On Multirobot Path Planning Based on Petri Net Models and LTL Specifications

Sofia Hustiu $^\diamond$, Cristian Mahulea $^\diamond$, Senior Member, IEEE, Marius Kloetzer $^\diamond$, and Jean-Jacques Lesage $^\diamond$

Abstract—This article proposes a method exploiting the advantages of Petri net (PN) and Büchi automata models by joining them in a newly defined composed PN 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—Autonomous robots, formal methods, high-level specifications, Petri nets.

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. Tumova and Dimarogonas [1] proposed 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 graphlike topology for the motion of the entire team, and an invariant model w.r.t. number of robots. Kloetzer and Mahulea [6] aimed 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 in [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.

1) Defining the composed PN, which joins the multiagent 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.

2) Providing a sound algorithm ensuring collision-free trajectories that accomplish the LTL mission.

3) 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 rest of this article is organized as follows. The article captures the challenges in the field of robot motion planning based on related work in Section II; asserts the problem statement in Section III; Seizes the mathematical notation in Section IV; and defines the contributions of the new proposed framework in Sections V and VI. The effectiveness of the current work is assessed through numerical simulations and comparisons in Section VII. Finally, Section VIII concludes this article.

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 BA [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, e.g., using the rapidly exploring random graph (RRG) algorithm applied to an incremental transition system modeling the robot [13], and for multiagent systems, e.g.,
computes an accepted run in the BA, and automata-based approaches, while other works focus on cooperation of agents ensuring local LTL tasks.

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 1) the environment, 2) the changes in the workspace, and 3) the task plan based on the compositions of different types of events and actions toward the goal.

Lacerda and Lima [4] proposed 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 BA is satisfied. The main idea in [6] computes an accepted run in the BA, 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 2-D 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 $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$. Furthermore, we assume that the environment is partitioned into a set of regions called cells, e.g., using a cell decomposition method [3], [19]. Each cell belongs entirely to one or more ROI, or it is included in the free space. The partitioned workspace can be further handled by a PN model, which is invariant w.r.t. the number of robots.

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 $h: P \rightarrow 2^Y$. If the cell $p_i$ is included in the intersection of ROIs $y_i$ and $y_k$, then $h(p_i) = \{y_i, y_k\}$, while if $p_i$ lies in the free space, then $h(p_i) = \emptyset$.

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 Definition 4.1. Given a global LTL specification over the set $Y$, 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:

\[
\varphi = \Diamond (y_1 \land y_2 \land y_3) \land \neg (y_1 \lor y_2) U (y_1 \land y_2)
\]  

IV. NOTATIONS AND DEFINITIONS

Definition 4.1 ([3]): An RMPN system is a tuple $Q = \langle N, m_0, Y \rangle$, where $N = \langle P, T, \text{Post}, \text{Pre} \rangle$ is a PN, $m_0$ is the initial marking, where $m_0[p]$ gives the number of robots initially deployed in cell $p \in P; Y \cup \{\emptyset\}$ is the set containing the output symbols; $h: P \rightarrow 2^Y$ 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 $\text{ROI}(h(p_i))$ is (are) visited. The tuple $N$ is composed of a set of places (one for each cell) $P$; a set of transitions $T$ modeling the movement of the robots; $\text{Post}, \text{Pre} \in \{0, 1\}^{|P| \times |Y|}$ is the coincidence, respectively, preincidence matrix, defining the arcs from transitions to places and vice versa, e.g., $\text{Pre}[p, t] = 1$ if $p \in P$ is connected with place $t \in T$, otherwise $\text{Pre}[p, t] = 0$.

Example 4.2: Let us consider the environment in Fig. 1(a), with three ROIs ($Y = \{y_1, y_2, y_3\}$) and two robots, initially located in $p_{2h}$ and $p_{20}$. Therefore, $h(p_{1h}) = h(p_{23}) = y_1, h(p_{17}) = h(p_{18}) = h(p_{24}) = h(p_{26}) = y_2, h(p_4) = h(p_{10}) = y_3, h(p_{13}) = \{y_1, y_2\}$, 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 $Y$ and the observation map $h$ are given above. The initial marking $m_0$ is a vector of dimension 26 having all elements equal to zero except $m_0[p_{2h}] = m_0[p_{20}] = 1$.

