Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-28 (2022) 9–15

Optimal Multi-Robot Path Planning for Cyclic Tasks using Petri Nets [⋆]

Peng Lv [*] Guangqing Luo [*] Xiang Yin [*] Ziyue Ma [**] Shaoyuan Li [*]

[*] *Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: {lv-peng,luogq11,yinxiang,syli}@sjtu.edu.cn).*

[**] *School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: maziyue@xidian.edu.cn).*

Abstract: In this paper, we investigate the problem of optimal multi-robot path planning for a cyclic task represented by a particular type of linear-temporal logic (LTL) formulae. Specifically, the team of robot needs to fulfill a given LTL formula and at the same time, accomplishes some particular tasks *infinitely often*. To avoid the state-space explosion when the number of robot increases, we use Petri nets to model the team of multi-robot. Our goal is to find an optimal infinite sequence in the *prefix-suffix form* for each robot such that *the average cost per task* is minimized. We propose an efficient planning method based on the notion of basis reachability graph (BRG), which is a compact representation of the reachability space of the PN. We demonstrate the computational efficiency of our method through illustrative examples.

Keywords: Petri Nets, Linear Temporal Logic, Multi-Robot Systems, Task Planning

1. INTRODUCTION

Multi-robot systems are widely used in many applications such that autonomous warehouses, environment surveillance and information gathering. Traditionally, researches on multi-robot path planning mainly focus on finding trajectories for each single robot to meet some low-level requirements such as collision avoidance or reaching some target regions according to the physic dynamics of robots. In the recent years, considerable attentions have been paid on multi-robot planning for *high-level tasks* described by complex temporal logic formulae Kress-Gazit et al. (2009); Liu et al. (2022); Shi et al. (2022); Yang et al. (2020); Yu et al. (2022).

Linear temporal logic (LTL) is a wide used language in expressing formal requirements. In the past years, considerable attentions have been paid to robot planning for LTL tasks; see, e.g., Guo and Dimarogonas (2015); Smith et al. (2011); Ulusoy et al. (2014). In the context of multi-robot LTL path planning, the main challenge is the the curse of dimensionality since the overall state-space grows exponentially fast when the number of robots increases. To resolve this issue, many different approaches have been proposed in the literature to mitigate the computational complexity. For example, in Kantaros and Zavlanos (2018), sampling-based approach is used for LTL planning without constructing the entire state-space. Distributed coordination algorithms are also developed such that the overall task can be achieved with or without communications Schillinger et al. (2018); Yu and Dimarogonas (2021).

The aforementioned works are mainly based on the automata model of the robots. In contrast, Petri nets (PN) is a more efficient model for representing concurrent systems. In particular, it provides a more compact representation of the synchronization of a set of sub-systems without enumerating the entire state. Therefore, PN model is particular suitable for modeling multi-robot systems. In the context of LTL planning, PN-based approaches have been investigated in the literature recently, which provides a promising way to mitigate the computation complexity He et al. (2022); Mahulea et al. (2020). For example, in Mahulea and Kloetzer (2018), the authors investigate multi-robot path planning for Boolean task using PNs. The approach was further extended by Kloetzer and Mahulea (2020) to handle general LTL tasks. However, the result in Kloetzer and Mahulea (2020) only focuses on the feasibility of the LTL task, and the optimality of the synthesized plan is not guaranteed.

In this paper, we also investigate the LTL planning problem for multi-robot systems represented by Petri nets. We focus on a particular type of LTL formulae, where the robot needs to accomplish a finite task while achieving a cyclic task infinitely often. In contrast to existing works, where only qualitative LTL requirement is considered, here we further consider an optimal planning problem with quantitative performance. Specifically, we consider an optimality metric called the *average cost per task*, which was proposed in our previous work Lv et al. (2021). Our objective is to find a plan for the team of robot such that (i) the LTL task is satisfied; and (ii) the cost for each task cycle is minimized.

In order to solve the optimal LTL path planning problem, we use the structure of *basis reachability graph* (BRG), which is a compact representation of the reachable space of the PN without enumerating the entire state space Ma et al. (2016, 2021). Specifically, we use Büchi automaton to represent the LTL specification and construct a product PN based on the original PN and the Büchi automaton. Then by constructing the BRG of the product PN, we show that an optimal path in the prefix-suffix form can be effectively synthesized. We also

[⋆] This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61803259) and by the National Key Research and Development Program of China (2018AAA0101700).

discuss how to further mitigate the synthesis complexity using the structural property of the PN. Our experimental results show that the proposed PN-based planning algorithm is more scalable compared with the standard automata-product-based approach for multi-robot systems.

