yields a collision-free movement. This is because of the following aspects:

- The robots synchronize in each marking $\boldsymbol{m}_j$, $j = 1, \ldots, N_r + 1$, i.e., the team definitely reaches the deployment given by $\boldsymbol{m}_j$ and after that the robots can continue to move.
- From $\boldsymbol{m}_{j-1}$ until $\boldsymbol{m}_j$, each place of $\mathcal{Q}$ is visited by at most one robot or token, as imposed by the sixth set of constraints from (6). Together with the above synchronization, this implies that along all movements from $\boldsymbol{m}_0$ up to $\boldsymbol{m}_{N_r+1}$, there is not a single moment in which two or more robots could be in the same region.
- Since the robots are identical and the cost function of (6) reduces the number of firings, two robots cannot exchange their positions (cells), because such a swapping would increase the cost function without any benefit in terms of satisfying the constraints. Thus, collisions cannot occur in traversed cells, nor when crossing the edges between adjacent cells.

MILP (6) can be solved by using existing software tools as GLPK, CPLEX - Makhorin (2012); IBM (2016). We note that the number of N_r intermediate markings between $\boldsymbol{m}_0$ and $\boldsymbol{m}_{N_r+1}$ is sufficient, i.e. MILP (6) cannot be infeasible due to a too small number. A quick explanation is that in the worst case, each intermediate firing count vector would drive only one robot from its position in $\boldsymbol{m}_0$ to its position to be reached in $\boldsymbol{m}_{N_r+1}$. Thus, under assumptions (i)-(iv), if MILP (6) has no solution, it means that the requirement on trajectory cannot be satisfied, e.g., the only way to reach some regions to be visited would be by crossing some negated propositions. Therefore, the formulation given by MILP (6) is complete, since a solution is found whenever it is possible to satisfy the intermediate requirements. Furthermore, the number of synchronizations can be smaller than $N_r + 1$, if some vectors $\boldsymbol{\sigma}_j$ are null in solution of (6).

Solution for final requirements. After reaching the marking $\boldsymbol{m}_{N_r+1}$ at which the trajectory part of φ is satisfied, the robots should go to some stopping positions where they satisfy the part of φ given on $\mathcal{Y}_f$. For this, we extend MILP (6) to MILP (7), in which the robots start from marking $\boldsymbol{m}_{N_r+1}$ (returned by MILP (6)). For simplicity of notations, let us further denote by $\boldsymbol{m}_0$ the value $\boldsymbol{m}_{N_r+1}$ obtained from (6) (as being a new initial marking). Now, the idea is to reach a final marking $\boldsymbol{m}_{N_r+2}$ where the final requirements are true. We now need a number of $N_r + 1$ intermediate markings (with one extra than in MILP (6)). We have to make sure that the RMPN places reached between $\boldsymbol{m}_0$ and $\boldsymbol{m}_{N_r+2}$ do not violate the intermediate negated regions, i.e., no region from $\bar{\mathcal{Y}}$ is visited. Note that it is possible that a region avoided

along trajectory should be visited in the final positions (e.g., a formula as $\varphi = \neg Y_1 \wedge y_1$). For coping with this, we need the extra marking $\boldsymbol{m}_{N_r+1}$ right before $\boldsymbol{m}_{N_r+2}$, such that from $\boldsymbol{m}_{N_r+1}$ to $\boldsymbol{m}_{N_r+2}$ each robot performs at most one movement.

