This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

Robot Planning based on Boolean Specifications using Petri Net Models

Cristian Mahulea, *Senior Member, IEEE,* and Marius Kloetzer

***Abstract*—In this paper we propose an automated method for planning a team of mobile robots such that a Boolean-based mission is accomplished. The task consists of logical requirements over some regions of interest for the agents' trajectories and for their final states. In other words, we allow combinatorial specifications defining desired final states whose attainment includes visits to, avoidance of, and ending in certain regions. The path planning approach should select such final states that optimize a certain global cost function. In particular, we consider minimum expected traveling distance of the team and reduce congestions. A Petri net (PN) with outputs models the movement capabilities of the team and the regions of interest. The imposed specification is translated to a set of linear restrictions for some binary variables, the robot movement capabilities are formulated as linear constraints on PN markings, and the evaluations of the binary variables are linked with PN markings via linear inequalities. This allows us to solve an Integer Linear Programming problem whose solution yields robotic trajectories satisfying the task.**

***Index Terms*—Discrete Event Systems, Autonomous Robots, Optimization, Petri nets.**

I. INTRODUCTION

A fair amount of research proposes planning algorithms for mobile robots. The motion tasks range from classical single-robot target reachability and obstacle avoidance [2] to high-level missions for a whole team [3], [4]. Many approaches reduce the robot interaction with the environment into finite representations, and then reason on the obtained discrete event systems [5], [6], [7], [8], [9], [10].

In general, the robot model that is used for the discrete abstraction is based on *transition systems* or *Markov decision processes*, i.e., a graph based model. The high-level mission is given in general as a Linear Temporal Logic (LTL) formula that is automatically transformed into a *Büchi* or *Rabin* automaton. By doing the synchronous product of the team model with the Büchi or Rabin automaton the robot trajectories can be computed by using a shortest path algorithm on the graph which has polynomial time complexity. However, if the number of robots in the team is increased, the number of states of the team model is also increased, being necessary to perform synchronous product of different transition systems as in [10] or duplicate the automatons of robots as in [8].

C. Mahulea is with the Aragón Institute of Engineering Research (I3A), University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain {cmahulea@unizar.es}.

M. Kloetzer is with the Dept. of Automatic Control and Applied Informatics, Technical University "Gheorghe Asachi" of Iasi, Romania {kmarius@ac.tuiasi.ro}.

This work has been partially supported by CICYT - FEDER (Spain-EU) under Grants DPI2014-57252-R and DPI2017-88233-R, by the Aragonese Government (Spain) under grant T27 (GISED group), and by PN-III-P1-1.1-TE-2016-0737 grant.

This work is based on our preliminary results from [1].

To overcome the state-space explosion problem for the team model, in [9] we used a Petri net (PN) to model the team of robots. These models are scalable with the size of the team under the assumption that the robots are identical. If one wants to add one more robot to the team, the structure of the PN model is not changing, being necessary only to add a new token. Moreover. the PN models can be used to study other properties of the system related to task planning, plan execution and plan analysis. In particular, for plan analysis, properties such as boundedness and liveness of Petri nets correspond to checking if resources' usage is stable and plans have no deadlocks [11].

In this paper we propose the problem of planning a team of identical robots such that a Boolean-based specification over some regions of interest is accomplished. The robotic environment is known and static, while the specification imposes Boolean requirements on regions visited during the team motion, as well as on the final robot positions. The specification is globally given for the whole team, without allowing specific robot-to-task assignments. For developing a solution, we model the team movement and the satisfaction of regions with a discrete event system in the form of a PN with outputs. Then, we convert the mission into a set of linear inequalities, we link the binary variables from these inequalities with PN markings and we obtain an Integer Linear Programming (ILP) formulation for the initial problem. The solution yields individual robot trajectories optimal in the sense of minimizing a cost function that includes possible congestions and expected traveled distances for robot trajectories that cross through specific waypoints.

The paper is structured as follows. Some related works and their differences with our approach are discussed in Sec. II. Sec. III includes preliminaries, team model construction and definition of the supported specifications, while Sec. IV formulates the problem. The solution is given in Sec. V, by minimizing a weighted cost based on traveled distance and possible congestions. Sec. VI shows some simulation results and Sec. VII provides concluding remarks.

II. RELATED WORKS

Related problems to the one we consider are reported in works as [3], [7], [12]. Although the specifications we consider here are less expressive than Linear Temporal Logic or regular expressions (as in [6], [3]), our solution is completely different and brings advantages especially in terms of computational feasibility. Thus, instead of combining individual robot abstractions and specification automaton into a complex model, the PN model we construct has fixed topology and only the number of tokens varies with the number of robots (similar to models from [9], where simpler reachability tasks were

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

solved). Performed numerical simulations (Sec. VI) show that our solution can solve demanding situations (e.g. with 10 robots), while such scenarios are not computationally suitable for approaches based on parallel compositions of individual robot models and task automaton (as [3], [10]), due to the state-space explosion problems. Due to the assumed specification, the robots can individually follow their trajectories, without having to synchronize as it is necessary in the case of more complex specification formalisms [3], [13]. Moreover, some methods for high level planning of mobile agents assume individual specifications for each robot [13], [14], whereas the current approach falls in the class of problems that impose a global specification for the whole team, without any specific priori assignment for agents.

Other related discrete path planning solutions are presented in [7], [8], where the authors assume tasks combining Boolean variables on the graph nodes and define a new language as an extension of the Generalized Traveling Salesman Problem for final system states, and in [15], [16], [17], where MILP (Mixed ILP) techniques were proposed for solving different planning or allocation problems in a centralized or distributed fashion. With respect to these works, our method allows Boolean specifications also on robot trajectories, not only on the system final states. Also, the PN team model can be used for other analysis purposes, and our solution is based on a mathematical programming that accepts various cost functions.

PN models have been used for modeling and controlling mobile robots in the recent literature [11], [18], [19], [20], [21], [22], [23]. The modeling methodology is distinct and the models have different significance, in our case the environment being partitioned depending on the regions of interest.

Recently, abstractions characteristic to Resource Allocation Systems were used based on finite automata [24], [25], [26], [27] or PNs [28], [29], [30], while methods available for deadlock avoidance have been adapted for enforcing the reachability of desired final states. In this paper we are interested in computing robot trajectories to accomplish Boolean based specifications for the robots by using the PN models, while the collision avoidance is partially solved.

Many works exist in PN literature dealing with verification of Petri nets properties with Boolean or LTL specifications [31], [32], [33]. Even if some structural properties exist, it is usually necessary to explore the reachability space. Our problem is a synthesis problem, not a verification one.

