On Multi-Robot Path Planning based on Petri Net Models and LTL Specifications

Sofia Hustiu, Cristian Mahulea, Marius Kloetzer, and Jean-Jacques Lesage

***Abstract*—This paper proposes a method exploiting the advantages of Petri net (PN) and Büchi automata models, by joining them in a newly defined *Composed Petri net* representation. Based on the latter model, collision-free trajectories are computed for a team of robots. The path planning algorithm is divided into two steps: computing a solution in a reduced PN model, and projecting it to the PN assigned to the environment. The results, given by a set of Mixed Integer Linear Programming (MILP) problems, yield lower computational complexity when compared with previous approaches.**

***Index Terms*—High-level specifications, formal methods, Petri nets, autonomous robots**

I. Introduction

The significance of developing path planning methods for a team of mobile robots (also known as *agents*) has intensified in recent years. Different models are used for robots' movements in a given environment, such as transition systems (TS) [1], [2] or Petri net (PN) models [3], [4]. In multiple scenarios, robots are required to fulfill a global goal. Known formalisms expressing the mission for the team of robots are based on high-level specifications, such as Linear Temporal Logic (LTL) [5]. Motion planning should ensure the given mission by computing collision-free trajectories. Certainly, the association between the robots and the specification can be computed in various ways to return a solution, e.g., use of TS abstractions for heterogeneous robotic systems and model-checking algorithms for the LTL mission, modeled by a Büchi automaton (BA).

As far as we know, there is no method to decompose a global LTL mission, without considering certain assumptions or particular classes of the LTL formalism, a fact explored through the current work. [1] proposes to distribute the specification into individual tasks solved by only one robot, resulting in a smaller number of states compared to the centralized approach with an exponential increase of discrete state w.r.t. the number of robots. Another planning strategy relies on PN representations [6], which have the benefits of a graph-like topology for the motion of the entire team, and an invariant model w.r.t. number of robots. Work [6] aims to compute robots' trajectories in the PN model that follows a specific run (path) in the BA associated with the LTL formula. The algorithm considers a set of k runs in the form of *prefix* and *suffix* in BA using the k shortest path algorithm, where $k \in \mathbb{N}$ is a design parameter. One inconvenience that appears is the reiteration of the procedure whenever a run in BA cannot be followed due to the generation of different observations based on robotic movement in the workspace.

This work considers a parallel approach compared to the sequential one of [6]. We propose a sound algorithm that ensures the attainment of a correct solution (expressed as collision-free robot trajectories). The contributions of the current work are as follows:

- Defining the *Composed Petri net*, which joins the multi-agent system with the given global LTL mission. A reduced model (Quotient PN) is computed w.r.t. the original PN of the environment, which is further combined with a Büchi PN model associated with the LTL specification's BA. The movement sequence for the team of robots is returned by an algorithm that includes the solution of two Mixed Integer Linear Programming problems;
- Providing a sound algorithm ensuring collision-free trajectories that accomplish the LTL mission;
- Scalability of the model for the robotic system, with the total number of places of the composed PN being represented by a sum, rather than a product, e.g., as for automaton product [7].

The paper captures the challenges in the field of robot motion planning based on related work (Section II), asserts the problem statement (Section III), seizes the mathematical notation (Section IV), and defines the contributions of the new proposed framework (Sections V, VI). The effectiveness of the current work is assessed through numerical simulations and comparisons (Section VII), while the last section contains conclusions and further improvements.

II. Related work

Path planning for mobile robots started from point-to-point trajectory for a single agent [8], being extended to multiple agents (often assumed to be identical) to ensure given mission(s). An intuitive high-level language is LTL, which encodes human representation in a logical and temporal formalism, e.g., "visit region A, then region B and always avoid region C", for which a Rabin automaton [9], or a Büchi automaton [6] is associated.

This formalism is efficient in expressing both local (one agent) [10] and global (the entire team) [11] missions, being suitable to plan optimal motion trajectories [12]. Sampling-based planning methods operate effectively for a single agent,

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

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

J.J. Lesage is with Université Paris-Saclay, ENS Paris-Saclay, LURPA, 91190, Gif-sur-Yvette, France {jean-jacques.lesage@ens-paris-saclay.fr}

The work of C. Mahulea has been partially supported by the MINECO "Salvador de Madariaga" program. This work was partially supported by Grants PID2021-125514NB-I00 and TED2021-130449B-I00 funded by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR".

e.g., using the Rapidly Exploring Random Graph (RRG) algorithm applied to an incremental transition system modeling the robot [13], and for multi-agent systems, e.g., considering stochastic optimization techniques and excluding synchronizations [14].

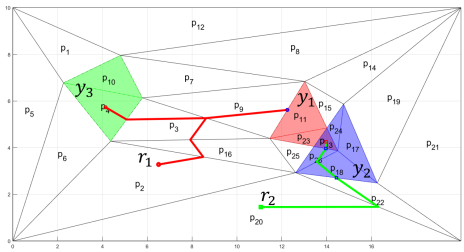
Multi-robot navigation field is directed towards concepts as (1) types of approaches: centralized, decentralized, distributed; (2) problem formulation with one global or a set of individual missions. Some LTL formulae given as global missions can be decomposed for PN-based [11] and automata-based [15] approaches, while other works focus on cooperation of agents ensuring local LTL tasks [10].

As mentioned in Section I, one mechanism to combine an LTL formula with a representation of the robotic system is based on TS associated for each agent, based on a partitioned environment [2], [3]. A different approach is illustrated in [16], where the nodes illustrate the kinematics of individual agents and the edges capture the interagent constraints. These approaches encounter the same problem of model complexity due to TS products [7]. One solution to overcome this downside is to decompose the problem into multiple finite-horizon planning issues and solve them iteratively [1]. Other works consider a fixed topology w.r.t. the number of agents, based on PN models, e.g., [17] proposed a PN path planning method considering partially unknown environments. In [18], the PN model is subject to three characteristics, representing (i) the environment, (ii) the changes in the workspace, and (iii) the task plan based on the compositions of different types of events and actions towards the goal.

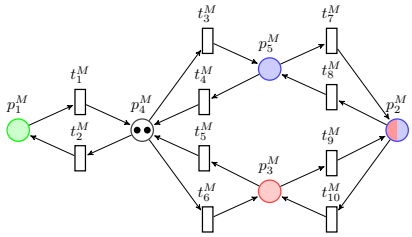
Work [19] proposes a PN-based method through a "high-level" framework enforcing parallel execution of transitions in the PN of the workspace and the BA of the LTL formula, based on the construction of a new PN supervisor model, such that a transition in the PN of the environment is fired only if a transition in the Büchi automaton is satisfied. The main idea in [6] computes an accepted run in the Büchi automaton, and then searches the trajectories in the PN to follow the imposed run of BA through generation of observations. This sequential process may produce robot movements deviating from the imposed run, the effect requiring the iteration of the entire algorithm, which is based on solving MILP problems. Compared to the latter work, the current one proposes a faster computational time solution based on a parallel approach, mentioned in the contributions from Section I.