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 $C = \text{Post} - \text{Pre}$, the firing of a transition $t_j$ corresponds to the movement of a robot from an input place toward an output place w.r.t. to $t_j$, by applying a control law

---

$^1$By definition, RMPN systems considered in this article belong to the class of state-machine, i.e., each transition has one input and one output place.
that drives the robot from associated input and output cells. There exist approaches to designing such continuous laws in specific scenarios [20], [21]. We will be interested in finding sequences of transitions to be fired such that the team fulfills a given specification. If an RMPN marking \( \mathbf{m} \) can be reached from \( \mathbf{m} \) through a finite sequence of transitions \( \sigma \), we denote by \( \sigma \in \mathbb{N}_{>0}^T \) 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

\[
\mathbf{m} = \mathbf{m} + \mathbf{C} \cdot \sigma. \tag{2}
\]

A firing vector \( \sigma \) can be found, having a minimum number of transitions that drives the live\(^2\) RMPN to the desired marking \( \mathbf{m} \), i.e., by solving the optimization problem \( \min \mathbf{1}^T \cdot \sigma \). 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 [22] containing recursive formulae defined over a set of atomic propositions \( \mathcal{Y} \), by using Boolean operators (¬ - negation, ∧ - conjunction, ∨ - disjunction, ⇒ - implication, and ⇔ - equivalence) and some temporal operators (\( \mathcal{U} \) - until, ◯ - eventually, and □ - always), while excluding \( \bigcirc \) - the next operator. For simplicity of notation, we write LTL instead of LTL-\( X \) [22], [23], [24].

Any LTL formula can be automatically transformed into a nondeterministic BA using the available tools [25], [26]. The property of LTL being close under stuttering [26] 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 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); \( \Sigma_B \) 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 \( \pi(l, s) \) 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 \( \pi_{\mathcal{Y}} \). Alternatively, the input can be expressed as a combination of active observations from the set \( 2^3 \), where \( \emptyset \) represents the free space.

An infinite accepted run in \( B \) drives the automaton toward a final state from an initial state through 1) prefix and 2) suffix (path toward the same final state reached by the prefix). The run can be written as prefix, suffix, suffix... [27].

**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 \( 2^3 \). On the other hand, Fig. 2(b) illustrates the Büchi PN model, having the Boolean formula of transitions represented by red color. This model will be described in Section V. An accepted run that satisfies the formula could be \( s_1, s_2, s_3, s_4, \ldots \) with the prefix \( s_1 \) and the suffix \( s_4 \). Note that the unique final state \( s_4 \) 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 \( p_6 \). Clearly, entering \( p_{13} \) directly is not possible, since it would require first activating \( y_1 \) or \( y_2 \) and would violate \( \varphi(1) \).

A possible run generated by the robots is \( s_1, s_2, s_3, s_4, \ldots \) with prefix \( s_1, s_2, \) and suffix \( s_4 \). For this, robots should first enter \( y_1 \) and \( y_2 \) synchronously (generating \( \pi(s_1, s_2) = y_1 \land y_2 \)), then one robot should go to \( p_{13} \) (the intersection of \( y_1 \) and \( y_2 \)). Finally, a robot should enter \( p_4 \) 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 this article, using \( \cdot \) as a placeholder for components of various representations.

### Table I

| Notation | Description |
|----------|-------------|
| \(<\cdot\,>\) | Notation for quotient RMPN (Substep 1.1) |
| \(<\cdot\,B\) | Notation for Büchi RMPN (Substep 1.2) |
| \(<\cdot\,C\) | Notation for composed Petri net (Substep 1.3) |

V. **Solution Step 1: Composed PN Model**

Fig. 3 provides a general overview of the entire proposed solution of the Problem 3.1. The first phase is responsible for computing the newly defined composed PN model based on a reduced representation of the environment (substep 1.1) and the Büchi automaton associated with the LTL specification (substep 1.2). The second phase focuses on an iterative algorithm that requires two actions, expressed by substeps 2.1 and 2.2.