2. PRELIMINARY AND PROBLEM FORMULATION

2.1 Petri Net Model of Multi-Robot System

We consider a team of identical robots moving in the same workspace that contains a set of regions with connecitivity constraints. In this work, the connectivity and properties of the workspace is modeled as a weighted PN

$$\mathcal{Q} = (P, T, Pre, Post, \Pi, h, g),$$

where P is a set of m *places*; T is a set of n *transitions*; $Pre : P \times T \to \{0, 1\}$ and $Post : P \times T \to \{0, 1\}$ are *pre-* and *post-*incidence functions, respectively, which can also be considered as matrices; Π is a set of *atomic propositions*; $h : T \to 2^{\Pi}$ is a labeling function that assigns each transition a set of atomic propositions; and $g : T \to \mathbb{N}^+$ is a weight function assigns each transition an integer. The *incidence matrix* is defined by $C = Post - Pre \in \mathbb{N}^{m \times n}$. For a transition $t \in T$, we use ${}^\bullet t$ and $t^\bullet$ to denote its *input places* and *output places*, respectively. Input transitions ${}^\bullet p$ and output transition $p^\bullet$ are defined analogously. We say $\mathcal{Q}$ is a *state machine (SM)* if $|{}^\bullet t| = |t^\bullet| = 1, \forall t \in T$. In this paper, the environment is always modeled as a SM.

In robot planning, each place represents a region in the workspace and each transition represent the action of going to a region from another. We use Π to denote all basic properties of interest. Then for each transition $t \in T$, $h(t)$ denotes the set of atomic propositions that hold at this unique output place. Therefore, we require that $\forall p \in P, \forall t_1, t_2 \in T : t_1, t_2 \in {}^\bullet p \Rightarrow h(t_1) = h(t_2)$. We denote by P_Π the set of places with non-empty propositions. The labeling function is also extended to sequences by $h(\sigma) = h(t_0)h(t_1)\cdots$, where $\sigma = t_0 t_1 \cdots$.

Each robot in the workspace is represented as a *token* in the PN. A *marking* $M : P \to \mathbb{N}$ is a vector that represents the distribution of robots in the workspace. We use M_0 to denote the initial distribution of robots and $M(p)$ is the number of robots at place p in marking M. A transition t is enabled at M if $M \geq Pre(\cdot, t)$ and a new marking $M' = M + C(\cdot, t)$ is reached when firing t. We use $M[\sigma\rangle M'$ to denote that M' is reachable from M by firing sequence $\sigma = t_0 t_1 \cdots t_n$. We denote by $R(\mathcal{Q})$ the set of all reachable markings from M_0. Given a sequence $\sigma = t_0 t_1 \cdots$, we call the resulting marking sequence $\rho_\sigma = M_0 M_1 \cdots \in (R(\mathcal{Q}))^\omega$ a *run* of $\mathcal{Q}$, where $M_{i+1} = M_i + C(\cdot, t_i), \forall i = 0, 1, \cdots$. Also, given a feasible run ρ, we use σ_ρ to denote the sequence generating ρ. We denote by $\Sigma(\mathcal{Q})$ the set of all feasible sequences in $\mathcal{Q}$. We use $\boldsymbol{y}_\sigma \in \mathbb{N}^T$ to denote the firing counting vector. We denote by P_0 the set of all places p such that $M_0(p) > 0$. We make the following mild assumption on the initial disbribution of robots: $P_0 \cap P_\Pi = \emptyset$, i.e., each robot starts from a region with no property.

2.2 Cyclic Task

Our general objective is to find a plan, which is an infinite sequence, for the team of robot such that it satisfies an LTL formula. The syntax of LTL formulae (without next) is given as

$$\phi = \top \mid \pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathsf{U} \phi_2 \tag{1}$$

where $\pi \in \Pi$ is an atomic proposition, $\neg$ (negation) and $\wedge$ (conjunction) are standard Boolean operators, and U (until) is a temporal operator. These operators also induce such as $\vee$ (disjunction), $\rightarrow$ (implication), $\Diamond$ (eventually) and $\Box$ (always). We say that ϕ is a co-safe LTL (scLTL) formula if negation is only applied in front of atomic proposition. The semantics of LTL formula ϕ is defined over infinite words on $(2^\Pi)^\omega$; the reader is referred to Baier and Katoen (2008) for details. Given a run ρ of $\mathcal{Q}$ and ϕ, we denote $\rho \vDash \phi$ if $h(\sigma_\rho)$ satisfies ϕ and use $\Sigma_\phi(\mathcal{Q})$ to denote the set of all sequences satisfying ϕ.