III. Problem Statement

Let us consider a set of identical and omnidirectional point robots denoted $R = \{r_1, r_2, \ldots, r_{|R|}\}$, initially placed in the free space of a known and static 2D workspace. The environment captures several convex polygonal shapes that could overlap, be denoted as regions of interest (ROI) and labeled with elements from the set $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$. Furthermore, we assume that the environment is partitioned into a set of regions called cells, e.g., using a cell decomposition method [3], [20]. Each cell belongs entirely to one or more regions of interest, or it is included in the free space. The partitioned workspace can be further handled by a Petri net model, which is invariant w.r.t. the number of robots.



(a) The environment used in Example 3.2.



(b) Quotient Petri net of the RMPN

Fig. 1: Environment and the corresponding RMPN's quotient

The set of cells is denoted by $P = \{p_1, p_2, \ldots, p_{|P|}\}$ and the set of labels assigned to a cell is given by a function $h : P \to 2^{\mathcal{Y}}$ If the cell p_i is included in the intersection of ROIs y_j and y_k, then $h(p_i) = \{y_j, y_k\}$, while if p_l lies in the free space, then $h(p_l) = \emptyset$.

***Problem** 3.1:* Assume that the movement of a robotic team is abstracted to a Robot Motion Petri Net (RMPN) model to be defined in Def. 4.1. Given a global LTL specification over the set $\mathcal{Y}$, automatically compute the robots' trajectories to fulfill the specification.

***Example** 3.2:* Let us consider the environment in Fig. 1(a), and the following mission:

$$\varphi = \diamondsuit (y_1 \wedge y_2 \wedge y_3) \wedge \neg (y_1 \vee y_2) \mathcal{U} (y_1 \wedge y_2) \qquad (1)$$

This task means *eventually* visiting regions y_1, y_2 and y_3, while reaching y_1 and y_2 simultaneously. ■

IV. Notations and Definitions

***Definition** 4.1:* [3] A Robot Motion Petri Net system (RMPN) is a tuple $\mathcal{Q} = \langle \mathcal{N}, \boldsymbol{m}_0, \mathcal{Y}, h \rangle$, where: $\mathcal{N} = \langle P, T, \boldsymbol{Post}, \boldsymbol{Pre} \rangle$ is a Petri net, $\boldsymbol{m}_0$ 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; $h : P \to 2^{\mathcal{Y}}$ 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 ROI(s) $h(p_i)$ is (are) visited. The tuple $\mathcal{N}$ is composed from a set of places (one for each cell) P; a set of transitions T modeling the movement of the robots; $\boldsymbol{Post}, \boldsymbol{Pre} \in \{0,1\}^{|P| \times |T|}$ is the post-incidence, respectively pre-incidence matrix, defining the arcs from transitions to places and vice-versa, e.g. $\boldsymbol{Pre}[p,t] = 1$ if $p \in P$ is connected with place $t \in T$, otherwise $\boldsymbol{Pre}[p,t] = 0$.

***Example** 4.2:* Let us consider the environment in Fig. 1(a), with three ROIs ($\mathcal{Y} = \{y_1, y_2, y_3\}$) and two robots, initially located in p_2 and p_{20}. Therefore, $h(p_{11}) = h(p_{23}) = y_1$,

$h(p_{17}) = h(p_{18}) = h(p_{24}) = h(p_{26}) = y_2$, $h(p_4) = h(p_{10}) = y_3$, $h(p_{13}) = \{y_1, y_2\}$, and $h(p_i) = \emptyset$ otherwise.

The RMPN models the movement capabilities of this team of 2 robots, including a set of 26 places and 74 transitions. The set of outputs $\mathcal{Y}$ and the observation map h are given above. The initial marking $\boldsymbol{m}_0$ is a vector of dimension 26 having all elements equal to zero except $\boldsymbol{m}_0[p_2] = \boldsymbol{m}_0[p_{20}] = 1$. ■

Each token in the RMPN models the current location (cell) of a robot, thus the total number of tokens is equal to $|R|$. Notice that the structure of the model (number of places, transitions, and arcs) does not change if robots are added or removed, only the marking (state) of the RMPN.

RMPN[1] is captured by the token flow matrix $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$, the firing of a transition t_j corresponds to the movement of a robot from an input place towards an output place w.r.t. to t_j, by applying a control law that drives the robot from associated input and output cells. There exist approaches to designing such continuous laws in specific scenarios [21], [22]. We will be interested in finding sequences of transitions to be fired such that the team fulfills a given specification. If a RMPN marking $\tilde{\boldsymbol{m}}$ can be reached from $\boldsymbol{m}$ through a finite sequence of transitions σ, we denote by $\boldsymbol{\sigma} \in \mathbb{N}_{\geq 0}^{|T|}$ the firing count vector, i.e., its j^{th} element is the cumulative number of firings of t_j. The state (or fundamental) equation is satisfied:

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{C} \cdot \boldsymbol{\sigma} \tag{2}$$

A firing vector $\boldsymbol{\sigma}$ can be found, having a minimum number of transitions that drives the live[2] RMPN to the desired marking $\tilde{\boldsymbol{m}}$, i.e., by solving the optimization problem $\min \mathbf{1}^T \cdot \boldsymbol{\sigma}$. The details of transforming the firing vector into a sequence of robot movements are captured in [3].

LTL formulae and Büchi automata. The global motion task that the robot team should fulfill is specified as an LTL_{-X} formula [23] containing recursive formulae defined over a set of atomic propositions $\mathcal{Y}$, by using Boolean operators ($\neg$ - negation, $\wedge$ - conjunction, $\vee$ - disjunction, $\Rightarrow$ - implication, and $\Leftrightarrow$ - equivalence) and some temporal operators ($\mathcal{U}$ - until, $\Diamond$ - eventually, and $\Box$ - always), while excluding $\bigcirc$ - the next operator. For simplicity of notation, we write LTL instead of LTL_{-X} [23], [24], [25].

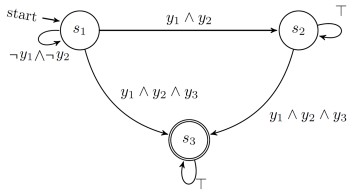
Any LTL formula can be automatically transformed into a nondeterministic Büchi automaton using the available tools [26], [27]. The property of LTL being close under stuttering [27] is exploited in Section VI (any finite repetition of the same input does not modify the truth value of the input string).