The main contributions of this work consist in defining a PN system that easily handles a whole team of agents and in developing an ILP formulation that embeds the constructed model and the team specification, while allowing to compute robot trajectories. The performed simulations suggest that the method is computationally tractable for complex scenarios. Solution's limitations include the task expressivity, which relies on logical requirements on visiting and avoiding regions along trajectories and in final states, without permitting temporal sequencing. Robot congestions are partially addressed, while collision and deadlock avoidance are briefly commented by pointing to additional steps that can be addressed after a movement plan is obtained. Also, local maneuvers that may be employed in case of possible congestion would increase the travel distances with respect to the nominal distances assumed in the motion planning part.

III. Preliminaries and team model

Sec. III-A defines the discrete event model that we will use for a team of identical robots. Sec. III-B introduces the formalism for expressing mission requirements for a team of robots.

A. Petri net model

This subsection introduces the basic notions of PN.

***Definition** 3.1:* A Petri net (PN) is a 3-tuple $\mathcal{N} = \langle P, T, F\rangle$ with P and T two finite, non-empty and disjoint sets of places and transitions; $F \subseteq (P \times T) \cup (T \times P)$ is the set of direct arcs from places to transitions or transitions to places.

Instead of considering general PN where the arcs can have weights greater than one, in this paper we consider that all arcs are unitary. In the PN literature, this class of PN is called ordinary PNs. Because of this, the PN structure can be represented by two binary matrices: $\boldsymbol{Pre}, \boldsymbol{Post} \in \{0,1\}^{|P|\times|T|}$, with $\boldsymbol{Pre}[p_i,t_j] = 1$ if $\exists(p_i,t_j) \in F$, and $\boldsymbol{Pre}[p_i,t_j] = 0$ otherwise; $\boldsymbol{Post}[p_i,t_j] = 1$ if $\exists(t_j,p_i) \in F$, otherwise $\boldsymbol{Post}[p_i,t_j] = 0$. [1]

For $x \in P \cup T$, the sets of its input and output nodes (places or transitions) are denoted as ${}^\bullet x$ and $x^\bullet$, respectively. Let p_i, $i = 1,\dots,|P|$ and t_j, $j = 1,\dots,|T|$ denote the places and transitions. Each place can contain a non-negative integer number of tokens, and this number represents the marking of the place. The distribution of tokens in places is denoted by $\boldsymbol{m}$, where $\boldsymbol{m}[p_i]$ is the marking of place p_i. The initial token distribution, denoted by $\boldsymbol{m}_0 \in \mathbb{N}_{\geq 0}^{|P|}$, is called the initial marking of the net system. A PN with an initial marking is a PN system $\langle \mathcal{N}, \boldsymbol{m}_0\rangle$.

Because the PN is ordinary, a transition $t_j \in T$ is enabled at $\boldsymbol{m}$ if all its input places contain at least one token, i.e., $\forall p_i \in {}^\bullet t_j, \boldsymbol{m}[p_i] \geq 1$. An enabled transition t_j can fire leading to a new state $\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C}[\cdot,t_j]$, where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the token flow matrix and $\boldsymbol{C}[\cdot,t_j]$ is the column corresponding to t_j. It will be said that $\tilde{\boldsymbol{m}}$ is a reachable marking that has been reached from $\boldsymbol{m}$ by firing t_j and it is written as $\boldsymbol{m}[t_j\rangle\tilde{\boldsymbol{m}}$.

If $\tilde{\boldsymbol{m}}$ is reachable from $\boldsymbol{m}$ through a finite sequence of transitions $\sigma = t_{i_1}t_{i_2}\dots t_{i_k}$, the following state (or fundamental) equation is satisfied:

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

where $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ is the firing count vector, i.e., its j^{th} element is the number of times transition t_j appears in sequence σ. Notice that Eq. (1) is only a necessary condition for the reachability of a marking. The marking solutions of (1) that are not reachable are called *spurious markings*. In general, checking if a marking $\boldsymbol{m}$ is reachable or not is not an easy problem due to these spurious markings.

A PN with each transition having at most one input and at most one output place is called *state machine*. Formally,

[1] Throughout this paper, instead of using integers to reference elements of matrices or vectors, we use symbolic variables which reference the element corresponding to the used symbol.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

a PN is state machine if $|^{\bullet}t| \le 1$ and $|t^{\bullet}| \le 1$, $\forall t \in T$. A PN is called live if from any reachable marking any transition can eventually fire (possibly after first firing other transitions). It is well known that for state machine PNs, liveness is equivalent to strongly connectedness and non-emptiness of (initial) marking. Moreover, in a live state machine, there exist no spurious markings [34], i.e., the solutions of the fundamental Eq. (1) give the set of reachable markings.

We will use the PN to model a team of identical robots evolving in an environment where some convex polygonal regions of interest exist. The regions of interest are labeled with elements from set $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Pi|}\}$. For this reason, we define a class of Petri nets with outputs, which is a restrictive class of Interpreted Petri nets, without inputs associated to transitions.

***Definition** 3.2:* A Petri net $\mathcal{Q}$ with outputs is a 4-tuple $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, \Pi, h \rangle$, where:

- $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is a Petri net system;
- $\Pi \cup \{\emptyset\}$ is the output alphabet (set containing the possible output symbols (observations)), where $\emptyset$ denotes the empty observation;
- $h : P \to 2^{\Pi}$ is an observation map, where $h(p_i)$ yields the output of place $p_i \in P$. If p_i has at least one token, then observations from $h(p_i)$ are active.

Let $\boldsymbol{v}_{\Pi_i} \in \{0,1\}^{1 \times |P|}$ be the characteristic row vector of the observation $\Pi_i \in \Pi$ such that $\boldsymbol{v}_{\Pi_i}[p_k] = 1$ if $\Pi_i \in h(p_k)$ and $\boldsymbol{v}_{\Pi_i}[p_k] = 0$ otherwise. It is easy to observe that, for a reachable marking $\boldsymbol{m}$, if the product $\boldsymbol{v}_{\Pi_i} \cdot \boldsymbol{m} > 0$ then the observation Π_i is active at $\boldsymbol{m}$. Let $\boldsymbol{V} \in \{0,1\}^{|\Pi| \times |P|}$ be the matrix formed by the characteristic vectors of all observations, i.e, the first row is the characteristic vector of Π_1, etc. The product $\boldsymbol{V} \cdot \boldsymbol{m}$ is a column vector of dimension $|\Pi|$ where the i^{th} element is non-zero if observation Π_i is active. We denote by $||\boldsymbol{V} \cdot \boldsymbol{m}||$ the set of outputs corresponding to non-zero elements of $\boldsymbol{V} \cdot \boldsymbol{m}$, i.e. $||\boldsymbol{V} \cdot \boldsymbol{m}||$ is the set of active observations (element of 2^{Π}) at marking $\boldsymbol{m}$.