$$
\begin{aligned}
&\min \mathbf{1}^T \cdot \sum_{j=1}^{N_r+2} j \cdot \boldsymbol{\sigma}_j && (a)\\
\text{s.t. } &\boldsymbol{m}_j = \boldsymbol{m}_{j-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_j, j = 1, 2, \ldots, N_r + 2, && (b)\\
&\sum_{\gamma \in \mathcal{Y}_f} (\alpha_j(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{Y}_f} min\left(\alpha_j(\gamma), 0\right), \forall \varphi_j && (c)\\
&N_r \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+2}, \forall \gamma \in \mathcal{Y}_f && (d)\\
&x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+2}, \forall \gamma \in \mathcal{Y}_f && (e)\\
&\boldsymbol{\eta} \cdot \boldsymbol{\sigma}_j = 0,\ j = 1, 2, \ldots, N_r + 1 && (f)\\
&\boldsymbol{Post} \cdot \boldsymbol{\sigma}_j + \boldsymbol{m}_{j-1} \leq \mathbf{1},\ j = 1, 2, \ldots, N_r + 2 && (g)\\
&\boldsymbol{m}_{N_r+1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_{N_r+2} \geq \mathbf{0} && (h)\\
&\boldsymbol{m}_j \in \mathbb{R}^{|P|}_{\geq 0},\ j = 1, 2, \ldots, N_r + 2,\\
&\boldsymbol{\sigma}_j \in \mathbb{N}^{|T|}_{\geq 0},\ j = 1, 2, \ldots, N_r + 2, \boldsymbol{x} \in \{0, 1\}^{|\mathcal{Y}_f|}. && (7)
\end{aligned}
$$

We next include brief explanations of formulation (7):

- The cost function (7a) mimics the one of (6a), by having the same idea but one extra intermediate marking.
- The sets of constraints (b) to (g) exactly mimic the constraints from MILP (6), with the justifications given there. Of course, we use the set $\mathcal{Y}_f$ for regions captured by final requirements, while regions from $\bar{\mathcal{Y}}$ are still avoided until $\boldsymbol{m}_{N_r+1}$ due to equalities involving $\boldsymbol{\eta}$. As explained before MILP (7), $\boldsymbol{m}_{N_r+2}$ should not avoid those intermediate regions.
- The constraint (7h) imposes that each robot takes at most one transition between $\boldsymbol{m}_{N_r+1}$ and $\boldsymbol{m}_{N_r+2}$. This enables formulas that require some regions from $\bar{\mathcal{Y}}$ to become true in final state.
- MILP (7) has more unknowns than (6): an extra marking and firing vector, and more binary variables.

MILP (7) provides a complete method for satisfying the requirements on final positions. As before, the firing count vectors $\boldsymbol{\sigma}_j$ given by (7) are mapped to sequences of robot movements. Again, the robots should synchronize in each intermediate marking returned by MILP (7), before executing transitions given by $\boldsymbol{\sigma}_j$. Otherwise, the unitary capacity of places could be violated. As for MILP (6), some synchronizations may not appear, indicated by some null vectors $\boldsymbol{\sigma}_j$.

Resolution Algorithm. The overall solution based on MILPs (6) and (7) is captured in Alg. 1, which provides the pseudo-code for the planning strategy. Note that the procedure begins by testing the assumptions from Sec. 2. These tests are easy (according to the given assumptions) and if any of them fails we use the more general solution from Mahulea and Kloetzer (2018). However, that approach does not always enforce collision-free plans, since it was developed for more general formulas and scenarios. Recall that MILP (6) gives the robot movement plans up to a marking that fulfills the intermediate requirements on visiting regions along trajectories. Similarly, MILP (7) continues the movement plans up to a marking where robots stop and fulfill the final requirements. As explained, the synchronizations in intermediate markings are needed in order to ensure a collision-free movement.

Algorithm 1. Solution pseudo-code

Input: RMPN $\mathcal{Q}$, set $\mathcal{Y}$, formula φ, N_r
Output: Robot movement strategies
if *Any assumption (i)-(iv) from Sec. 2 is False* **then**
  Use the more complex and general method from Mahulea and Kloetzer (2018) and Return solution;
Solve MILP (6);
Construct movement plans from non-empty $\boldsymbol{\sigma}_j$, $j = 1, \ldots, N_r + 1$;
Robots synchronize in each marking $\boldsymbol{m}_j$ for which $\boldsymbol{\sigma}_j \neq \mathbf{0}$, $j = 1, \ldots, N_r + 1$;
Set $\boldsymbol{m}_0 := \boldsymbol{m}_{N_r+1}$, to use in MILP (7);
Solve MILP (7);
Robots further move based on non-empty $\boldsymbol{\sigma}_j$, $j = 1, \ldots, N_r + 2$;
Robots synchronize in each marking $\boldsymbol{m}_j$ for which $\boldsymbol{\sigma}_j \neq \mathbf{0}$, $j = 1, \ldots, N_r + 2$;

Since methods (6) and (7) are complete, the problem is infeasible if there is no solution for (6) or (7). In terms of number of unknowns, MILP (6) has $(N_r + 1) \cdot |P|$ real variables, $(N_r + 1) \cdot |T|$ integer variables, and $|\mathcal{Y} \setminus \bar{\mathcal{Y}}|$ binary variables. MILP (7) has $(N_r + 2) \cdot |P|$ real variables, $(N_r + 2) \times |T|$ integer variables, and $|\mathcal{Y}|$ binary variables.