***Definition** 4.3:* The Büchi automaton (BA) corresponding to an LTL formula on the set $\mathcal{Y}$ has structure $B = (S, S_0, \Sigma_B, \rightarrow_B, F)$, where: S is a finite set of states; $S_0 \subseteq S$ is the set of initial states (assumed singleton); Σ_B is the finite set of inputs; $\rightarrow_B \subseteq S \times \Sigma_B \times S$ is the transition relation and $F \subseteq S$ is the set of final states. ■

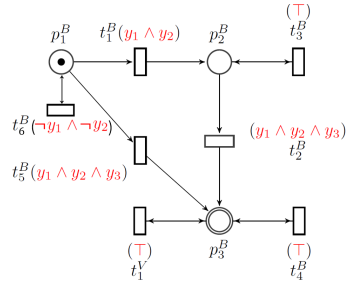
We denote by $\pi(s_i, s_j)$ the set of all inputs that enable a transition from s_i to s_j, expressed as a Boolean formula over the set $\mathcal{Y}$, reduced to a Disjunctive Normal Form (DNF), where a single conjunctive element is denoted by α_k. Alternatively, the input can be expressed as a combination of active observations from the set $2^{\mathcal{Y}}$, where $\emptyset$ represents the free space.

[1]By definition, RMPN systems considered in this paper belongs to the class of state-machine, i.e., each transition has one input and one output place.

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



(a) Büchi automaton for the LTL formula in Ex. 3.2.



(b) Büchi Petri net corresponding to the Büchi automaton

Fig. 2: Example of Büchi automaton and Büchi Petri net

An *infinite accepted run* in B drives the automaton towards a final state from an initial state through (i) *prefix* and (ii) *suffix* (path towards the same final state reached by the prefix). The run can be written as *prefix, suffix, suffix* ... [28].

***Example** 4.4:* Let us recall Example 3.2. The BA corresponding to this LTL task is given in Fig. 2(a), where symbol $\top$ (True) means any observation from $2^{\mathcal{Y}}$. On the other hand, Fig. 2(b) illustrates the Büchi Petri net model, having the Boolean formula of transitions represented with red color. This model will be described in Section V. An accepted run that satisfies the formula could be s_1, s_3, s_3, ... with the prefix s_1 and the suffix s_3. Note that the unique final state s_3 is visited infinitely often. However, as may be observed in the environment of Fig. 1(a), this run cannot be generated by the two robots, because the two robots should evolve only through cells belonging to free space until one enters p_{13}, while the other simultaneously enters p_4 or p_{10}. Clearly, entering p_{13} directly is not possible, since it would require first activating y_1 or y_2 and would violate φ (1).

A possible run generated by the robots is: s_1, s_2, s_3, s_3, ... with prefix s_1, s_2 and suffix s_3. For this, robots should first enter y_1 and y_2 synchronously (generating $\pi(s_1, s_2) = y_1 \wedge y_2$), then one robot should go to p_{13} (the intersection of y_1 and y_2). Finally, a robot should enter y_3 while crossing the free space to enable the transition to s_3 in BA. The solution is not unique since the self-loop transition in s_2 can be taken with any possible input. ■

Table I provides a summarized description of the following notations throughout the paper, using $< \cdot >$ as a placeholder for components of various representations.

V. Solution Step 1: Composed Petri Net Model

Fig. 3 provides a general overview of the entire proposed solution of the Problem 3.1. The first phase is responsible

Notation	Description
$<.^M>$	Notation for Quotient RMPN (Sub-step 1.1)
$<.^B>$	Notation for Büchi RMPN (Sub-step 1.2)
$<.^C>$	Notation for *Composed Petri net* (Sub-step 1.3)

TABLE I: Notations for various PNs to be used

for computing the newly defined *Composed Petri net* model based on a reduced representation of the environment (sub-step 1.1) and the Büchi automaton associated with the LTL specification (sub-step 1.2). The second phase focuses on an iterative algorithm that requires two actions, expressed by sub-steps 2.1 and 2.2.

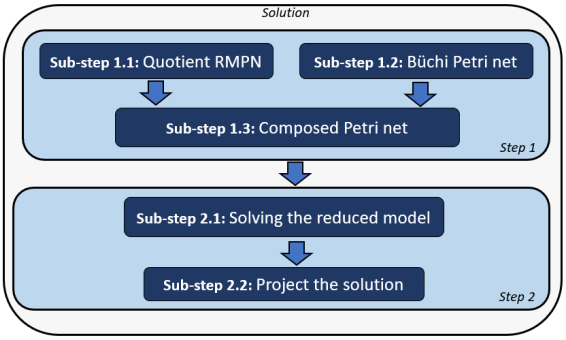


Fig. 3: Diagram for the global algorithm

Sub-step 1.1. Quotient of the RMPN in Def. 4.1. Given an RMPN system $\mathcal{Q}$ as in Def. 4.1, the idea of obtaining the quotient $\mathcal{Q}^M$ is to iteratively combine any p_i and p_j places of P that satisfy $p_j \in (p_i^\bullet)^\bullet$ and $h(p_i) = h(p_j)$. This reduction technique is synthesized in [29], and the reduced PN model $\mathcal{Q}^M$ has the property that its output is changed when a transition is fired. Thus, after firing one transition in $\mathcal{Q}^M$, one transition in Büchi should also be fired. A transition of $\mathcal{Q}^M$ corresponds to a set of trajectories in $\mathcal{Q}$.

***Example** 5.1:* Consider the RMPN of the environment as in Fig. 1(a). By aggregating the states with the same observation, the quotient PN $\mathcal{Q}^M$ in Fig. 1(b) is obtained, also being an RMPN model according to Def. 4.1. The correspondence between the reduced and the original RMPN models of the environment is captured by a projection matrix $\boldsymbol{Pr}$ with size 5×26, having all elements equal to zero except adjacent places sharing the same observation, e.g., $\boldsymbol{Pr}[p_1^M, p_4] = \boldsymbol{Pr}[p_1^M, p_{10}] = 1$.

Sub-step 1.2. Büchi Petri net. Starting from the Büchi automaton $B = \langle S, S_0, \Sigma_B, \rightarrow_B, F\rangle$ as in Def. 4.3, Alg. 1 obtains the corresponding Büchi Petri net system $\mathcal{Q}^{\mathcal{B}}$. For each state $s_i \in S$, a new place p_i^B is added to the PN (line 1). The first loop (lines 3 - 7) is executed for each transition from s_i to s_j in the Büchi automaton. The second loop (lines 4 - 7) is executed for each conjunctive element α_k from $\pi(s_i, s_j)$ and adds a new transition t_{τ_k} to the Büchi PN from p_i^B to p_j^B. Note that all transitions corresponding to the conjunctive elements in $\pi(s_i, s_j)$ have the same input, respectively, output place of the Büchi PN. In line 8, the marking vector is initialized, with p_0^B associated with $s \in S_0$. Observe that one token in Büchi

Algorithm 1: Büchi Petri net

Input: $B = \langle S, S_0, \Sigma_B, \rightarrow_B, F\rangle$
Output: $\mathcal{Q}^B = \langle\langle P^B, T^B \cup T^V, [\boldsymbol{Pre}^B\ \boldsymbol{Pre}^V],$
$[\boldsymbol{Post}^B\ \boldsymbol{Post}^V]\rangle, \boldsymbol{m}_0^B, \mathcal{Y}, h\rangle$