In this work, the objective of the robot is given as an LTL formula ϕ of the following form

$$\phi = \varphi \wedge \Box\Diamond\pi_{sur}, \tag{2}$$

where φ is an scLTL formula over 2^Π representing a finite-horizon task and $\pi_{sur} \in \Pi$ is a special proposition representing a cyclic task that should be satisfied *infinitely often*.

The objective LTL ϕ can be translated into a *Büchi automata* (BA) $\mathcal{A}_\phi = (Q, q_0, 2^\Pi, \delta, Q_F)$, where Q is a finite set of states, q_0 is the initial state, 2^Π is the set of all subsets of Π, $\delta : Q \times 2^\Pi \times Q$ is the transition function and Q_F is the set of all accepting states. In general, the BA can have multiple initial state. Here we assume that for ϕ, its corresponding BA $\mathcal{A}_\phi$ has only one initial state, which is restrictive but still expressive enough for most tasks. Then the BA $\mathcal{A}_\phi$ exactly accepts all infinite words satisfying ϕ in the sense that $\sigma \in \Sigma_\phi(\mathcal{Q})$ iff there exists a path from q_0 and visits Q_F infinitely under $h(\sigma)$ in $\mathcal{A}_\phi$.

2.3 Problem Formulation

Our objective is to find an infinite sequence in the following prefix-suffix form

$$\sigma = \sigma_{pre}(\sigma_{sur})^\omega$$

in $\mathcal{Q}$ that is optimal and satisfies ϕ. Regarding the optimality condition, we refer each satisfaction of π_{sur} as a task cycle. Then the cost of the infinite sequence $\sigma = \sigma_{pre}(\sigma_{sur})^\omega \in \Sigma(\mathcal{Q})$ with $\rho_\sigma \vDash \phi$ is defined as the *average cost per cycle* of σ as follows:

$$\mathsf{Cost}_{\mathcal{Q}}^{AveT}(\sigma) = \frac{\boldsymbol{g}^T \cdot \boldsymbol{y}_{\sigma_{sur}}}{N(\sigma_{sur})} \tag{3}$$

where $N(\sigma) = \sum_{t \in \mathbb{T}_{sur}} \boldsymbol{y}_\sigma(t)$ represents the number of visiting π_{sur} along σ, where $\mathbb{T}_{sur}$ is the π_{sur} *input transition set* defined as follows: $\mathbb{T}_{sur} = \{t \in T | \pi_{sur} \in h(t)\}$. Note that, since σ_{pre} is a finite sequence, it does not affect the infinite behavior of the system, and we do not consider it in (3). This leads to the *Multi-robot Optimal Path Planning Problem for Average Cost Per Task* (MOTP-AT) we solve in this work:

Problem 1. (MOTP-AT) Given a PN $\mathcal{Q}$ which models a team of identical robots moving in an environment with $\pi_{sur} \in \Pi$ being the atomic proposition which indicates some regions need to be surveilled infinitly often, and a temporal logic specification ϕ in form (2) for the team, find a optimal sequence σ for the team such that

- $\sigma^\star \in \Sigma_\phi(\mathcal{Q})$;
- $\sigma^\star$ is in prefix-suffix form, namely $\sigma^\star = \sigma^\star_{pre}(\sigma^\star_{sur})^\omega$;
- $\forall \sigma \in \Sigma_\phi(\mathcal{Q}) : \mathsf{Cost}_{\mathcal{Q}}^{AveT}(\sigma^\star) \leq \mathsf{Cost}_{\mathcal{Q}}^{AveT}(\sigma)$.

Remark 1. As shown in the above problem, part of our work is to plan the trajectory for the agent team meeting the specification ϕ. In the previous work (Kloetzer and Mahulea (2020)), this problem has been solved, but they do not consider the optimal control problem, that means obtaining the optimal trajectory which optimizes the surveillance operation cost.