A *run* (or *trajectory*) of $\mathcal{Q}$ is a *finite sequence* $r = \boldsymbol{m}_0[t_{j_1}\rangle \boldsymbol{m}_1[t_{j_2}\rangle \boldsymbol{m}_2[t_{j_3} \ldots t_{j_{|r|}}\rangle \boldsymbol{m}_{|r|}$ that induces an output *word* denoted by $h(r)$, which is the observed sequence of elements from 2^{Π}, i.e., $h(r) = ||\boldsymbol{V} \cdot \boldsymbol{m}_0||, ||\boldsymbol{V} \cdot \boldsymbol{m}_1||, \ldots, ||\boldsymbol{V} \cdot \boldsymbol{m}_{|r|}||$, $h(r) \in \left(2^{\Pi}\right)^*$, where $\left(2^{\Pi}\right)^*$ is the Kleene closure of set 2^{Π}.

Team model. The above PN with outputs can model the movement capabilities of a team of identical mobile robots in a partitioned environment cluttered with overlapping and static regions of interest denoted by elements of set Π. Such finite abstractions can be constructed based on partitions yielded by cell decompositions [35] and control laws for specific robot dynamics [36], [37]. The main idea is that the environment is partitioned based on regions of interest, every place of $\mathcal{N}$ corresponds to a partition cell, while transitions of PN correspond to robot's movement capabilities between adjacent cells. The satisfaction map h shows the regions from Π that are satisfied (visited) when the robots are inside particular cells, with empty observation corresponding to partition cells that are not included in any region from Π. The number of tokens of the PN model is equal with the number of robots, and the initial marking is given by the cells initially occupied by the team. Thus, adding a robot in the team implies adding a token to a place, without changing the PN structure.

We further assume that the model $\mathcal{Q}$ for robots evolving in an environment is already available. The informal steps that lead to its construction are captured in Alg. 1. For polygonal regions of interest, multiple cell decomposition techniques can be used in line 1 [35], our approach not being tailored for a specific one. The transitions added on lines 4-7 assume fully-actuated point robots, which can move from the current cell to any adjacent cell, by straight movement to the middle point of the line segment shared by the two cells. Alg. 1 also returns the vector $\boldsymbol{w} \in \mathbb{R}_{\ge 0}^{|T|}$ that contains the average distance for traveling between adjacent cells. Due to the polytopal cells and the mentioned piece-wise straight movements of robots, the expected distance for moving from cell p_i to p_j is chosen, on line 9, as being the average of distances between the exit point (middle of line segment shared by p_i and p_j) and any possible entry point (middle of line segments shared by p_i with all other neighboring cells). For different robot dynamics, the condition from line 6 can be replaced with the existence of control laws steering the robot from cell p_i to adjacent cell p_j in finite time, e.g., works as [36], [37] describe the case of affine or multi-affine dynamics in polytopal or rectangular environments. Line 11 adds the tokens, based on robots' initial positions. The observation map from line 12 is well-defined, since the referred cell decomposition techniques preserve boundaries and intersections of regions from Π.

Algorithm 1: Construct the PN system $\mathcal{Q}$

Input: Environment, regions Π, initial team deployment
Output: Team model $\mathcal{Q}$
1 Construct a cell decomposition of the environment based on the polygonal regions of interest from Π;
2 Associate each cell from decomposition to a place from P; let $P = \{p_1, p_2, \ldots, p_{|P|}\}$;
3 Let $T = \emptyset$, $F = \emptyset$, $\boldsymbol{w} = \boldsymbol{0}$;
4 **for** $p_i \in P$ **do**
5   **for** $p_j \in P$, $p_i \neq p_j$ **do**
6   **if** cells p_i and p_j are adjacent **then**
7   Add transition $t_{i,j}$ to T;
8             $F := F \cup \{(p_i, t_{i,j}), (t_{i,j}, p_j)\}$;
9             $\boldsymbol{w}[t_{i,j}]$ = average distance traveled by a robot that moves from cell p_i to p_j;
10 **for** $p_i \in P$ **do**
11     $\boldsymbol{m}_0[p_i]$ = no. of robots initially deployed in cell p_i;
12     $h(p_i) = \{\Pi_j \in \Pi | \text{cell } p_i \text{ is included in region } \Pi_j\}$;

***Remark** 3.3:* The construction from Alg. 1 ensures that the obtained PN is a state machine.

This result holds because all transitions added in step 7 of Alg. 1 have a single input and a single output arc. If at least one robot is deployed in the environment, the PN system is live and no blocking situation will appear during robot motion.

B. Boolean-based specifications

Assume the finite set of atomic propositions $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Pi|}\}$, where in a robot-inspired scenario, Π_i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

labels a specific region of interest from the environment.

Syntactically, we assume requirements expressed as Boolean logic formulae defined over the set of variables $\mathcal{P} = \mathcal{P}_t \cup \mathcal{P}_f$, where $\mathcal{P}_t = \Pi$ and $\mathcal{P}_f = \{\pi_1, \pi_2, \ldots, \pi_{|\Pi|}\}$, by using the standard logical connectors $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction). The sets $\mathcal{P}_t$ and $\mathcal{P}_f$ refer to the same regions of interest, but the elements of $\mathcal{P}_t$ suggest regions that should be visited (or avoided, when negated) along a trajectory, while $\mathcal{P}_f$ suggests regions that should be visited (or avoided) in the last state of a run, as explained in the following semantics.

The specifications are interpreted over finite words over the set 2^{Π}, as are those generated by the PN system with outputs $\mathcal{Q}$ from Def. 3.2. Semantically, the lower- and upper-case notations from the above set $\mathcal{P}$ have the following meaning when interpreted over the word generated by a run $r = \boldsymbol{m}_0[t_{j_1}\rangle \boldsymbol{m}_1[t_{j_2}\rangle \boldsymbol{m}_2[t_{j_3} \ldots t_{j_{|r|}}\rangle \boldsymbol{m}_{|r|}$:

- $\Pi_i \in \mathcal{P}_t$ evaluates to *True* over word $h(r)$ if and only if $\exists j \in \{0, 1, \ldots, |r|\}$ such that $\Pi_i \in ||\boldsymbol{V} \cdot \boldsymbol{m}_j||$;
- $\pi_i \in \mathcal{P}_f$ evaluates to *True* over word $h(r)$ if and only if $\Pi_i \in ||\boldsymbol{V} \cdot \boldsymbol{m}_{|r|}||$.

In other words, an upper-case variable refers to a proposition that is evaluated along the whole run, while a lower-case one refers only to the final (terminal) marking. Under this explanation, the formal definitions of syntax and semantics of used specifications are not included, and can be found in any study including Boolean formulae [38]. From now on, we will assume that any Boolean-based requirement φ is expressed into a Conjunctive Normal Form (CNF), the conversion into such a form being possible for any logical expression [38],[39].