4. SIMULATION RESULTS

Let us consider again the environment in Fig. 1 (from Ex. 1 and Ex. 2), but now assuming all 10 robots initially deployed in the left cells.

Case 1. Reachability of the regions at the middle. Let us first assume the following Boolean formula:

$$\varphi_1 = \bigwedge_{i=1}^{10} y_i,$$

meaning that the regions at the middle of the environments should be reached at the final state.

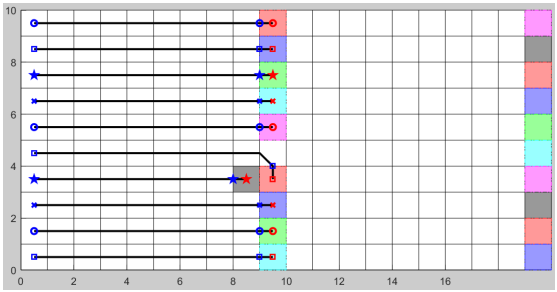
The simulation data is shown in Tab. 1 and it can be observed that the solution proposed in this paper is slower than the one presented in Mahulea and Kloetzer (2018) (TAC2018), mainly because of the number of intermediate markings that is bigger. For TAC2018 approach, a number of $k = 9$ intermediate markings was chosen (with smaller k the problem is infeasible) - as mentioned, in TAC2018 intermediate markings have different meaning than here, referring to at most one movement of each robot between two markings. For the approach proposed here, intermediate markings are imposed by $N_r = 10$ for both MILPs. But, for TAC2018 approach a collision between two robots could occur in the grey region in the middle of the environment, as shown in Fig. 3(b). Fig. 3(a) shows the trajectories obtained by using the approach in this paper.

Case 2. Reachability of the regions at the right by avoiding middle regions. Let us now consider the following Boolean specification:

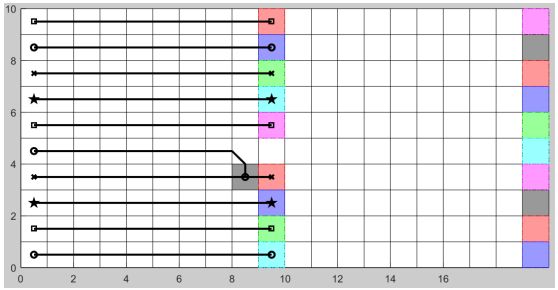
$$\varphi_2 = \left(\bigwedge_{i=1}^{10} \neg Y_i\right) \wedge \left(\bigwedge_{i=11}^{20} y_i\right),$$

	Approaches				
	Alg. 1 (proposed in this paper)		**TAC2018**		
	# variables (MILPs (6) - (7))	optimization time [s]	k	# variables	optimization time [s]
Case 1 - φ_1	10,360 - 11,300	0.78	9	8,501	0.19
Case 2 - φ_2	10,360 - 11,300	0,93	24	22,601	1.72
Case 3 - φ_3	10,360 - 11,300	1.08	36	33,881	6.5

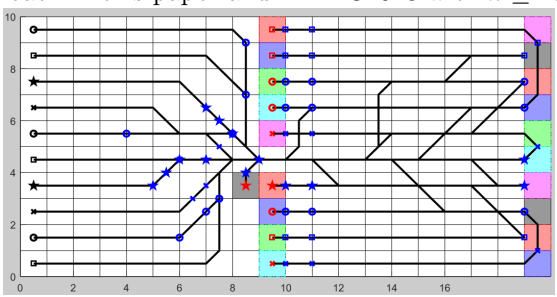
Table 1. Numerical data for simulations by using the current and the TAC2018 approaches; simulations were run on a laptop with i7 (8^{th} generation) CPU, 16 GB RAM.