3. SOLUTION

As we consider a bounded and conservative PN model, we can construct its reachability graph to search the optimal sequence. However, the state space is extremely large, which makes the method having no advantages on planning complexity compared with traditional product automaton based method. Therefore, we take an alternative method here based on a compact representation of the reachability graph called *basic reachability graph* to do the search.

Our solution can be mainly divided into four parts. In IV-A, we first create a product PN $\mathcal{Q}'$ between $\mathcal{Q}$ and BA $\mathcal{A}_\phi$. As $\mathcal{Q}$ is a SM, we simplify $\mathcal{Q}$ by abstracting those transitions with non-empty propositions, and acquire abstracted PN $\mathfrak{Q}$, which has less places compared to $\mathcal{Q}$ and can also be used to construct the product PN $\mathcal{Q}'$. Next, based on the product PN $\mathcal{Q}'$, we reformulate problem 1 as problem 2 and show the equivalence between these two problems in IV-B. In section IV-C, we briefly introduce the theory related to BRG and construct the BRG $\mathcal{B}'$ to represent the reachability set of $\mathcal{Q}'$ and prove that the deduction of $\sigma^\star_{\mathcal{Q}'}$ can be simplified to a prefix-suffix form trajectory planning problem between these basic markings in $\mathcal{B}'$. And in section IV-D, we propose the whole planning algorithm for the above three parts to get the solution.

3.1 Construction of product PN $\mathcal{Q}'$ between $\mathcal{Q}$ and $\mathcal{A}_\phi$

First, we need to recall some basic concepts about graph theory, which will be used later.

Recall that a *path* l in a *directed multi-graph* $\mathcal{G} = (V, E)$ is a sequence of vertices and edges $l = v_1e_1v_2e_2\cdots e_{n-1}v_n$ such that $v_i \in V, e_i \in \langle v_i, v_{i+1}\rangle \subseteq E, \forall i \geq 1$, where $\langle v_i, v_{i+1}\rangle$ is the set containing all edges from v_i to v_{i+1}. For a vertex $v \in V$, we define $\text{Succ}_\mathcal{G}(v) = \{v' \in V | \langle v, v'\rangle \subseteq E\}$ as the set of successor vertices of v. A path is said to be a *cycle* if $v_1 = v_n$. For a cycle, if $\forall 1 < i < j < n, v_i \neq v_j, v_i \neq v_1$ and $v_j \neq v_1$, we call it a *simple cycle*. Otherwise, it is a *compound cycle*. We use $L_\mathcal{G}, cyc_\mathcal{G}, Scyc_\mathcal{G}$ and $Ccyc_\mathcal{G}$ to denote the set containing all paths, cycles, simple cycles and compound cycles in $\mathcal{G}$ respectively. $\mathcal{G}$ is called a *finite graph* if $|V| \neq \infty$ and $|E| \neq \infty$.

Besides, when

- $\forall v_i, v_j \in V, [\langle v_i, v_{i+1}\rangle \subseteq E] \Rightarrow [|\langle v_i, v_{i+1}\rangle| = 1]$,

$\mathcal{G}$ is called a *directed simple graph* and we use $l = e_1e_2\cdots e_n$ to denote a path for simplicity.

When given a temporal logic formula ϕ, in order to find those marking sequences in $\mathcal{Q}$ satisfying ϕ, we first define the product PN between the $\mathcal{Q}$ and $\mathcal{A}_\phi$.

Definition 1. Given $\mathcal{Q} = (P, T, Pre, Post, \Pi, h, g)$ with the initial marking M_0 and BA $\mathcal{A} = (Q, q_0, 2^\Pi, \delta, Q_F)$, we define the product of $\mathcal{Q}$ and $\mathcal{A}$ as a new PN $\mathcal{Q}' = (P', T', Pre', Post', M_0', \Pi, h', g')$, where

- $P' = P \cup Q$ is a finite set containing all places;
- $T' \subseteq \{T \times \delta\} \cup T$ is a finite set containing all transitions;
- Pre' (resp.,$Post'$) : $P' \times T' \to \{0, 1\}$ is the pre (resp., post)-incidence functions defined by ${}^\bullet t$ and $t^\bullet$ as follows: $\forall t \in T', \exists o(t) \in T$,
  - $\cdot$ $h(o(t)) = \emptyset$,
    - * ${}^\bullet t \cap P' = {}^\bullet o(t) \cap P,\ t^\bullet \cap P' = o(t)^\bullet \cap P$;
  - $\cdot$ $h(o(t)) \neq \emptyset$,
    - * $|{}^\bullet t \cap Q| = |t^\bullet \cap Q| = 1$;
    - * ${}^\bullet t \cap P' = {}^\bullet o(t) \cap P,\ t^\bullet \cap P' = o(t)^\bullet \cap P$;
    - * $[q_1 = ({}^\bullet t \cap Q), q_2 = (t^\bullet \cap Q)] \Rightarrow [\exists o(t) \in T] \wedge [\delta(q_1, h(o(t))) = q_2]$,