1 Let $P^B = \{p_1^B, p_2^B, \ldots, p_{|S|}^B\}$ be the set of $|S|$ places;
2 Let $T^B = T^V = \emptyset, \boldsymbol{Pre}^B = \boldsymbol{Post}^B = \emptyset, \boldsymbol{Pre}^V = \boldsymbol{Post}^V = \emptyset$;
3 **forall** $(s_i, \tau, s_j) \in \rightarrow_B$ **do**
4 **forall** *conjuctive element* α_k *of* $\pi(s_i, s_j)$ **do**
5 $T^B = T^B \cup t_{\tau_k}$;
6 Add a new column to $\boldsymbol{Pre}^B$ and to $\boldsymbol{Post}^B$ corresponding to t_{τ_k};
7 $\boldsymbol{Pre}^B[p_i^B, t_{\tau_k}] = \boldsymbol{Post}^B[p_j^B, t_{\tau_k}] = 1$;
8 Let $\boldsymbol{m}_0^B = \boldsymbol{0}^{|S|\times 1}$;
9 $\boldsymbol{m}_0^B[p_0^B] = 1$;
10 **forall** $s_f \in F$ **do**
11 $T^V = T^V \cup t_{s_f}$;
12 Add a new column to $\boldsymbol{Pre}^V$ and to $\boldsymbol{Post}^V$ corresponding to t_{s_f};
13 $\boldsymbol{Pre}^V[p_f^B, t_{s_f}] = \boldsymbol{Post}^V[p_f^B, t_{s_f}] = 1$;

PN is sufficient to mark the current state of the automaton B, since $|S_0| = 1$. Alg. 1 returns also *virtual transitions* T^V, by adding on $t_i^V \in T^V$ to each final state Büchi PN, expressed in $\boldsymbol{Pre}^V$ and $\boldsymbol{Post}^V$. These transitions will have zero firing cost when the solution of MILP (3) is computed, to maintain the Büchi PN in the final state when possible.

***Example** 5.2:* Consider the Büchi automaton in Fig. 2(a). Applying Alg. 1, the Büchi PN in Fig. 2(b) is obtained. In this example, the places p_2^B and p_3^B are connected with a single transition corresponding to input $\pi(s_2, s_3) = y_1 \wedge y_2 \wedge y_3$, visualized in red. Fig. 2(b) illustrates the addition of the virtual transition t_1^V connected to the final state s_3 by a reading (bidirectional) arc. ■

Sub-step 1.3. Composition of Quotient RMPN and Büchi Petri net systems. Alg. 2 includes the full strategy to return the *Composed PN* system $\mathcal{Q}^C$, by the composition of the robotic team model $\mathcal{Q}^M$ with the one of the specification $\mathcal{Q}^B$. For this, a number of $2 \times |\mathcal{Y}|$ places are needed, from which half models the active observations, given by the set P^O, while the other $|\mathcal{Y}|$ places model inactive observations $P^{\neg O}$. Initially, places p_i^O have zero tokens, while places $p_i^{\neg O}$ have the number of tokens equal with $|R|$, denoting no active observation in the initial marking line (3). The sum of the tokens in $p_i^{\neg O}$ and p_i^O is always equal to $|R|$. The lines 1 - 2 define the sets of places, respectively, transitions for the *Composed PN* $\mathcal{Q}^C$.

Matrices $\boldsymbol{Pre}^C$ and $\boldsymbol{Post}^C$ are initialized in lines 4 - 5. Lines 6 - 10 are executing for each observation y_i. Input arcs are added to place p_i^O for each $p_k \in P^M$, $y_i \in h(p_k)$. Additionally, arcs from p_i^O to all output transitions of p_k are added. In this way, when a robot enters a region p_k with output y_i, one token is added to p_i^O. If $\boldsymbol{m}[p_i^O] > 0$ then observation y_i is active. The place $p_i^{\neg O}$ is the complementary place of p_i^O, therefore is connected with the same transitions as p_i^O but with arcs oriented in the other sense. If $\boldsymbol{m}[p_i^{\neg O}] = |R|$ then

observation y_i is not active. Finally, loop in lines 11 to 17 connects the places p_i^O, respectively $p_i^{\neg O}$ with the transitions t_τ^B by a reading arc, according to the assigned Boolean formula (conjunction), considering weights 1, respectively $|R|$.

Algorithm 2: Composed Petri net system

Input: $P^O, P^{\neg O}, \mathcal{Q}^M = \langle\langle P^M, T^M, \boldsymbol{Pre}^M, \boldsymbol{Post}^M\rangle, \boldsymbol{m}_0^M, \mathcal{Y}, h\rangle$, $\mathcal{Q}^B = \langle\langle P^B, T^B \cup T^V, [\boldsymbol{Pre}^B\ \boldsymbol{Pre}^V], [\boldsymbol{Post}^B\ \boldsymbol{Post}^V]\rangle, \boldsymbol{m}_0^B, \mathcal{Y}, h\rangle$,

Output: $\mathcal{Q}^C = \langle\langle P^C, T^C, \boldsymbol{Pre}^C, \boldsymbol{Post}^C\rangle, \boldsymbol{m}_0^C, \mathcal{Y}, h\rangle$

1 Let $P^C = P^M \cup P^B \cup P^O \cup P^{\neg O}$;

2 Let $T^C = T^M \cup T^B \cup T^V$;

3 $\boldsymbol{m}_0^C = [\boldsymbol{m}_0^M,\ \boldsymbol{m}_0^B, \boldsymbol{m}_0^O]$;

4 Let $\boldsymbol{Pre}^C =$

$$\begin{bmatrix} \boldsymbol{Pre}^M & \mathbf{0}^{|P^M|\times|T^B|} & \mathbf{0}^{|P^M|\times|T^V|} \\ \mathbf{0}^{|P^B|\times|T^M|} & \boldsymbol{Pre}^B & \boldsymbol{Pre}^V \\ \mathbf{0}^{|P^O|\times|T^M|} & \mathbf{0}^{|P^O|\times|T^B|} & \mathbf{0}^{|P^O|\times|T^V|} \\ \mathbf{0}^{|P^{\neg O}|\times|T^M|} & \mathbf{0}^{|P^{\neg O}|\times|T^B|} & \mathbf{0}^{|P^B|\times|T^V|} \end{bmatrix};$$

5 Let $\boldsymbol{Post}^C =$

$$\begin{bmatrix} \boldsymbol{Post}^M & \mathbf{0}^{|P^M|\times|T^B|} & \mathbf{0}^{|P^M|\times|T^V|} \\ \mathbf{0}^{|P^B|\times|T^M|} & \boldsymbol{Post}^B & \boldsymbol{Post}^V \\ \mathbf{0}^{|P^O|\times|T^M|} & \mathbf{0}^{|P^O|\times|T^B|} & \mathbf{0}^{|P^O|\times|T^V|} \\ \mathbf{0}^{|P^{\neg O}|\times|T^M|} & \mathbf{0}^{|P^{\neg O}|\times|T^B|} & \mathbf{0}^{|P^O|\times|T^V|} \end{bmatrix};$$