For example, a specification for mobile robots as $\varphi = (\Pi_1 \vee \Pi_2) \wedge \neg\pi_1 \wedge \neg\Pi_3$ requires that either region Π_1 or Π_2 is visited along the run, Π_3 is always avoided, and region Π_1 is not true (no robot occupies it) in the final state, i.e., when all robots stop. A specification as $\varphi = \Pi_1 \wedge \Pi_2$ requires that regions Π_1 and Π_2 are visited along robot trajectories. An implication formula as $\varphi = \neg\Pi_1 \vee \Pi_2$ is not interpreted in the intuitive sense that a visit to Π_1 implies a further visit to Π_2, but it is interpreted over the entire trajectories (e.g., the task is accomplished if a robot visited Π_2 at a moment, even if Π_1 was visited after). One cannot impose a specific order or simultaneity when visiting Π_1 and Π_2, as it is possible when using more complex specification formalisms or robot-specific tasks [13], [10]. For more than one robot, the specification imposes a global requirement on the attainment or avoidance of regions, without allowing individual requirements as visiting two disjoint regions with the same agent. However, this lack of expressivity, together with the PN model, will yield solutions whose complexity is independent of the number of robots.

IV. Problem definition

The problem we solve is formulated as follows:

Problem. Consider a team of N identical mobile robots evolving in an environment where regions of interest labeled with elements from set Π are defined. Given a Boolean-based specification φ for the team, as in Sec. III-B, plan the robotic motion such that the resulting trajectories satisfy φ.

Assumptions. As stated in Sec. III-A, the team is abstracted into a PN system with outputs $\mathcal{Q}$ having the form from Def. 3.2. Under the natural assumption of a connected environment, the PN model $\mathcal{Q}$ is strongly connected (i.e. $\forall x_i, x_j \in P \cup T$ there exists a path starting in x_i and ending in x_j). Thus, the PN has no spurious markings and the set of its reachable markings can be characterized by the state equation (1).

Let us assume that the requirement φ (expressed in CNF) consists of a conjunction of n terms: $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$. Each term φ_i, $i = 1, \ldots, n$ is a disjunction of n_i variables (negated or not) from set $\mathcal{P}$ from Sec. III-B, having the form $\varphi_i = [\Pi_{j_1} \,|\, \neg\Pi_{j_1}] \vee [\pi_{j_1} \,|\, \neg\pi_{j_1}] \vee [\Pi_{j_2} \,|\, \neg\Pi_{j_2}] \vee [\pi_{j_2} \,|\, \neg\pi_{j_2}] \vee \ldots \vee [\Pi_{j_{n_i}} \,|\, \neg\Pi_{j_{n_i}}] \vee [\pi_{j_{n_i}} \,|\, \neg\pi_{j_{n_i}}]$. In the expression of φ_i, the square brackets "[…]" contain optional appearing terms, while "|" denotes a choice between two variables.

Solution main steps. Our solution begins by converting specification φ into n linear restrictions over a set of $2 \cdot |\Pi|$ binary variables, as described in [1]. Then, links between these binary variables and proposition satisfactions are enforced by using linear inequalities based on the PN model $\mathcal{Q}$. This will yield a solution for our problem based on an ILP formulation and an algorithmic translation of ILP outcome to robot trajectories (sequences of firings in the PN model). The ILP objective function aims to decrease the total distance traveled by robots and the number of possible congestions, when more robots can meet in the same partition cell. For simplicity, we first handle final state requirements, i.e., formulae over $\mathcal{P}_f$ (Sec. V-A), and then we present the general case of trajectory requirements (Sec. V-B). Sec. V-C further discusses the presented solutions. Due to the abstract model and the definition of weighting $\boldsymbol{w}$ from Alg. 1, the optimality from Sec. V does not refer to minimizing the actual traveled distance, but to minimizing a cost function that includes the expected trajectory length.

V. Solution

Vector $\boldsymbol{x} = [x_{\Pi_1}, x_{\Pi_2}, \ldots, x_{\Pi_{|\Pi|}}, x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_{|\Pi|}}]^T \in \{0,1\}^{2\cdot|\Pi|}$ includes the above-mentioned binary variables, with the following interpretation:

- $x_{\Pi_i} = 1$ (or $\boldsymbol{x}[\Pi_i] = 1$) if proposition Π_i evaluates to *True* (i.e., region labeled with Π_i is visited along the team trajectory), and $x_{\Pi_i} = 0$ (or $\boldsymbol{x}[\Pi_i] = 0$) otherwise;
- $x_{\pi_i} = 1$ (or $\boldsymbol{x}[\pi_i] = 1$) if proposition π_i evaluates to *True* (i.e., a robot stops inside the region labeled with Π_i), and $x_{\pi_i} = 0$ (or $\boldsymbol{x}[\pi_i] = 0$) otherwise, $\forall i = 1, \ldots, |\Pi|$.

To construct the mentioned inequalities, for each φ_i, $i = 1, \ldots, n$, we define a function $\alpha_i : \mathcal{P} \to \{-1, 0, 1\}$ showing what variables from $\mathcal{P}$ appear in disjunction φ_i and which of them are negated:

$$\alpha_i(\gamma) = \begin{cases} -1, & \text{if } \neg\gamma \text{ appears in } \varphi_i \\ 0, & \text{if } \gamma \text{ does not appear in } \varphi_i \\ 1, & \text{if } \gamma \text{ appears in } \varphi_i \end{cases}, \forall \gamma \in \mathcal{P} \quad (2)$$

A. Solution for constraints on the final state

When finding a solution for the proposed problem, one can consider various performance measures for the resulting robot movements. In the current formulation, we aim to reduce

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

(a) the total expected distance traveled by agents and (b) the number of situations in which robots can collide. For intention (a), we weight the fired transitions with average distances for moving a robot between two adjacent cells, i.e., we aim to minimize $\boldsymbol{w}^T \cdot \boldsymbol{\sigma}$, with $\boldsymbol{w}$ computed in Alg. 1. For intention (b), we note that, for a given firing count vector $\boldsymbol{\sigma}$, the elements of vector $\boldsymbol{Post} \cdot \boldsymbol{\sigma}$ contain the cumulative number of tokens from each place of PN induced by firings of transitions from $\boldsymbol{\sigma}$. Thus, $\boldsymbol{Post} \cdot \boldsymbol{\sigma}$ gives the number of visits (not necessarily at the same time moment) in partition cells, and by reducing these values we reduce the possibilities of having more robots in the same cell. We combine intentions (a) and (b) as the cost function $\lambda \cdot \boldsymbol{w}^T \cdot \boldsymbol{\sigma} + \mu \cdot \|\boldsymbol{Post} \cdot \boldsymbol{\sigma}\|_\infty$, where λ and μ are design parameters and $\|.\|_\infty$ denotes the maximum norm of a vector. For obtaining a linear cost function, we minimize $\lambda \cdot \boldsymbol{w}^T \cdot \boldsymbol{\sigma} + \mu \cdot b$, where b upper bounds any element of $\boldsymbol{Post} \cdot \boldsymbol{\sigma}$. The above considerations together with the goal of obtaining a final marking at which the formula is satisfied are captured by ILP formulation (3).