  where $o : T' \to T$ is the projection function from T' to T;
- M_0' is the initial marking defined by:
  - $\cdot$ $\forall p \in P, M_0'(p) = M_0(p)$;
  - $\cdot$ $M_0'(q_0) = 1$;
  - $\cdot$ $\forall q \in Q/q_0, M_0'(q) = 0$;
- Π is a finite set containing all the atomic propositions;
- $h' : T' \to 2^\Pi$ is a function that specifies a subset of atomic propositions $\Pi' \subseteq \Pi$ for every transition $t \in T'$ defined by: $\forall t \in T', h'(t) = h(o(t))$;
- $g' : T' \to \mathbb{N}^+$ is a function specifying the weight of each transition $t \in T'$ defined by: $\forall t \in T', g'(t) = g(o(t))$.

As $\mathcal{Q}$ is bounded and conservative, from definition 1, we know that $\mathcal{Q}'$ is also bounded and conservative. Besides, we know that $\forall M \in R(\mathcal{Q}'), \Sigma_{p' \in P'} M(p') = \Sigma_{p \in P} M_0(p) + 1$. Compared with $\mathcal{Q}$, there is one more token in $\mathcal{Q}'$. This token will be used to track the completion of the temporal logic task.

Next, we define the projection function $\mathcal{P} : \Sigma(\mathcal{Q}') \to \Sigma(\mathcal{Q})$ as follows: $\forall \sigma = t_0t_1t_2\cdots \in \Sigma(\mathcal{Q}'), \mathcal{P}(\sigma) = o(t_0)o(t_1)o(t_2)\cdots \in \Sigma(\mathcal{Q})$.

Remark 2. Actually, given a temporal logic formula ϕ, the sequence satisfying ϕ can be divided into infinite stages according to atomic proposition, and we usually do not pay attention to the transfer behavior on those transitions with empty proposition. In some cases, the atomic propositions in $\mathcal{Q}$ is sparse, which means that $|P_\Pi| < |P|$. Besides, as we only consider $\mathcal{Q}$ as a SM, we can simplify $\mathcal{Q}$ by abstracting the transitions with non-empty propositions to reduce $|P|$, which will also reduce $|P'|$. Besides, as we will see later in section IV-C, the BRG $\mathcal{B}'$ is a compact representation of the reachability graph of $\mathcal{Q}'$, whose scale is polynomial with $|P'|$. And we will use $\mathcal{B}'$ to search optimal sequence. Therefore, reducing $|P'|$ can also reduce the scale of $\mathcal{B}'$, which can reduce the complexity of searching optimal sequence.

As $\mathcal{Q}$ is a SM, it can also be considered as a simple graph with P as the vertex set and T as the edge set, which we make some simplification on to acquire a simpler SM model. Given a sequence in $\mathcal{Q}$ satisfying the ϕ, it can be divided into infinite stages according to atomic propositions. To optimize the sequence cost, given any $p_1, p_2 \in P_\Pi$, we are only interested in the shortest sequence, which do not pass any place in P_Π, starting from p_1 to p_2. Such an issue is captured by the shortest path planning problem in the *single stage planning simple graph* (SSPG) defined as follows. Given $\mathcal{Q} = (P, T, Pre, Post, \Pi, h, g)$ and $r \in P_0 \cup P_\Pi$ being a target vertex, the single stage planning simple graph (SSPG) rooted at r is a six-tuple $\mathcal{T}_r = (P_\mathcal{T}, T_\mathcal{T}, \Pi, h_\mathcal{T}, g_\mathcal{T}, \hat{r})$, where