6 **forall** $y_i \in \mathcal{Y}$ **do**

7 Let $P' = \{p \in P^M | y_i \in h(p)\}$;

8 **forall** $p_k \in P'$ **do**

9 $\boldsymbol{Post}^C[p_i^O, {}^\bullet p_k] = \boldsymbol{Pre}^C[p_i^O, p_k{}^\bullet] = 1$;

10 $\boldsymbol{Pre}^C[p_i^{\neg O}, {}^\bullet p_k] = \boldsymbol{Post}^C[p_i^{\neg O}, p_k{}^\bullet] = 1$;

11 **forall** $t_\tau^B \in T^B$ **do**

12 Let π_i be the DNF formula assigned to t_τ^B;

13 **if** $\pi_i \neq \top$ **then**

14 **forall** *atomic propositions* y_i *appearing* ***not negated*** *in* π_i **do**

15 $\boldsymbol{Pre}^C[p_i^O, t_\tau^B] = \boldsymbol{Post}^C[p_i^O, t_\tau^B] = 1$;

16 **forall** *atomic propositions* y_i *appearing* ***negated*** *in* π_i **do**

17 $\boldsymbol{Pre}^C[p_i^{\neg O}, t_\tau^B] = \boldsymbol{Post}^C[p_i^{\neg O}, t_\tau^B] = |R|$;

Fig. 4 depicts a part of *Composed Petri net* with its initial marking, returned by Alg. 2. For the sake of clarity, Fig. 4 considers only the arcs for one region on interest (y_3). When a transition in $\mathcal{Q}^M$ fires and the observation changes, the $\mathcal{Q}^M$ deposits one token to the respective active observation, e.g., t_2^M is enabled when y_3 is not active (the robots being in the free space initially), and if it fires, a token is produced to both p_1^M and p_3^O. In $\mathcal{Q}^B$, the transitions are fired based on the assigned Boolean formula, being triggered by the reading arcs of $P^O, P^{\neg O}$, e.g., t_5^B and t_2^B depend on active observation y_3. The transitions in Büchi PN from Fig. 4 are colored according to the required active observations, i.e., red - y_1, blue - y_2, green - y_3. This rationale is maintained towards the rest of active and inactive intersection $y_1 \wedge y_2$.

Remark. Quotient PN is polynomial in ($\mathcal{O}(|P|^2)$), subject

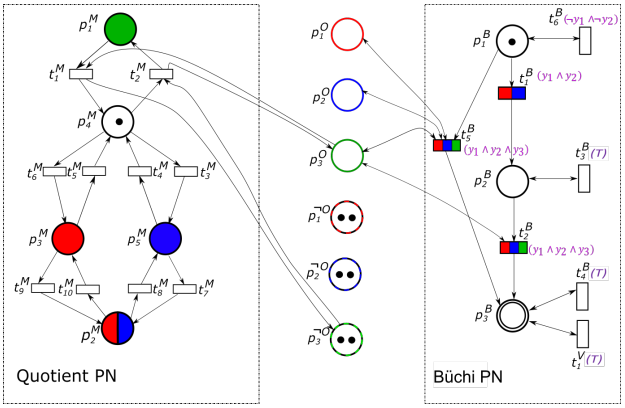


Fig. 4: Part of the Composed Petri net, based on active and inactive observations of y_3

to the partitioned environment. Alg. 1 is polynomial w.r.t. the inputs $\rightarrow_B$ of the Büchi automaton $\mathcal{B}$ and the number of atomic propositions over set $\mathcal{Y}$. In general, the LTL formulae have a reduced number of atomic propositions, the exponential upper-bound of $2^{|\mathcal{Y}|}$ being seldom achieved. Lastly, Alg. 2 is polynomial over cardinality of sets $\mathcal{Y}$ and T^B.

VI. Solution Step 2: Algorithm

The robot navigation algorithm requires two actions to be achieved: (i) compute a solution on the reduced model $\mathcal{Q}^C$, (ii) project the searched solution in the original PN of the environment, outputting the multi-robot system trajectories. The proposed algorithm rules out solutions that cannot be projected into the PN of the environment, by imposing in the first step to obtain a different solution in the reduced model. This idea is similar to the one in [30], being here applied to a PN-based approach, rather than a graph-based approach.

Global algorithm. Alg. 3 computes robots' movements while ensuring the mission φ. The algorithm stops when a set of trajectories is returned for a place $p_{f_i}^B \in Set_f$, with Set_f modeling the final states in B (line 15). The main idea is to first search for a feasible run in $\mathcal{Q}^C$ (lines 3 - 12) based on MILP (3), which is individually called for *prefix* and *suffix*, with different initial marking. The *suffix* is computed through MILP (3) only when the last active observation of a final place $p_{f_i}^B$ is included in its self-loop (p_j^O for observation y_j has at least one token). A feasible Run is obtained when both *prefix* and *suffix* are not empty (lines 11 - 15), for which a projection of this solution is explored based on MILP (4).

If Run cannot be projected into the original RMPN, then we memorize the previous solutions returned by MILP (3) for both *prefix* and *suffix* considering only the transitions in $\mathcal{Q}^M$. Line 17 appends these solutions $\sum_{i=1}^{k} \sigma_i^M$ of prefix (subscript $_p$), respectively for suffix (subscript $_s$) to sets of bad solutions $CE_{p|s}$. These sets will be considered counterexamples in MILP (3), imposing to compute a different Run. When no overall solution is outputted for any p_{f_i}, the entire process is iterated by increasing the number of steps k.

Sub-step 2.1. Solution on the reduced model. The main idea of MILP (3) is to drive the PN to a state corresponding to a final state in Büchi (marking $\boldsymbol{m}_{2k}^C$). The MILP is solved

Algorithm 3: Global solution for robot's trajectories

Input: RMPN $\mathcal{Q}$, *Composed Petri net* $\mathcal{Q}^C$, set $\mathcal{Y}$, $|R|$, set of active observations P^O, set of final places Set_f , finite horizon k

Output: $Traj$ = Sequence of firing transitions

Let $CE_{p|s} = \emptyset$ and $flag = False$;
while *($k \leq U$) OR ($k > U$ AND $flag = True$)* **do**
  forall $p^B_{f_i} \in Set_f$ **do**
    Compute *prefix* for $p^B_{f_i}$ with MILP (3);
    if prefix $\neq \emptyset$ **then**
      Let P^O_f be last active observation for $p^B_{f_i}$;
      if $P^O_f \not\models \pi(s_f, s_f)$ **then**
        Compute *suffix* with MILP (3), where $\boldsymbol{m}^C_0 = \boldsymbol{m}^C_k$;
      else
        suffix = s_f;
    if *(*prefix $\neq \emptyset$ *AND* suffix $\neq \emptyset$*)* **then**
      Run = *prefix suffix suffix* ...;
      Project $Traj$ with MILP (4);
      if $Traj \neq \emptyset$ **then**
        Return $Traj$;
    $flag = True$;
    $CE_{p|s} = CE_{p|s} \cup \sum_{i=1}^{k} \sigma^M_i$
  Increase k;

individually for *prefix* and *suffix*, each for k steps, with $k \geq 1$ being a design parameter. For each odd, respectively even step, a transition in Quotient PN, respectively in Büchi PN is fired. The upper-bound of k is $U = \big(|P^M| - 1\big) \times \big(|P^B| - 1\big)$, as it may be necessary to move a token through all places P^M to produce a token in the next place Büchi PN.