**Substep 1.1. Quotient of the RMPN in Definition 4.1:** Given an RMPN system \( Q \) as in Definition 4.1, the idea of obtaining the quotient \( Q^M \) is to iteratively combine any \( p_i \) and \( p_j \) places of \( P \) that satisfy \( p_j \in (p_i^*)^* \) and \( h(p_i) = h(p_j) \). This reduction technique is
Algorithm 1: Büchi Petri net.

```
Input: B = (S, S₀, Σ_B, →, F)
Output: Q^B = (\{P^B, T^B \cup T^V, \text{Pre}^B \text{Pre}^V, \text{Post}^B \text{Post}^V\}, m^0_B, Y, h)
1. Let P^B = \{p^B_1, p^B_2, \ldots, p^B_{|B|}\} be the set of |S| places;
2. Let T^B = T^V = \emptyset, \text{Pre}^B = \emptyset, \text{Pre}^V = \emptyset, \text{Post}^V = \emptyset;
3. forall \{s_i, \tau, s_j\} \in →_B do
   4. forall conjunctive element \alpha_k of π(s_i, s_j) do
      5. T^B = T^B \cup t_{\alpha_k};
      6. Add a new column to \text{Pre}^B and to \text{Post}^B corresponding to t_{\alpha_k};
      7. \text{Pre}^B[p^B_{i}, t_{\alpha_k}] = \text{Post}^B[p^B_{i}, t_{\alpha_k}] = 1;
8. Let m^0_B = 0^{S \times 1};
9. m^0_B[p^B_1] = 1;
forall s_f \in F do
10. T^V = T^V \cup t_{s_f};
11. Add a new column to \text{Pre}^V and to \text{Post}^V corresponding to t_{s_f};
12. \text{Pre}^V[p^V_{i}, t_{s_f}] = \text{Post}^V[p^V_{i}, t_{s_f}] = 1;
```

with \( p^B_i \) associated with \( s \in S_0 \). Observe that one token in Büchi PN is sufficient to mark the current state of the automaton \( B \), since \( |S_0| = 1 \). Algorithm 1 also returns virtual transitions \( T^V \), by adding on \( t^V_i \in T^V \) to each final state Büchi PN, expressed in \( \text{Pre}^V \) and \( \text{Post}^V \). 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 BA in Fig. 2(a). Applying Algorithm 1, the Büchi PN in Fig. 2(b) is obtained. In this example, the places \( p^B_2 \) and \( p^B_5 \) are connected with a single transition corresponding to input \( \pi(s_2, s_3) = y_1 \land y_2 \land y_3 \), visualized in red. Fig. 2(b) illustrates the addition of the virtual transition \( t^V_1 \) connected to the final state \( s_3 \) by a reading (bidirectional) arc.

Substep 1.3. Composition of quotient RMPN and Büchi PN systems:
Algorithm 2 includes the full strategy to return the composed PN system \( Q^C \), by the composition of the robotic team model \( Q^M \) with the one of the specification \( Q^B \). For this, a number of \( 2 \times |Y| \) places are needed, from which half models the active observations, given by the set \( P^O \), while the other \( |Y| \) places model inactive observations \( P^I \). Initially, places \( p^O_i \) have zero tokens, while places \( p^O_i \) have the number of tokens equal to \( |R| \), denoting no active observation in the initial marking line (3). The sum of the tokens in \( p^O_i \) and \( p^I_i \) is always equal to \( |R| \).

The lines 1–2 define the sets of places, respectively, transitions for the composed PN \( Q^C \).

Matrices \( \text{Pre}^C \) and \( \text{Post}^C \) are initialized in lines 4–5. Lines 6–10 are executing for each observation \( y_i \). Input arcs are added to place \( p^O_i \) for each \( p_k \in P^M \), \( y_i \in h(p_k) \). Additionally, arcs from \( p^O_i \) 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 \( p^O_i \). If \( m[p^C_i] > 0 \) then observation \( y_i \) is active. The place \( p^O_i \) is the complementary place of \( p^O_i \), therefore is connected with the same transitions as \( p^O_i \) but with arcs oriented in the other sense. If \( m[p^C_i] = |R| \) then observation \( y_i \) is not active. Finally, loop in lines 11–17 connects the places \( p^O_i \), respectively, \( p^O_i \) with the transitions \( t^B_i \) by a reading arc, according to the assigned Boolean formula (conjunction), considering weights 1, respectively, \( |R| \).