$$
\begin{array}{ll}
 & \min \lambda \cdot \boldsymbol{w}^T \cdot \boldsymbol{\sigma} + \mu \cdot b \\
\text{s.t.} & \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \\
 & \sum_{\gamma \in \mathcal{P}_f} (\alpha_i(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{P}_f} min\,(\alpha_i(\gamma), 0)\,, \forall \varphi_i \\
 & N \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}, \forall \gamma \in \mathcal{P}_f \\
 & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}, \forall \gamma \in \mathcal{P}_f \\
 & \boldsymbol{Post} \cdot \boldsymbol{\sigma} \leq b \cdot \boldsymbol{1}^T \\
 & \boldsymbol{m} \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}, \boldsymbol{x} \in \{0\}^{|\Pi|} \times \{0,1\}^{|\Pi|}, b \geq 0
\end{array}
\tag{3}
$$

In (3), $\boldsymbol{v}_\gamma$ is the characteristic vector of $\gamma \in \mathcal{P}_f$, and the first $|\Pi|$ binary variables from $\boldsymbol{x}$ (for trajectory requirements) are set to zero, since specifications from this subsection do not include such constraints. ILP (3) has $(2 \times |\Pi| + n + 2 \times |P|)$ constraints and $(|P| + |T| + |\Pi| + 1)$ unknowns, from which $|\Pi|$ variables are binary.

The second set of constraints from (3) links formula's conjunctions to binary variables for final regions. If the final region γ is not captured in φ_i, then its corresponding binary variable is unconstrained (coefficient $\alpha_i(\gamma)$ is zero). Regions that appear non-negated or negated in disjunction φ_i yield (through (2)) coefficients "+1" or "-1", respectively, in the left-hand term, and the negated regions also decrease the value of the right-hand term. E.g., if $\varphi_i = \pi_1 \vee \pi_2$, at least one of the two regions should be visited such that $x_{\pi_1} + x_{\pi_2} \geq 1$. If $\varphi_i = \neg\gamma$, then x_γ should be 0, i.e. $1 - x_\gamma = 1$, and since x_γ is binary we can write $1 - x_\gamma \geq 1$; the first "1" from here is placed in the right-hand term via function $min\,(\alpha_i(\gamma), 0)$.

The third and fourth constraints from (3) enforce the correct values of binary variables x_{π_i} corresponding to observations in final positions. Recall that N is the number of robots (tokens of $\mathcal{Q}$), and here it can be replaced with any bigger number.

As an alternative cost function for ILP (3), it is possible to minimize the number of transitions (robot movements) along the team trajectory, by choosing the objective function $\boldsymbol{1}^T \cdot \boldsymbol{\sigma}$.

Based on the optimal solution $\boldsymbol{\sigma}$ of (3), the robot (token) trajectories are obtained by firing the enabled transitions and by storing the sequence of places visited by each token. The strategy is given in Alg. 2.

Lemma 5.1: If the optimal solution $\boldsymbol{\sigma}$ of (3) satisfies $\|\boldsymbol{Post} \cdot \boldsymbol{\sigma}\|_\infty = 1$ (that is equivalent to $b = 1$), then there are no collisions possible during robot movements.

Proof: Since $\boldsymbol{Post} \cdot \boldsymbol{\sigma}$ counts the number of tokens in each place corresponding to the firing vector $\boldsymbol{\sigma}$, the hypothesis basically says that each partition cell is visited at most once during team movement. ∎

Note that a path planning problem can be divided into two steps: (a) the first one (tackled by current work) is to compute mission-fulfilling trajectories for the robots (while trying to avoid the congestion); (b) second, having the trajectories, one can try to avoid collisions and deadlocks by adding an additional controller. If $\|\boldsymbol{Post} \cdot \boldsymbol{\sigma}\|_\infty > 1$, congestion can occur in places $p \in P$ for which $(\boldsymbol{Post} \cdot \boldsymbol{\sigma})[p] > 1$, and further steps have to be taken for collision avoidance and deadlock prevention. To this goal, one can try to use specific Petri net models with *capacity* constraints on some places [29] and supervisory control theory of discrete event systems [27], [24], [26], [40], [41]. However, there are no guarantees that a deadlock free movement is possible for any obtained trajectories, and in such cases the procedure for generating trajectories should be altered. The additional strategies for solving the above step (b) go outside the current scope of this paper.

Algorithm 2: Iterative construction of agent strategies

Input: $\langle P, T, \boldsymbol{C} \rangle$, $\boldsymbol{m}_0$, $\boldsymbol{\sigma}$
Output: Robot movement strategies

```
Let m = m_0;
while 1^T · σ > 0 do
    Let t ∈ T s.t. σ[t] > 0 ∧ m[•t] > 0;
    Pick any robot i in •t;
    Assign movement according to t to robot i;
    Let m := m + C[·, t];
    Let σ[t] := σ[t] − 1;
```

Two properties of the PN model for the system considered here are used to guarantee the correctness of the Alg. 2:

- The *PN is a live state machine*, hence all solutions of the state equation (1) are reachable markings. This ensures that the marking $\boldsymbol{m}$ solution of (3) is a reachable marking, i.e., not a spurious one;
- Since $\boldsymbol{w} \geq 0$ (that is a natural assumption being related to distances or energy), the *paths of the robots have no cycles*. This property also ensures that $\boldsymbol{\sigma}$ solution of (3) is not a *'spurious'* vector, i.e., there exists a fireable firing sequence σ with the firing count vector $\boldsymbol{\sigma}$.

B. Solution for general constraints on trajectory and final state

For allowing constraints on final team deployment (set $\mathcal{P}_f$) and on team trajectory (set $\mathcal{P}_t$), the first idea was to include constraints on the firing count vector $\boldsymbol{\sigma}$ in (3). However, due to general constraints, some robot trajectories may necessitate cycles. When solving (3), these cycles would *not be included* in the obtained solutions, i.e., *spurious firing vectors* would appear. This can be observed by considering the state equation corresponding to a reachable marking $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$. Let us assume that $\boldsymbol{\sigma}$ corresponds to a firing sequence σ that contains

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

a cycle, i.e., $\boldsymbol{\sigma} = \boldsymbol{\sigma'} + \boldsymbol{\sigma''}$, with $\boldsymbol{\sigma''}$ the cycle's firing count vector. Since in a state machine PN a T-semiflow is a cycle, this implies that $\boldsymbol{C} \cdot \boldsymbol{\sigma''} = 0$ [34]. Obviously, the cost function of (3) would yield vector $\boldsymbol{\sigma'}$ rather than $\boldsymbol{\sigma}$ as the optimal solution, so the firing sequence σ would not be obtained.