Parameters: $|R|$ - number of robots; p^B_f - place modeling a final state in Büchi; $\boldsymbol{C}^C$ - token flow matrix of $\mathcal{Q}^C$; $\boldsymbol{Pre}^C$ - pre-incidence matrix of $\mathcal{Q}^C$; $CE_{p|s}$ - set of bad solutions for both prefix and suffix.

Variables:

- $\boldsymbol{m}^C_i = \big[\boldsymbol{m}^M_i \ \boldsymbol{m}^B_i \ \boldsymbol{m}^O_i \ \boldsymbol{m}^{\neg O}_i\big]^T$ - marking column vector at step i of $\mathcal{Q}^C$ composed by the marking of Quotient PN ($\boldsymbol{m}^M_i$), Büchi PN ($\boldsymbol{m}^B_i$), active observation places ($\boldsymbol{m}^O_i$) and inactive observation places ($\boldsymbol{m}^{\neg O}_i$);
- $\boldsymbol{\sigma}^C_i = \begin{bmatrix} \boldsymbol{\sigma}^M_i \\ \boldsymbol{\sigma}^B_i \\ \boldsymbol{\sigma}^V_i \end{bmatrix}$ - firing vector at step i of the *Composed PN*, composed by the firing vector of Quotient PN ($\boldsymbol{\sigma}^M_i$), Büchi PN ($\boldsymbol{\sigma}^B_i$) and of virtual transitions ($\boldsymbol{\sigma}^V_i$);
- $\boldsymbol{z}^1, \boldsymbol{z}^2 \in \{0,1\}^{|T^M|}$ - binary vectors with $\boldsymbol{z}^1[j] = 1$ if $\boldsymbol{\zeta} - \sum_{i=1}^{k} \sigma^M_i \geq 1$, and $\boldsymbol{z}^1[j] = 0$ otherwise, respectively $\boldsymbol{z}^2[j] = 1$ if $\boldsymbol{\zeta} - \sum_{i=1}^{k} \sigma^M_i \leq 1$, otherwise $\boldsymbol{z}^2[j] = 0$, with $\boldsymbol{\zeta} \in CE_{p|s}$.

Objective:

$$\min \sum_{i=1}^{2\cdot k} i \cdot \big(\mathbf{1}^T \cdot \boldsymbol{\sigma}^M_i + \mathbf{1}^T \cdot \boldsymbol{\sigma}^B_i\big) \tag{3a}$$

Constraints:

$$\boldsymbol{m}^C_i - \boldsymbol{m}^C_{i-1} - \boldsymbol{C}^C \cdot \boldsymbol{\sigma}^C_i = 0, \qquad i=\overline{1,2\cdot k} \tag{3b}$$

$$\left.\begin{array}{rl} \boldsymbol{m}^C_i - \boldsymbol{Pre}^C \cdot \boldsymbol{\sigma}^C_i & \geq 0, \\ \mathbf{1}^T \cdot \boldsymbol{\sigma}^M_i & \leq |R|, \\ \mathbf{1}^T \cdot \boldsymbol{\sigma}^B_i + \mathbf{1}^T \cdot \boldsymbol{\sigma}^V_i & = 0, \end{array}\right\} \quad \begin{array}{l} i=2\cdot j+1 \\ j=\overline{0,k-1} \end{array} \tag{3c}$$

$$\left.\begin{array}{rl} \mathbf{1}^T \cdot \boldsymbol{\sigma}^M_i & = 0, \\ \mathbf{1}^T \cdot \boldsymbol{\sigma}^B_i & = 1, \\ \mathbf{1}^T \cdot \boldsymbol{\sigma}^V_i & = 0, \end{array}\right\} \quad i=2 \tag{3d}$$

$$\left.\begin{array}{rl} \mathbf{1}^T \cdot \boldsymbol{\sigma}^M_i & = 0, \\ \mathbf{1}^T \cdot \boldsymbol{\sigma}^B_i + \mathbf{1}^T \cdot \boldsymbol{\sigma}^V_i & = 1, \end{array}\right\} \quad \begin{array}{l} i=2\cdot j \\ j=\overline{2,k} \end{array} \tag{3e}$$

$$\boldsymbol{m}^C_i[p^B_f] = 1, \qquad i=2\cdot k \tag{3f}$$

$$\left.\begin{array}{rl} \boldsymbol{\zeta} - \sum_{i=1}^{k} \boldsymbol{\sigma}^M_i & \leq N \cdot (\mathbf{1} - \boldsymbol{z}^j), \\ -\boldsymbol{\zeta} + \sum_{i=1}^{k} \boldsymbol{\sigma}^M_i & \leq \mathbf{1} + N \cdot \boldsymbol{z}^j, \end{array}\right\} \quad \begin{array}{l} \forall \boldsymbol{\zeta} \in CE_{p|s} \\ i=\overline{1,2\cdot k}, j=\overline{1,2} \end{array} \tag{3g}$$

$$\mathbf{1}^T \cdot (\boldsymbol{z}^1 + \boldsymbol{z}^2) \geq 1 \tag{3h}$$

Explanations on MILP (3) are as follows: (3a) the cost function minimizes the number of fired transitions of sets T^M, T^B, enforcing to reach a solution during the first steps of k steps, when it is possible; (3b) correspond to the state equation (2); (3c) allow robots to advance at most one place in Quotient PN, which lead to a change in observation ($h(p^M_i) \neq \emptyset$) such that, in the following step (when i is even) one transition in Büchi should be fired; (3d) imposes firing a transition of set T^B, forcing the leave from the initial state; (3e) allows firing of one transition in Büchi PN; (3f) ensures the marking after k steps is the final state in Büchi, denoted as p^B_f; (3g) and (3h) guarantee that the current solution is different than any other previous bad solution $\boldsymbol{\zeta}$ from set $CE_{p|s}$, using a big number (N) method [31].