Fig. 4 depicts a part of composed PN with its initial marking, returned by Algorithm 2. For the sake of clarity, Fig. 4 considers only the arcs for one region on interest \( y_3 \). When a transition in \( Q^M \) fires and the observation changes, the \( Q^M \) deposits one token to the respective active observation, e.g., \( t^M_2 \) 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 \( p^I_2 \) and \( p^O_i \). In \( Q^B \), the transitions are fired based on the assigned Boolean formula, being triggered by the reading arcs of \( t^O_i \), \( t^O_i \), e.g., \( t^O_i \) and synthesized in [28], and the reduced PN model \( Q^M \) has the property that its output is changed when a transition is fired. Thus, after firing one transition in \( Q^M \), one transition in Büchi should also be fired. A transition of \( Q^M \) corresponds to a set of trajectories in \( Q \).

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 \( Q^M \) in Fig. 1(b) is obtained, also being an RMPN model according to Definition 4.1. The correspondence between the reduced and the original RMPN models of the environment is captured by a projection matrix \( P_r \) with size \( 5 \times 26 \), having all elements equal to zero except adjacent places sharing the same observation, e.g., \( P_r[p^M_{35}, p^M_i] = P_r[p^M_{38}, p^M_i] = 1 \).

Substep 1.2. Büchi PN: Starting from the BA \( B = (S, S_0, \Sigma_B, \rightarrow, F) \) as in Definition 4.3, Algorithm 1 obtains the corresponding Büchi PN system \( Q^B \). For each state \( s_i \in S \), a new place \( p^B_{i} \) 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 BA. The second loop (lines 8–17) is executed for each conjunctive element \( \alpha_k \) of \( \pi(s_i, s_j) \), and adds a new transition \( t_{\alpha_k} \) to the Büchi PN from \( p^B_{i} \) to \( p^B_{j} \). 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,
Algorithm 2: Composed Petri net system.

Input: $P^O$, $P^O_-$, $Q^M$

Output: $Q^C = (PC^C, TC^C, Pre^C, Post^C, m^C_0, \gamma, h)$

1. Let $PC^C = (P^M \cup P^B \cup P^O \cup P^O_-)$;
2. Let $TC^C = (T^M \cup T^B \cup T^V);$
3. $m^C_0 = [m^M_0, m^B_0, m^O_0];$
4. Let $Pre^C = [Pre^M | Pre^B | Pre^V];$
5. Let $Post^C = [Post^M | Post^B | Post^V];$
6. forall $y_i \in \gamma$ do
   7. Let $p^r = \{ p \in D^M | y_i \in h(p) \};$
   8. forall $p_k \in p^r$ do
      9. $Post^C[p^O, p_k] = Pre^C[p^O, p_k] = 1；$
      10. $Pre^C[p^O, p_k] = Post^C[p^O, p_k] = 1；$
   11. forall $t^B_\tau \in T^B$ do
      12. Let $\pi_1$ be the DNF formula assigned to $t^B_\tau$;
      13. if $\pi_1 \neq \top$ then
         14. forall atomic propositions $y_i$ appearing not negated in $\pi_1$ do
            15. $Pre^C[p^O, t^B_\tau] = Post^C[p^O, t^B_\tau] = 1；$
      16. forall atomic propositions $y_i$ appearing negated in $\pi_1$ do
         17. $Pre^C[p^O, t^B_\tau] = Post^C[p^O, t^B_\tau] = |R|；$

Global algorithm: Algorithm 3 computes robots’ movements while ensuring the mission $\varphi$. The algorithm stops when a set of trajectories is returned for a place $p_f_i \in Set_f$, with $Set_f$ modeling the final states in $B$ (line 15). The main idea is to first search for a feasible run in $Q^C$ (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 $p_f_i$ is included in its self-loop ($p^O_j$ 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).

Substep 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 $m^C_{k_f}$). The MILP is solved individually for prefix and suffix considering only the transitions in $Q^M$. Line 17 appends these solutions $\sum_{i=1}^k \sigma^M_i$ of prefix (subscript $p$), respectively, for suffix (subscript $s$) to sets of bad solutions $CE_{p} \cup CE_{s}$ for both prefix and suffix. 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 $k$.