To avoid spurious firing count vectors, we consider a sequence of k markings $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$ such that: $\boldsymbol{m}_1 = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}_1$, $\boldsymbol{m}_0 - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_1 \geq \boldsymbol{0}$; $\boldsymbol{m}_2 = \boldsymbol{m}_1 + \boldsymbol{C} \cdot \boldsymbol{\sigma}_2$, $\boldsymbol{m}_1 - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_2 \geq \boldsymbol{0}$; ... Informally, these constraints enforce that between PN states $\boldsymbol{m}_{i-1}$ and $\boldsymbol{m}_i$ each token moves *at most* through one transition, i.e., each robot advances maximum one cell. This artifice also simplifies the construction of agents' strategies.

Putting together the cost function concept from ILP (3), the PN state equations for the sequence of k markings, and the restrictions concerning the binary variables x_{π_i} and x_{Π_i}, the following optimization problem is obtained:

$$
\begin{array}{ll}
 & \min \lambda \cdot \boldsymbol{w}^T \cdot \sum_{i=1}^{k} \boldsymbol{\sigma}_i \, + \, \mu \cdot b \\
\text{s.t.} & \boldsymbol{m}_i = \boldsymbol{m}_{i-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_i, i = 1, \ldots, k \\
 & \boldsymbol{m}_{i-1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_i \geq 0, i = 1, \ldots, k \\
 & \sum_{\gamma \in \mathcal{P}} \left(\alpha_i(\gamma) \cdot x_\gamma \right) \, \geq \, 1 + \sum_{\gamma \in \mathcal{P}} min \left(\alpha_i(\gamma), 0 \right), \forall \varphi_i \\
 & N \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \forall \gamma \in \mathcal{P}_f \\
 & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \forall \gamma \in \mathcal{P}_f \\
 & N \cdot (k+1) \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^{k} \boldsymbol{m}_i \right), \forall \gamma \in \mathcal{P}_t \\
 & x_\gamma \leq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^{k} \boldsymbol{m}_i \right), \forall \gamma \in \mathcal{P}_t \\
 & \left(\boldsymbol{Post} \cdot \sum_{i=1}^{k} \boldsymbol{\sigma}_i \right) \leq b \cdot \boldsymbol{1}^T \\
 & \boldsymbol{m}_i \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{\sigma}_i \in \mathbb{N}_{\geq 0}^{|T|}, i = 1, \ldots, k \\
 & \boldsymbol{x} \in \{0,1\}^{|\mathcal{P}|}, b \geq 0
\end{array}
\tag{4}
$$

The optimization problem (4) is a standard ILP problem [42], for which there exist complete algorithms for obtaining the optimal solution, e.g., [43]. Its solution $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_k)$ constitutes a sequence of firing count vectors for PN model $\mathcal{Q}$ and it is converted to robot trajectories as follows. For each $\boldsymbol{\sigma}_i$, $i = 1, \ldots, k$, any token moves at most through one transition, and lines 3-5 of Alg. 2 indicate the moving robots.

Summing up the above details, (4) gives a solution for the problem formulated in Sec. IV, while the cost function accounts for the total expected distance traveled by robots and the possible congestions in cells from the partitioned environment. The constraints of (4) ensure the following:

- the correct functioning of model $\mathcal{Q}$ (first two lines with constraints); in total $((2 \times k) \times |P|)$ constraints;
- the satisfaction of formula φ through its disjunctive terms and binary variables (third constraint); in total n constraints;
- the link between binary variables corresponding to the formula and PN markings for the final requirements (constraints 4 and 5; in total $(2 \times |\Pi|)$ constraints) and for the trajectory requirements (constraints 6 and 7; in total $(2 \times |\Pi|)$ constraints),
- upper bound b for elements of vector $\boldsymbol{Post} \cdot \sum_{i=1}^{k} \boldsymbol{\sigma}_i$, for capturing its maximum norm (constraints 8; in total $|P|$ constraints),
- positivity restrictions for unknown variables $\boldsymbol{m}_i$, $\boldsymbol{\sigma}_i$ and b; $(k \times (|P| + |T|) + 1)$ constraints.

***Remark** 5.2:* Instead of considering the second term of cost function from ILP (4), one could completely avoid collisions (rather than reducing congestions) by adding constraints of form $\boldsymbol{\sigma}_k[t_{i,j}] + \boldsymbol{\sigma}_k[t_{j,i}] \leq 1$, $\forall i, j, k$. Such constraints would forbid two robots from adjacent cells to switch positions. However, such a team movement strategy would require synchronizations when robots change cells, in order to exactly follow the order of firings from successive firing count vectors $\boldsymbol{\sigma}_k$.

***Remark** 5.3:* The constant k in ILP (4) is a design parameter giving the maximum number of intermediate discrete states (markings) of each robot. The theoretical upper-bound of k is $|T|$, because in the worst case scenario, a robot has to once follow each transition from PN (e.g., imagine a string-like PN where the "first" and "last" places have different outputs, a robot starts from the "first" place, and the formula requires to satisfy along trajectory the output of the "last" place and to satisfy in the final state the output of the "first" one). However, in practice, much lower values of k suffice. When k is chosen too small, the problem (4) becomes unfeasible. If k is larger than needed, some intermediate firing vectors $\boldsymbol{\sigma}_i$ will become zero in solution of (4).

C. Discussion on the above solutions

Solution to use. When the Boolean-based specification φ contains only symbols from $\mathcal{P}_f$, one should use the solution from Sec. V-A, consisting in ILP (3) and Alg. 2. In this case, the ILP (3) has far less constraints and unknowns than ILP (4).

For a general specification that also includes symbols from $\mathcal{P}_t$, the solution from Sec. V-B (ILP (4)) is to be used. One can start with a fairly low value for k, solve ILP (4) and increase k if the optimization fails to return a solution. The moving strategy for each robot results by concatenating the transitions given by the obtained sequence of firing count vectors.

Robot synchronization. For both above solutions, the obtained trajectory of each robot basically satisfies a part of formula φ, such that the whole team accomplishes task φ. Because φ is a Boolean-based formula as in Sec. III-B, it cannot impose specific orderings or simultaneous visits of regions in Π. Therefore, each robot can individually follow its trajectory, without synchronizing with other team members.

Recalling the limitations of our approach - lack of expressivity for imposing orders when visiting regions, and reducing the possible congestions rather than ensuring a collision-free movement with no deadlocks - we mention that robot synchronization would become necessary for specifications or for movement procedures that try to reduce such conservativeness.