Sub-step 2.2. Projecting the solution. Let $M = \langle \boldsymbol{m}^M_1, \boldsymbol{m}^M_2, \ldots, \boldsymbol{m}^M_{2k} \rangle$ be the sequence of markings returned by MILP (3). Successive identical markings are removed. Let us add $\boldsymbol{m}^M_0$ as the first element in M. Furthermore, let $G = \langle \boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_{2k} \rangle$ be a sequence of $2k$ vectors s.t. $\boldsymbol{g}_i \in \{0,1\}^{|P^M|}$ with $\boldsymbol{g}_i[j] = 1$ if $\boldsymbol{m}^M_i[j] = 0$, and $\boldsymbol{g}_i[j] = 0$ otherwise. This vector will be used in MILP (4) to not allow activation of other observations between two steps.

The following MILP extends every marking from the Quotient RMPN into a string of markings in the original RMPN. The string has the same active observations, to ensure the validity of the LTL formula when exist finite repetitions. To avoid collisions, $|R|$ intermediate markings are additionally introduced between two successive markings, to ensure that a maximum one robot crosses each region.

Parameters: M - sequence of markings returned by (3); $\boldsymbol{Pr}$ - projection matrix between Q^B and RMPN; $\boldsymbol{C}$ - token flow matrix of the RMPN; $\boldsymbol{Pre}, \boldsymbol{Post}$ - pre/post-incidence matrices of RMPN; $\boldsymbol{m}^M_i$ - marking at step i of $\mathcal{Q}^M$.

Variables: $\boldsymbol{m}_{i,j}$ - marking at step i of RMPN, considering the intermediate marking j; $\boldsymbol{\sigma}_{i,j}$ - firing vector at step i of RMPN, for the intermediate marking j with $i = \overline{0,|M|}, j = \overline{1,|R|+1}$.

Objective:

$$\min \mathbf{1}^T \cdot \sum_{i,j} \boldsymbol{\sigma}_{i,j} \tag{4a}$$

Constraints:

$$\boldsymbol{m}_{i,j} - \boldsymbol{m}_{i,j-1} - \boldsymbol{C} \cdot \boldsymbol{\sigma}_i = \mathbf{0}, \quad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4b}$$

$$\boldsymbol{m}_{i,0} - \boldsymbol{m}_{i-1,|R|+1} = \mathbf{0}, \quad i=\overline{1,|M|} \tag{4c}$$

$$\boldsymbol{Pr} \cdot \boldsymbol{m}_{i,j} - \boldsymbol{m}_i^M = \mathbf{0}, \quad i=\overline{0,|M|}, j=\overline{1,|R|} \tag{4d}$$

$$\boldsymbol{Post}[\boldsymbol{g}_i \cdot \boldsymbol{Pr}, \cdot] \cdot \boldsymbol{\sigma}_{i,j} = \mathbf{0}, \quad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4e}$$

$$\boldsymbol{Post} \cdot \boldsymbol{\sigma}_{i,j} + \boldsymbol{m}_{i,j-1} \leq \mathbf{1}, \quad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4f}$$

$$\boldsymbol{m}_{i,|R|+1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_{i,|R|+1} \geq \mathbf{0}, \quad i=\overline{0,|M|} \tag{4g}$$

The constraints in MILP (4) are as follows: (4b) is the state equation (2); (4c) ensures that the last intermediate marking is the same as the initial marking in the next step; (4d) keeps the same observations during the $\boldsymbol{m}_{i,1}$ to $\boldsymbol{m}_{i,|R|}$, since the $|R|$ intermediate markings are introduced to avoid collisions between $\boldsymbol{m}_i^M$ and $\boldsymbol{m}_{i+1}^M$; (4e) ensure that the corresponding firing vectors of the intermediate markings from $\boldsymbol{m}_{i,1}$ to $\boldsymbol{m}_{i,|R|}$ should not activate other observations; (4f) ensures the collision avoidance between successive markings of the original RMPN by imposing each region to be crossed by a maximum one agent between two intermediate markings; (4g) impose that the movements of robots from substep $\boldsymbol{m}_{i,|R|}$ to substep $\boldsymbol{m}_{i,|R|+1}$ is done synchronously by all robots (each robot fires only one transition) such that the generated observation of $\mathcal{Q}^M$ changes according to the transitions fired in Büchi.

Complexity. The algorithm is NP-hard, justified by the use of MILP problems. The total number of unknown variables in both MILPs is given by the number of markings and transitions in both $\mathcal{Q}^C$ and $\mathcal{Q}$, and a set of binary variables of size $2 \cdot |T^M| \cdot |CE_{p|s}|$, bounded by U. The total number of constraints depends on the design parameter k for MILP (3), while for MILP (4) it depends on the previous solution.

Remark. Alg. 3 is not complete due to the possibility of spurious transitions as a result of MILP (3) and the collision avoidance restrictions of MILP (4), but it is sound, as the returned solution satisfies the LTL formula.

VII. Simulation results and comparisons

The proposed solution is implemented and integrated into RMTool - MATLAB [32], using the CPLEX Optimizer solver for both MILPs, while herein the results were obtained on a laptop with i7 - 8^{th} gen. CPU @ 2.20GHz and 8GB RAM.

***Example** 7.1:* Let us recall the environment from Fig. 1 (a) and the LTL mission from (1). The robots need to reach all three regions of interest at the same time, ensuring y_1 and y_2 are reached simultaneously. Figure 1(a) illustrates the robot trajectories returned by Alg. 3. It can be observed that both robots move toward regions y_1 and y_2 to complete the concurrent requirement. Afterwards, r_1 advances and enters the last region of interest y_3, while the second robot enters $p_{13} = \{y_1, y_2\}$. For this result, the *Quotient PN model* with 5 places and 10 transitions was computed in 0.02 seconds, and the *Composed PN* model with 14 places and 16 transitions (out of which one is a virtual one) was computed in 0.02 seconds. MILP (3) reaches the final state in Büchi in 0.05 seconds, using 180 unknown variables (for k = 6). MILP (3) returns the *prefix* $s_1 s_2$, without the necessity of being called to compute the *suffix*, because the final state s_3 was True $\top$ for its self-loop. The run time to project the solution is 0.05 seconds for 900 unknown variables, having the cost function equal to 11 (number of cells crossed by the robots). ■

***Example** 7.2:* We now consider the LTL formula $\varphi = \square(\diamondsuit y_1 \wedge \diamondsuit y_3 \wedge \diamondsuit y_5 \wedge \diamondsuit y_6 \wedge \diamondsuit y_7 \wedge \diamondsuit y_8) \wedge \neg(y_5 \vee y_6)\,\mathcal{U}\,(y_5 \wedge y_6) \wedge \neg(y_4 \vee y_7)\,\mathcal{U}\,(y_4 \wedge y_7)$. This specification imposes the visit of several ROIs in an environment with 8 regions of interest, while the regions y_5 and y_6 are simultaneously reached, respectively y_4 and y_7. Fig. 5 exemplifies robot's trajectories for 6 robots, with black stars being represented the synchronization points of the team necessary when fulfilling φ. ■