VI. SOLUTION STEP 2: ALGORITHM

The robot navigation algorithm requires two actions to be achieved: 1) compute a solution on the reduced model $Q^C$; 2) project the searched solution in the original PN of the environment, outputting the multirobot 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 [29], being here applied to a PN-based approach, rather than a graph-based approach.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
1) \( \mathbf{m}_i^C = [\mathbf{m}_i^M, \mathbf{m}_i^B, \mathbf{m}_i^O] \) - marking column vector at step \( i \) of \( Q_C^m \) composed by the marking of Quotient PN (\( \mathbf{m}_i^M \)), Büchi PN (\( \mathbf{m}_i^B \)), active observation places (\( \mathbf{m}_i^O \)), and inactive observation places (\( \mathbf{m}_i^{O-} \)).

2) \( \sigma_j^P \) is a firing vector at step \( j \) of the composed PN, composed by the firing vector of Quotient PN (\( \sigma_j^M \)), Büchi PN (\( \sigma_j^B \)) and of virtual transitions (\( \sigma_j^V \)).

3) \( z^1, z^2 \in \{0, 1\}^{\sum k_i \sigma_j^M} \) - binary vectors with \( z^1[j] = 1 \) if \( \zeta - \sum_k \sigma_i^M \geq 1 \), and \( z^2[j] = 0 \) otherwise, respectively, \( z^3[j] = 1 \) if \( \zeta - \sum_k \sigma_i^M \leq 1 \), otherwise \( z^3[j] = 0 \), with \( \zeta \in CE_{P_{i,j}} \).

Objective:

\[
\min \sum_{i=1}^{2} i \cdot (1^T \cdot \sigma_i^M + 1^T \cdot \sigma_i^B) \quad (3a)
\]

Constraints:

\[
m_i^C - m_{i-1}^C - C^C \cdot \sigma_i^C = 0, \quad i = 1, 2, k \quad (3b)
\]

\[
m_i^C = Pr \Sigma \sigma_i^C \geq 0, \quad i = 2, j + 1 \quad j = 0, k - 1
\]

\[
1^T \cdot \sigma_i^B = 0, \quad i = 2, j \quad j = 2, k
\]

\[
1^T \cdot \sigma_i^V = 0, \quad i = 2, j \quad j = 2, k
\]

\[
m_i^C | \{p_{ij}^B\} = 1, \quad i = 2, k \quad (3f)
\]

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 \((h(p_t^F) \neq \emptyset)\) that, such as following the step \((i.e. \text{when } i \text{ is even}) \) one transition in Büchi should be fired; (3d) imposes firing of 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 \( p_{ij}^B \); (3g) and (3h) guarantee that the current solution is different than any other previous bad solution \( \zeta \) from set \( CE_{P_{i,j}} \), using a big number (%N method [30]).

Substep 2.2. Projecting the solution: Let \( M = \{m_1^M, m_2^M, \ldots, m_{2k}^M\} \) be the sequence of markings returned by MILP (3). Successive identical markings are removed. Let us add \( m_0^M \) as the first element in \( M \). Furthermore, let \( G = \{g_1, g_2, \ldots, g_{2k}\} \) be a sequence of 2 \( k \) vectors s.t. \( g_i \in \{0, 1\}^{3M} \) with \( g_i[j] = 1 \) if \( m_i^M[j] = 0 \), and \( g_i[j] = 0 \) 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); \( Pr \) - projection matrix between \( Q^B \) and RMPN; \( C \) - token flow matrix of the RMPN; \( Pr \Sigma \), \( P_{o} \) - pre/postincidence matrices of RMPN; \( m_i^M \) - marking at step \( i \) of \( Q^M \).

Variables: \( m_{i,j} \) - marking at step \( i \) of RMPN, considering the intermediate marking \( j \); \( \sigma_{i,j} \) - firing vector at step \( i \) of RMPN, for the intermediate marking \( j \) with \( i = 0, |M|; j = 1, |R| + 1 \).