- $\hat{r}$ is a new copy vertex of $r \in P_0 \cup P_\Pi$;
- $P_\mathcal{T} = P \cup \{\hat{r}\}$ is the vertex set;
- Π is a set containing all the atomic propositions;
- $T_\mathcal{T} \subseteq T \cup (\hat{r} \times P)$ is a set of edges satisfying the following constraints:
  - $\cdot$ $\forall p \in P_\Pi, |\text{Succ}_\mathcal{T}(p)| = 0$;
  - $\cdot$ $\forall p \in P \setminus P_\Pi, |\text{Succ}_\mathcal{T}(p)| = |\text{Succ}_\mathcal{Q}(p)|$;
  - $\cdot$ $|\text{Succ}_\mathcal{T}(\hat{r})| = |\text{Succ}_\mathcal{Q}(r)|$;

  · $\forall \hat{t} = \langle p_1, p_2\rangle \in T_{\mathcal{T}} : t = \langle H(p_1), p_2\rangle \in T$, where $H : P_{\mathcal{T}} \to P$ simply removes "hat" for each vertex, i.e., it maps $\hat{r}$ to r and does nothing to other vertices;
- $h_{\mathcal{T}} : T_{\mathcal{T}} \to 2^{\Pi}$ is a function that specifies a subset of atomic propositions $\Pi' \subseteq \Pi$ for every edge $t \in T_{\mathcal{T}}$ defined by: $\forall \hat{t} \in T_{\mathcal{T}}, h_{\mathcal{T}}(\hat{t}) = h(t)$;
- $g_{\mathcal{T}} : T_{\mathcal{T}} \to \mathbb{N}^{+}$ is a function that specifies a weight for every edge e defined by: $\forall \hat{t} \in T_{\mathcal{T}}, g_{\mathcal{T}}(\hat{t}) = g(t)$.

Note that a SSPG starts from $\hat{r}$ and terminates at vertices P_Π. Therefore, P_Π are also referred to the *termination vertices* of $\mathcal{T}_r$. Intuitively, a SSPG represents all the choices that we can make when staring from r before arriving P_Π and all the vertices in $P \setminus (P_\Pi \cup \{\hat{r}\})$ are places with empty proposition (When r is in P_0, $\hat{r}$ is also with empty proposition).

Given $p \in P_\Pi$, we use $L_{\mathcal{T}}(p) = \{l = t_1 \cdots t_n | {}^\bullet t_1 = \hat{r}, t_n^\bullet = p, t_i^\bullet = {}^\bullet t_{i+1}, t_i \in T_{\mathcal{T}}, \forall 1 \le i \le n-1\}$ to denote the set containing all the paths in $\mathcal{T}_r$ from $\hat{r}$ to p. In order to abstract the transition behaviors from r to p, we first solve the shortest path l_{rp} in $\mathcal{T}_r$ from $\hat{r}$ to place $p \in P_\Pi$, which is defined according to the cumulative weight g_{rp} of edges as follows:

$$g_{rp} = \min_{l = t_1 \cdots t_n \in L_{\mathcal{T}}(p)} \sum_{m=1}^{n} g_{\mathcal{T}}(t_m).$$

It is known that g_{rp} is easy to calculate by the Dijkstra's algorithm, and if $p \in P_\Pi$ is not reachable from r, we define $g_{rp} = \infty$. Therefore, we use l_{rp} to represent the transition relationship from r to p and for all $p' \in P_\Pi$, we can solve $l_{rp'}$ and all the transition behaviors starting from r can be summarized by these paths. Let $\mathfrak{T}$ be the set containing all the SSPGs constructed from $\mathcal{Q}$. Based on $\mathfrak{T}$, we use $\mathfrak{L}$ to denote the set containing all the shortest sequence in $\mathcal{Q}$ from $P_0 \cup P_\Pi$ to P_Π such that

- $\mathfrak{L} = \{t_1 t_2 \cdots t_n \in \Sigma(\mathcal{Q}) \mid \exists \mathcal{T}_{p_0} \in \mathfrak{T},\ l_{p_0 p_n} = \hat{t_1} t_1 \cdots t_n,\ g_{p_0 p_n} \neq \infty\}$.

Remark 3. Although we only care about the transition relationship between places in P_Π, we still need to acquire those transition behaviors from P_0 to P_Π to integrate the initial position information of the robots. Besides, when $r \in P_\Pi$, although we are interested in the transition behaviors from r to P_Π, the graph is actually rooted at its copy $\hat{r}$ because it may go back to r in one stage. Therefore, we use a copy $\hat{r}$ to distinguish the possible vertex from which the next stage starts. For formal unification, we also copy r when $r \in P_0$.

Then, given a SM $\mathcal{Q}$, based on $\mathfrak{L}$, we obtain the *abstracted SM* $\mathfrak{Q}$, which is a simplification model of $\mathcal{Q}$.