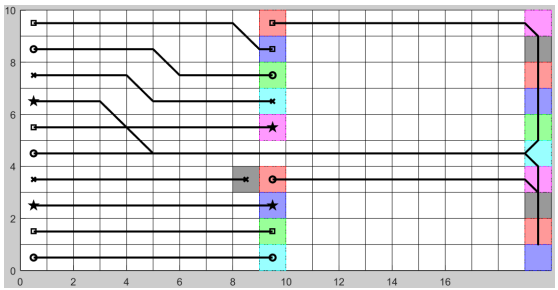
(a) Approach in this paper and in TAC2018 with $k \geq 10$ for φ_1.



(b) Approach in TAC2018 with $k = 9$ for φ_1.

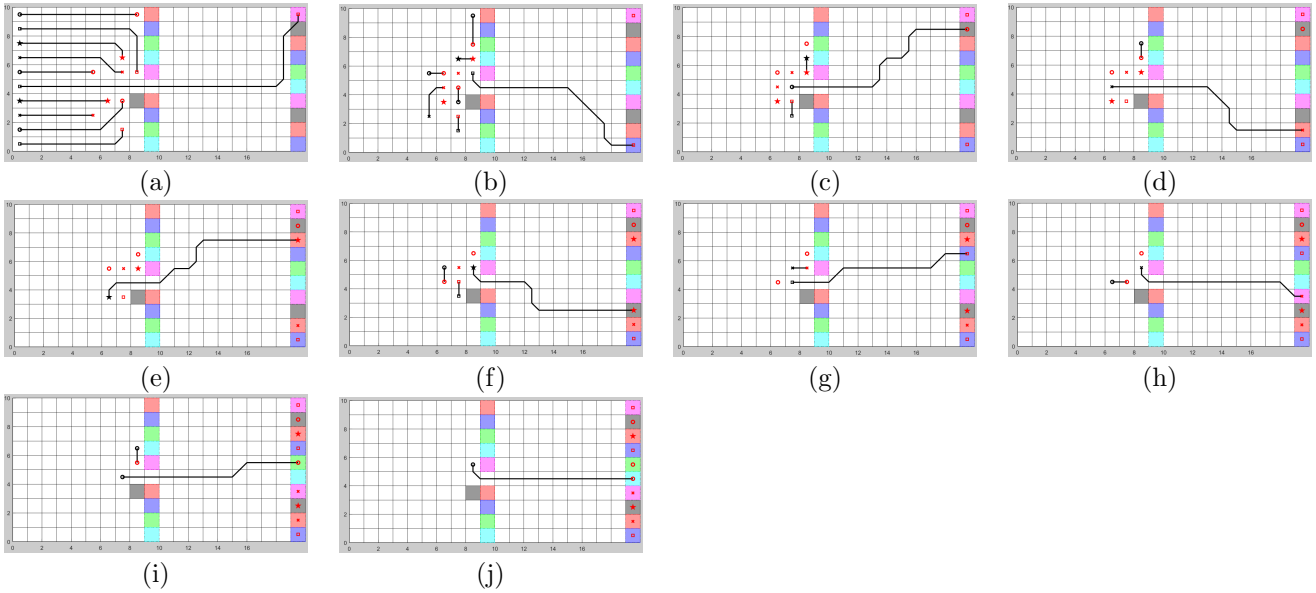


(c) Approach in this paper for φ_3.



(d) Approach in TAC2018 with $k = 36$ for φ_3.

Fig. 3. Trajectories for different simulation experiments.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Fig. 4. Synchronization positions for specification φ_2 using the approach presented in this paper.

meaning that the robots should reach the regions at the right by avoiding the regions in the middle. The complexity data regarding simulation is given in Tab. 1. For the approach presented in this paper, a solution is obtained in less than one second, ensuring the collisions avoidance by using 10 intermediate synchronization points shown in Fig. 4. The approach in TAC2018 returns a solution if the number of intermediate markings is greater than 24, case in which the optimization problem has more variables and the solution is obtained in 1.72 seconds. In the case of TAC2018 approach the intermediate markings of the solution have places with more than one robot, meaning that the collisions could appear.