Objective:

\[
\min 1^T \cdot \sum_{i,j} \sigma_{i,j} \quad (4a)
\]

Constraints:

\[
m_{i,j} - m_{i,j-1} - C \cdot \sigma_i = 0, \quad i = 0, |M|; j = 1, |R| + 1 \quad (4b)
\]

\[
m_{i,0} - m_{i-1,|R|+1} = 0, \quad i = 1, |M| \quad (4c)
\]

\[
Pr \Sigma \cdot m_{i,j} = 0, \quad i = 0, |M|; j = |R| \quad (4d)
\]

\[
Post \{g_i, Pr_i\} \cdot \sigma_{i,j} = 0, \quad i = 0, |M|; j = |R| + 1 \quad (4e)
\]

\[
Post \{\sigma_{i,j} + m_{i,j-1} \leq 1, \quad i = 0, |M|; j = |R| + 1 \quad (4f)
\]

\[
m_{i,|R|+1} - Pr \Sigma \cdot \sigma_{i,|R|+1} \geq 0, \quad i = 0, |M| \quad (4g)
\]

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 \( m_{i,1} \) to \( m_{i,|R|} \), since the \( |R| \) intermediate markings are introduced to avoid collisions between \( m_{i,1} \) and \( m_{i,|R|} \); (4e) ensure that the corresponding firing vectors of the intermediate markings from \( m_{i,1} \) to \( m_{i,|R|} \) should not actuate 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 \( m_{i,1} \) to substep \( m_{i,|R|+1} \) is done synchronously by all robots (each robot fires only one transition) such that the generated observation of \( Q^M \) 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 \( Q^C \) and \( Q \), and a set of binary variables of size \( 2^{|M|} \cdot |CE_{P_{i,j}}| \), 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: Algorithm 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 [31], using the CPLEX Optimizer solver for both MILPs, while herein the results were obtained on a laptop with i7 - 8th gen. CPU at 2.20 GHz and 8 GB 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 ROI at the same time, ensuring \( y_1 \) and \( y_2 \) are reached simultaneously. Fig. 1(a) illustrates the robot trajectories returned by Algorithm 3. It can be observed that both robots move toward regions \( y_1 \) and \( y_2 \) to complete the concurrent requirement. Afterward, \( n_1 \) advances and enters the last region of interest \( y_2 \), while the second robot enters \( n_2 \). For this result, the quotient PN model with five places and ten transitions.
was computed in 0.02 s, and the composed PN model with 14 places and 16 transitions (out of which one is a virtual one) was computed in 0.02 s. MILP (3) reaches the final state in Büchi in 0.05 s, 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 $\varphi = \square(\Diamond y_{1} \land \Diamond y_{3} \land \Diamond y_{5} \land \Diamond y_{6} \land \Diamond y_{7} \land \Diamond y_{8}) \land \neg(\Diamond y_{3} \lor y_{5}) \land (y_{5} \land y_{6}) \land \neg(\Diamond y_{3} \lor y_{7}) \land (y_{7} \land y_{8})$. This specification imposes the visit of several ROIs in an environment with 8 ROI, while the regions $y_{5}$ and $y_{6}$ are simultaneously reached, respectively, $y_{3}$ and $y_{7}$. Fig. 5 exemplifies robot’s trajectories for six robots, with black stars being represented the synchronization points of the team necessary when fulfilling $\varphi$.

Table II contains a result analysis of the current approach of Algorithm 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 BA. 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 $|P^{B}N| + |P^{B}G| + 2 \cdot |Y|$. On the other hand, the model returned by the procedure following Büchi contains the number of places given by the RMPN $Q$, while approaches based on PS [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), $11^{40} \times 10$ (quotient TS), as 11, respectively, 10, represent the number of states in the reduced transition system, respectively, in BA.

### 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                | 9.36               | 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                   |

### VIII. Conclusion

This article proposes a framework under PN formalism, for which an algorithm provides a collision-free planning strategy for multirobot systems. The team of agents receives a global LTL specification requiring to reach and/or avoid several ROI in the environment. The newly defined composed PN model values the advantages of two models: 1) robot motion PN (RMPN) model for the environment; and 2) BA 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 PN 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 multirobot systems under LTL specifications,” *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1047–1062, Apr. 2022.

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

[4] B. Lacerda and P. U. Lima, “Petti net based multi-robot task coordination from temporal logic specifications,” *Robot. Auton. Syst.*, vol. 122, pp. 343–352, 2019.

[5] M. H. Cohen and C. Beta, “Model-based reinforcement learning for approximate optimal control with temporal logic specifications,” in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. 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 Dyn. Syst.*, vol. 30, no. 1, pp. 55–79, 2020.