Solution complexity. An ILP problem belongs to the NP-hard complexity class [44]. Usually, the computational burden is characterized by the number of unknowns and constraints. The ILP (4) (for the case of a general specification on trajectory and final state) has a number of $(k \times (|P| + |T|) + 2 \times |\Pi| + 1)$ integer unknowns ($\boldsymbol{m}_i$, $\boldsymbol{\sigma}_i$, $\boldsymbol{x}$, b) and a total number of $(k \times (3 \times |P| + |T|) + 4 \times |\Pi| + |P| + 1)$ constraints. The number of constraints and unknowns of ILPs (3) and (4) does not depend on the team size N. Some data for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2017.2760249, IEEE Transactions on Automatic Control

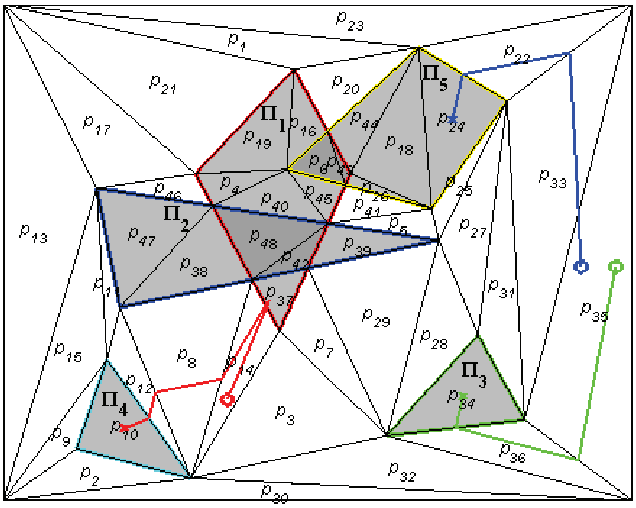

Fig. 1. Environment with five regions of interest, labeled with elements of set $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_5\}$, and three robots initially deployed in positions marked by the red, blue and green circles. Triangular partition of the environment has 48 cells. Solution (optimal with respect to the overall number of transitions) comprises a total number of 10 movements between cells. Each robot follows its trajectory and stops in the point marked with "x", and thus the team fulfills mission φ.

computational complexity is mentioned in the examples from Sec. VI.

VI. Simulation examples

This section illustrates the usage of our method for planning a team of mobile robots. The described approach was implemented in Matlab, as an addition to the Robot Motion Toolbox RMTool [45]. Our implementation includes the external ILP and LPP solvers from [43]. For exemplification purposes, we simply consider unitary weights $\boldsymbol{w} = \mathbf{1}$ and $\lambda = \mu = 1$ in cost functions of ILPs (3) and (4). Thus, in this section we refer to the total number of firing transitions as minimized cost.

We consider the environment depicted in Fig. 1, where five polygonal regions are defined and represented with differently colored borders, for easier observing their overlapping. For simplicity of constructing the team model, we consider $N = 3$ point and fully-actuated agents, whose initial positions are marked with circles in Fig. 1. Alg. 1 from Sec. III-A yields the PN system $\mathcal{Q}$ as follows. The environment is partitioned by using a constrained triangular decomposition [45], based on polygonal regions Π. The resulting partition has 48 cells (labeled with elements of set $P = \{p_1, p_2, \ldots, p_{48}\}$) and it is shown in Fig. 1. There result 140 transitions in T, given by adjacency between cells (two triangles are adjacent if they share an entire facet). The observation map h is easily created based on the inclusion of each cell in some regions of interest, e.g., $h(p_3) = \emptyset$, $h(p_{10}) = \Pi_4$, $h(p_{48}) = \{\Pi_1, \Pi_2\}$. System $\mathcal{Q}$ has three tokens and the initial marking is given by initial team deployment: $\boldsymbol{m}_0[p_{14}] = 1$, $\boldsymbol{m}_0[p_{35}] = 2$, and $\boldsymbol{m}_0[p_i] = 0$, $\forall i \in \{1, \ldots, 48\}$, $i \neq 14, 35$.

Considering the syntax and semantics explained in Sec. III-B, the team mission is given by the specification:

$$\varphi = \neg\Pi_2 \wedge \Pi_1 \wedge \neg\pi_1 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5. \tag{5}$$

In words, the second region should be avoided, the first region should be visited along run, but no robot should finally remain inside it, and the last three regions should be occupied when the robots stop.

Formula φ is converted into a system of 6 linear inequalities with 6 binary variables. By adopting the optimal solution described in Sec. V with a maximum number of steps $k = 10$, the firing sequences translate to the following runs for the robots, that can be followed without any synchronization among agents (Sec. V-C):

$$\begin{array}{rl} \text{red robot:} & p_{14}, p_{37}, p_{14}, p_8, p_{12}, p_{10} \\ \text{blue robot:} & p_{35}, p_{36}, p_{34} \\ \text{green robot:} & p_{35}, p_{33}, p_{22}, p_{24} \end{array} \tag{6}$$

The ILP problem from Sec. V includes 1891 variables (from which 1400 are integer and 10 binary), 480 equality constraints and 554 inequality constraints. The solution was obtained in around 0.01 seconds on an i7-6700 CPU. Under the same conditions, if k were set to 20, the running time increases to 1 second.

The actual robotic trajectories are presented in Fig. 1, and they were constructed by connecting the middle points of the common edges shared by successive cells from each robot's path, this being a fast method for constructing continuous trajectories for fully-actuated robots evolving in partitioned environment with convex cells [2]. Finally, each robot converges to the centroid of the last visited cell.

As mentioned, the solution complexity is not influenced by the team size. For example, if $N = 10$ robots were considered for the above case, the solution is obtained in the same amount of time. Some robots simply do not move, and the resulted number of transitions from the PN model decreased to 7. A scenario with 10 robots, 10 regions of interest and 66 PN places was solved in 0.42 seconds, thus supporting the computational feasibility of the method. More simulation results and comparisons are given in [46].

VII. Conclusions

We presented an approach that automatically plans a team of mobile robots based on a Boolean-based task given over a set of regions in the environment. The solution relies on solving an ILP optimization problem that is formulated over a discrete event system. Based on a partition of the environment, the robotic team is abstracted to a PN with outputs, which has the advantage that the topology remains fixed and only the number of tokens varies with the team size. The Boolean formula is represented through a set of linear inequalities in some binary variables, the evaluations of these variables are linked with a finite sequence of PN markings, and the PN's fundamental equation is used for making sure that any obtained marking is reachable through a firing sequence. Thus, we obtain an ILP formulation for the proposed problem, and its solution provides a set of firing PN transitions which are algorithmically converted to individual robotic trajectories. The solution is optimal with respect to a weighting of expected traveled distances and possible congestion situations. A simpler ILP is obtained for the particular case of a Boolean requirement only on the final team state, while the complexity increases for the general case of including trajectory restrictions. Due to the considered specifications, the robots can follow their trajectories without synchronizing with other team members. We implemented our procedure as a freely-downloadable Matlab software package whose usefulness is illustrated through included simulation results.