Case 3. Reachability of the right regions and finally of the middle regions. We now consider the following new formula:

$$\varphi_3 = \left(\bigwedge_{i=1}^{10} \neg Y_i\right) \wedge \left(\bigwedge_{i=11}^{20} Y_i\right) \wedge \left(\bigwedge_{i=1}^{10} y_i\right),$$

meaning that during the trajectories the robots should avoid the regions at the middle of the environment, should reach the regions at the right part, and should finally stop in the regions in the middle.

For the approach presented in this paper, the solution is obtained in about one second. Notice that both optimization problems have the same numbers of variables as in the previous cases. However, there is necessary one more synchronization point versus the previous case, just before entering the final state. The numerical results of the simulations are given in Tab. 1, while the trajectories are given in Fig. 3(c),(d). In particular, Fig. 3(c) shows the trajectories obtained for the approach in this paper, while Fig. 3(d) gives the trajectories with the approach in TAC2018. The TAC2018 solution drives two robots to visit all the intermediate regions (which is impossible with the current approach, due to assumption (ii)), but collisions are possible.

In subsequent works, we plan to exemplify the presented planning methods through real-time experiments in the laboratory setup reported in Kloetzer et al. (2019).

5. CONCLUSION

The work focuses on planning the movement of a team of robots such that a Boolean formula over a set of regions of interest is satisfied. The formula includes requirements both along trajectories and on final positions, and a discrete-event system model is assumed for the team. The solution starts with some additional assumptions than in our previous work Mahulea and Kloetzer (2018), and it is based on formulating two MILP sub-problems that give movement plans for the robotic team. Although the formula expressivity is limited compared with Mahulea and Kloetzer (2018), the current approach gives collision-free plans. Comparisons between the two approaches are included, together with simulations in RMTool. Further work will investigate scalability with the number of robots, which influences the intermediate number of markings of proposed MILPs.

REFERENCES

Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G. (2007). Symbolic Planning and Control of Robot Motion. *IEEE Robotics and Automation Magazine*, 14(1), 61–71.

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

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston.

Ding, X., Smith, S.L., Belta, C., and Rus, D. (2014). Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5), 1244–1257.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343–352.

Habets, L.C.G.J.M., Collins, P.J., and van Schuppen, J.H. (2006). Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51, 938–948.

IBM (2016). IBM ILOG CPLEX Optimization Studio. Software. https://www.ibm.com/products/ilog-cplex-optimization-studio/.

King, J., Pretty, R., and Gosine, R. (2003). Coordinated execution of tasks in a multiagent environment. *IEEE Trans. Systems, Man and Cybernetics*, 33(5), 615–619.

Kloetzer, M., Burlacu, A., Enescu, G., Caraiman, S., and Mahulea, C. (2019). Optimal indoor goods delivery using drones. In *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 1579–1582.

Kloetzer, M. and Mahulea, C. (2015). LTL-based planning in environments with probabilistic observations. *Trans. on Aut. Science and Engineering*, 12(4), 1407–1420.

Kloetzer, M. and Mahulea, C. (2020). Path planning for robotic teams based on LTL specifications and Petri net models. *Discrete Event Dynamic Systems*, 30, 55–79.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6), 1370–1381.

Lacerda, B. and Lima, P.U. (2019). Petri net based multi-robot task coordination from temporal logic specifications. *Robotics and Autonomous Systems*, 122, 343–352.

Mahulea, C. and Kloetzer, M. (2018). Robot Planning based on Boolean Specifications using Petri Net Models. *IEEE Trans. on Automatic Control*, 63(7), 2218–2225.

Mahulea, C., Kloetzer, M., and González, R. (2020). *Path Planning of Cooperative Mobile Robots Using Discrete Event Models*. Wiley-IEEE Press.

Makhorin, A. (2012). GNU linear programming kit. http://www.gnu.org/software/glpk/.

Parrilla, L., Mahulea, C., and Kloetzer, M. (2017). Rmtool: recent enhancements. In *IFAC-Papers*, 1, 5824–5830.

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

Silva, M., Teruel, E., and Colom, J.M. (1998). Linear algebraic and linear programming techniques for the analysis of P/T net systems. *Lecture on Petri Nets I: Basic Models*, 1491, 309–373.