[7] X. C. Ding, M. Kloetzer, Y. Chen, and C. Beta, “Automatic deployment of robotic teams,” *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 75–86, Sep. 2011.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[9] J. Espanz, J. Kříž, and S. Sickenet, “One theorem to rule them all: A unified translation of LTL into $\omega$-automata,” in *Proc. 33rd Ann. ACM/IEEE Symp. Log. Comput. Sci.*, 2018, pp. 384–393.

[10] M. Guo and D. V. Dimarogonas, “Multi-agent plan reconfiguration under local LTL specifications,” *Int. J. Robot. Res.*, 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 Proc. 24th Int. Conf. Syst. Theory, Control Comput., 2020, pp. 60–65.

[12] E. M. Wolff, U. Topcu, and R. M. Murray, “Optimization-based trajectory generation with linear temporal logic specifications,” in Proc. IEEE Int. Conf. Robot. Autom., 2014, pp. 5319–5325.

[13] C. I. Vasile and C. Belta, “Sampling-based temporal logic path planning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 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 Proc. IEEE Int. Conf. Robot. Autom., 2020, pp. 7712–7718.

[15] P. Schillinger, M. Bürger, and D. V. Dimarogonas, “Decomposition of finite LTL specifications for efficient multi-agent planning,” in Distrib. Auton. Robot. Syst.: 13th Int. Symp., 2018, pp. 253–267.

[16] P. Tabuada, G. J. Pappas, and P. Lima, “Motion feasibility of multi-agent formations,” IEEE Trans. Robot., vol. 21, no. 3, pp. 387–392, Jun. 2005.

[17] E. Montijano and C. Mahulea, “Probabilistic multi-seat path planning with high-level specifications using Petri net models,” in Proc. IEEE 17th Int. Conf. Autom. Sci. Eng., 2021, pp. 2188–2193.

[18] H. Costelha and P. Lima, “Robot task plan representation by Petri nets: Modelling, identification, analysis and execution,” Auton. Robots, vol. 33, no. 4, pp. 357–360, 2012.

[19] H. Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations. Cambridge, MA, USA: MIT Press, 2005.

[20] 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. Autom. Control, vol. 51, no. 6, pp. 938–948, Jun. 2006.

[21] C. Belta and L. Habets, “Controlling a class of nonlinear systems on rectangles,” IEEE Trans. Autom. Control, vol. 51, no. 11, pp. 1749–1759, Nov. 2006.

[22] C. Baier and J.-P. Katoen, Principles of Model Checking. Cambridge, MA, USA: MIT Press, 2008.

[23] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas, “Symbolic planning and control of robot motion,” IEEE Robot. Autom. Mag., vol. 14, no. 1, pp. 61–71, Mar. 2007.

[24] 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.

[25] G. Holzmann, The Spin Model Checker: Primer and Reference Manual. Boston, MA, USA: Addison-Wesley, 2004.

[26] P. Gastin and D. Oddoux, “Fast LTL to Büchi automata translation,” in Proc. 13th Conf. Comput. Aided V erification, H. C.G. Berry and A. Finkel, Eds., 2001, pp. 53–65.

[27] P. Wolper, M. Y. Vardi, and A. P. Sistla, “Reasoning about infinite computation paths,” in Proc. 24th Annu. Symp. Found. Comput. Sci., 1983, pp. 185–194.

[28] E. Vitolo, C. Mahulea, and M. Kloetzer, “A computationally efficient solution for path planning of mobile robots with Boolean specifications,” in Proc. 21st Int. Conf. Syst. Theory, Control Comput., Sinaia, Romania, 2017, pp. 63–69.

[29] S. F. Roselli, P.-L. Götswall, M. Fabian, and K. Åkesson, “A compositional algorithm for the conflict-free electric vehicle routing problem,” IEEE Trans. Autom. Sci. Eng., vol. 19, no. 3, pp. 1405–1421, Jul. 2022.

[30] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, Linear Programming and Network Flows. Hoboken, NJ, USA: Wiley, 2011.

[31] R. González, C. Mahulea, and M. Kloetzer, “A Matlab-based interactive simulator for mobile robotics,” in Proc. IEEE Int. Conf. Autom. Sci. Eng., 2015, pp. 310–315.