TABLE II: Comparison between current approach *with Büchi* and *following Büchi* captured in [6] for Example 7.2

Number of robots	**Run time to return a solution [sec]**		**Cost function value**	
	following Büchi	*with Büchi MILP* (3)	*following Büchi*	*with Büchi*
4	9.56	0.75	37	39
5	2.22	0.26	23	28
6	0.77	0.11	20	21
10	1.2	0.78	13	13

Table II contains a result analysis of the current approach of Alg. 3 in contrast with our previous work [6], focused on running time and value of the cost function (number of crossed cells of all robots) computed for $k = 10$ intermediate markings. Let us recall that [6] examines a transition sequence in the PN model of the environment while following an entire run computed in Büchi automaton. One can refer to the present work having a parallel approach (denoted here as *with Büchi*), while the previous one can be considered as a sequential approach (denoted as *following Büchi*). It should be noted that the collision avoidance strategy is ensured through the current work (restrictions (4f) of MILP (4)), contrary to the previous work out of which collision-free trajectories cannot be guaranteed.

Discussion. Based on the run simulations, this work yields lower computational time and model size w.r.t. previous works. The procedure *with Büchi* returns a scalable PN model w.r.t. the number of robots, having maximum number of places $|P^M| + |P^B| + 2 \cdot |\mathcal{Y}|$. On the other hand, the model returned by the procedure *following Büchi* contains the number of places given by the RMPN $\mathcal{Q}$, while approaches based on transition systems [7] are highly dependent on the size of the team. For example, the size of models in Example 7.2 are as follows: 37 places (*with Büchi*), 62 places (*following Büchi*), $11^{|R|} \times 10$ (quotient transition systems), as 11, respectively 10, represent the number of states in the reduced transition system, respectively in BA.

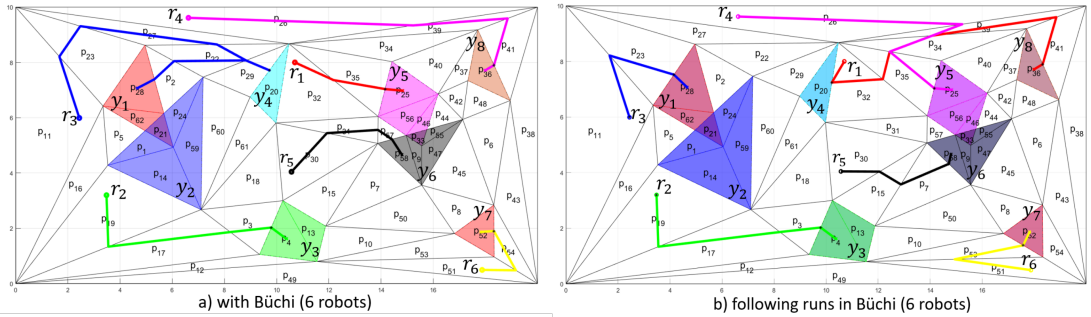


Fig. 5: Returned trajectories for Example 7.2 (red - r_1; green - r_2, blue - r_3, magenta - r_4, black - r_5, yellow - r_6)

VIII. Conclusion

This paper proposes a framework under Petri net formalism, for which an algorithm provides a collision-free planning strategy for multi-robot systems. The team of agents receives a global LTL specification requiring to reach and/or avoid several regions of interest in the environment. The newly defined *Composed Petri net* model values the advantages of two models: (1) Robot Motion Petri net (RMPN) model for the environment, and (2) Büchi automaton of the LTL formula. The numerical evaluation of the proposed method demonstrates a computationally attractive solution (slower decrease of run time and smaller overall model) when compared with both the transition system and Petri net previous approaches.

References

[1] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[2] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multi-robot systems under LTL specifications," *IEEE Trans. on Robotics*, vol. 38, no. 2, pp. 1047–1062, 2021.

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

[4] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, pp. 343–352, 2019.

[5] M. H. Cohen and C. Belta, "Model-based reinforcement learning for approximate optimal control with temporal logic specifications," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[6] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dynamic Systems*, vol. 30, no. 1, pp. 55–79, 2020.

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

[8] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[9] J. Esparza, J. Křetínskỳ, and S. Sickert, "One theorem to rule them all: A unified translation of LTL into ω-automata," in *33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, 2018, pp. 384–393.

[10] 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.

[11] I. Hustiu, M. Kloetzer, and C. Mahulea, "Distributed path planning of mobile robots with LTL specifications," in *24th Int. Conf. on System Theory, Control and Computing (ICSTCC)*, 2020, pp. 60–65.

[12] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 5319–5325.

[13] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4817–4822.

[14] C. Banks, S. Wilson, S. Coogan, and M. Egerstedt, "Multi-agent task allocation using cross-entropy temporal logic optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7712–7718.

[15] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*. Springer, 2018, pp. 253–267.

[16] P. Tabuada, G. J. Pappas, and P. Lima, "Motion feasibility of multi-agent formations," *IEEE Trans. on Robotics*, vol. 21, no. 3, pp. 387–392, 2005.

[17] E. Montijano and C. Mahulea, "Probabilistic Multi-Robot Path Planning with High-Level Specifications using Petri Net Models," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 2188–2193.

[18] H. Costelha and P. Lima, "Robot task plan representation by Petri nets: modelling, identification, analysis and execution," *Autonomous Robots*, vol. 33, no. 4, pp. 337–360, 2012.

[19] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, p. 103289, 2019.

[20] 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.

[21] 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 Transactions on Automatic Control*, vol. 51, pp. 938–948, 2006.

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

[23] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

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

[25] 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.

[26] G. Holzmann, *The Spin Model Checker, Primer and Reference Manual*. Reading, Massachusetts: Addison-Wesley, 2004.

[27] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *13th Conference on Computer Aided Verification (CAV'01)*, H. C. G. Berry and A. Finkel, Eds., no. 2102, 2001, pp. 53–65.

[28] P. Wolper, M. Vardi, and A. Sistla, "Reasoning about infinite computation paths," in *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, E. N. et al., Ed., Tucson, AZ, 1983, pp. 185–194.

[29] E. Vitolo, C. Mahulea, and M. Kloetzer, "A computationally efficient solution for path planning of mobile robots with boolean specifications," in *ICSTCC'2017: 21st International Conference on System Theory, Control and Computing*, Sinaia, Romania, 2017, pp. 63–69.

[30] S. F. Roselli, P.-L. Götvall, M. Fabian, and K. Åkesson, "A compositional algorithm for the conflict-free electric vehicle routing problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1405–1421, 2022.

[31] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows*. John Wiley & Sons, 2011.

[32] R. González, C. Mahulea, and M. Kloetzer, "A matlab-based interactive simulator for mobile robotics," in *IEEE CASE'2015: Int. Conf. on Autom. Science and Engineering*, 2015.