REFERENCES

[1] C. Mahulea and M. Kloetzer, "Planning mobile robots with boolean-based specifications," in *53rd IEEE Conf. on Decision and Control*, Los Angeles, USA, December 2014, pp. 5137–5142.

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

[3] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[4] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 61–71, 2007.

[5] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[6] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[7] F. Imeson and S. L. Smith, "A Language For Robot Path Planning in Discrete Environments: The TSP with Boolean Satisfiability Constraints," in *IEEE Conf. on Robotics and Automation*, May 2014, pp. 5772 – 5777.

[8] F. Imeson and S.-L. Smith, "Multi-robot task planning and sequencing using the SAT-TSP language," in *IEEE Conf. on Robotics and Automation*, May 2015, pp. 5397 – 5402.

[9] M. Kloetzer and C. Mahulea, "A Petri net based approach for multi-robot path planning," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 24, no. 4, pp. 417–445, 2014.

[10] ——, "LTL-based Planning in Environments with Probabilistic Observations," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 4, pp. 1407 – 1420, 2015.

[11] H. Costelha and P. Lima, "Robot task plan representation by Petri nets: modelling, identification, analysis and execution," *Journal of Autonomous Robots*, pp. 1–24, 2012.

[12] T. Wongpiromsarn, U. Topcu, and R.-M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[13] M. Guo, J. Tumova, and D. Dimarogonas, "Cooperative decentralized multi-agent control under local ltl tasks and connectivity constraints," in *53rd IEEE Conf. on Decision and Control*, 2014, pp. 75–80.

[14] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[15] S. Tanaka and M. Araki, "An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times," *Computers & Operations Research*, vol. 40, no. 1, pp. 344 – 352, 2013.

[16] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, "Gossip algorithms for heterogeneous multi-vehicle routing problems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, no. 1, pp. 156–174, 2013.

[17] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed task assignment based on gossip with guaranteed performance on heterogeneous networks," in *IFAC Conf. on Analysis and Design of Hybrid Systems*, Atlanta, USA, October 2015.

[18] T. Cao and A. Sanderson, "Task decomposition and analysis of robotic assembly task plans using petri nets," *IEEE Trans. on Industrial Electronics*, vol. 41, no. 6, pp. 620–630, 1994.

[19] F.-Y. Wang, "A Petri-net coordination model for an intelligent mobile robot," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 21, no. 4, pp. 777–789, 1991.

[20] C. Rust and M. Gruenewald, "Petri net based design of a multi-robot scenario - a case study," in *IEEE Conf. on Systems, Man and Cybernetics*, 2004.

[21] N. Wu and M. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Trans. Systems, Man and Cybernetics*, vol. 35, no. 6, pp. 1193–1201, 2005.

[22] B. Lacerda and P.-U. Lima, "LTL-based decentralized supervisory control of multi-robot tasks modelled as Petri nets," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 3081–3086.

[23] G. Yasuda, *Distributed Coordination of Multiple Robot Systems Based on Hierarchical Petri Net Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 602–613.

[24] S. Reveliotis and E. Roszkowska, "Conflict Resolution in Free-Ranging Multivehicle Systems: A Resource Allocation Paradigm," *IEEE Trans. on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[25] J. Goryca and R. Hill, "Formal synthesis of supervisory control software for multiple robot systems," in *American Control Conference*, June 2013, pp. 125–131.

[26] E. Roszkowska and S. Reveliotis, "A distributed protocol for motion coordination in free-range vehicular systems," *Automatica*, vol. 49, pp. 1639–1653, 2013.

[27] E. Dallal, A. Colombo, D. Del-Vecchio, and S. Lafortune, "Supervisory control for collision avoidance in vehicular networks using discrete event abstractions," *Discrete Event Dynamic Systems*, pp. 1–44, 2016.

[28] J. King, R. Pretty, and R. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Trans. Systems, Man and Cybernetics*, vol. 33, no. 5, pp. 615–619, 2003.

[29] M. Kloetzer, C. Mahulea, and J.-M. Colom, "Petri net approach for deadlock prevention in robot planning," in *IEEE Conf. on Emerging Technologies Factory Automation*, Cagliari, Italy, 2013.

[30] T. Nishi and R. Maeno, "Petri net decomposition approach to optimization of route planning problems for agv systems," *IEEE Trans. on Automation Science and Engineering*, vol. 7, no. 3, pp. 523–537, 2010.

[31] K. Klai, S. Haddad, and J.-M. Ilie, "Modular Verification of Petri Nets Properties: A Structure-Based Approach," in *Formal Techniques for Networked and Distributed Systems - FORTE 2005*, ser. Lecture Notes in Computer Science, F. Wang, Ed. Springer Berlin Heidelberg, 2005, vol. 3731, pp. 189–203.

[32] J. Esparza and C. Schrter, "Net reductions for ltl model-checking," in *Correct Hardware Design and Verification Methods*, ser. Lecture Notes in Computer Science, T. Margaria and T. Melham, Eds. Springer Berlin Heidelberg, 2001, vol. 2144, pp. 310–324.

[33] M. Rahim, M. Boukala-Ioualalen, and A. Hammad, "Petri Nets Based Approach for Modular Verification of SysML Requirements on Activity Diagrams," in *PNSE'14: Int. Workshop on Petri Nets and Software Engineering*, 2014, a satellite event of Petri Nets 2014.

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

[35] M. D. Berg, O. Cheong, and M. van Kreveld, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, 2008.

[36] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Automatic Control*, vol. 51, pp. 938–948, 2006.

[37] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Trans. on Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.

[38] F. Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd ed. Dover Publications, 2012.

[39] M. N. Velev, "Efficient Translation of Boolean Formulas to CNF in Formal Verification of Microprocessors," in *Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '04. Piscataway, NJ, USA: IEEE Press, 2004, pp. 310–315.

[40] K. Seow, C. Ma, and M. Yokoo, "Multiagent planning as control synthesis," in *Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, 2004, pp. 972–979.

[41] M. T. Pham and K. T. Seow, "Discrete-event coordination design for distributed agents," *IEEE Trans. on Automation Science and Engineering*, vol. 9, no. 1, pp. 70–82, 2012.

[42] J. Chinneck, *Practical Optimization: A Gentle Introduction*. http://www.sce.carleton.ca/faculty/chinneck/po.html, 2004.

[43] IBM. (2016) IBM ILOG CPLEX Optimization Studio. Software. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

[44] M. Earl and R. D'Andrea, "Iterative MILP methods for vehicle-control problems," *IEEE Trans. Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[45] R. Gonzalez, C. Mahulea, and M. Kloetzer, "A matlab-based interactive simulator for mobile robotics," in *IEEE Int. Conf. on Automation Science and Engineering*, Gothenburg, Sweden, 2015.

[46] L. Parrilla, C. Mahulea, and M. Kloetzer, "Rmtool: recent enhancements," in *IFAC World Congress*, Toulouse, France, July 2017.