Definition 2. Given a SM $\mathcal{Q} = (P, T, Pre, Post, M_0, \Pi, h, g)$, and the set $\mathfrak{L}$, the abstracted general SM is defined as a eight-tuple $\mathfrak{Q} = (P^{\mathfrak{a}}, T^{\mathfrak{a}}, Pre^{\mathfrak{a}}, Post^{\mathfrak{a}}, M_0^{\mathfrak{a}}, \Pi, h^{\mathfrak{a}}, g^{\mathfrak{a}})$, where

- $P^{\mathfrak{a}} = P_0 \cup P_\Pi$ is a finite set containing all places;
- $T^{\mathfrak{a}}$ is a finite set containing all transitions;
- $Pre^{\mathfrak{a}}$ (resp.,$Post^{\mathfrak{a}}$): $P^{\mathfrak{a}} \times T^{\mathfrak{a}} \to \{0,1\}$ is the pre (resp., post)-incidence function defined by ${}^\bullet t$ and $t^\bullet$ as follows: $\forall t \in T^{\mathfrak{a}}$,
  · $|{}^\bullet t| = |t^\bullet| = 1$;
  · $[(p_1 = {}^\bullet t) \wedge (p_2 = t^\bullet)] \Rightarrow [\exists l_{p_1 p_2} \in \mathfrak{L}, o^{\mathfrak{a}}(t) = l_{p_1 p_2}]$;
  where $o^{\mathfrak{a}} : T^{\mathfrak{a}} \to \mathfrak{L}$ is the projection function from $T^{\mathfrak{a}}$ to $\mathfrak{L}$;
- $M_0^{\mathfrak{a}}$ is the initial marking defined as follows:
  · $\forall p \in P_0, M_0^{\mathfrak{a}}(p) = M_0(p)$;
  · $\forall p \in P_\Pi, M_0^{\mathfrak{a}}(p) = 0$;
- Π is a finite set containing all the atomic propositions;
- $h^{\mathfrak{a}} : T^{\mathfrak{a}} \to 2^{\Pi}$ is a function that specifies a subset of atomic propositions $\Pi' \subseteq \Pi$ for every transition $t \in T^{\mathfrak{a}}$ defined by: $\forall t \in T^{\mathfrak{a}}, [h^{\mathfrak{a}}(t) = h_{\mathcal{T}}(\sigma_{|\sigma|})] \wedge [\sigma = o^{\mathfrak{a}}(t)]$, where $\sigma_{[i]}$ denotes the ith transition in σ.
- $g^{\mathfrak{a}} : T^{\mathfrak{a}} \to \mathbb{N}^{+}$ is a function that specifies a weight for every transition $t \in T^{\mathfrak{a}}$ defined by: $\forall t \in T^{\mathfrak{a}}, [g^{\mathfrak{a}}(t) = g_{p_1 p_2}] \wedge [p_1 = {}^\bullet t] \wedge [p_2 = t^\bullet]$.

Then, we define the projection function $\mathcal{P}' : \Sigma(\mathfrak{Q}) \to \Sigma(\mathcal{Q})$ as follows: $\forall \sigma = t_0 t_1 t_2 \cdots \in \Sigma(\mathfrak{Q})$, $\mathcal{P}'(\sigma) = o^{\mathfrak{a}}(t_0) o^{\mathfrak{a}}(t_1) o^{\mathfrak{a}}(t_2) \cdots \in \Sigma_{\mathcal{Q}}$.

From the definition above, we can see that $\nexists t \in T^{\mathfrak{a}}, h^{\mathfrak{a}}(t) = \emptyset$ and $\mathfrak{Q}$ is also a SM. However, compared to $\mathcal{Q}$, $\mathfrak{Q}$ only needs $|P_0| + |P_\Pi|$ places to model the same system, which is apparent from def 2, and in general case, we always have $|P_0| + |P_\Pi| < |P|$. Therefore, we can use $\mathfrak{Q}$ to construct the product PN $\mathfrak{Q}'$ with $\mathcal{A}$ by def 1, which allows $\mathfrak{Q}'$ to have fewer places compared with $\mathcal{Q}'$, thus leading to a smaller reachability graph.

3.2 Prefix-suffix form solution in $\mathcal{Q}'$

When given a product PN $\mathcal{Q}'$, for a prefix-suffix sequence $\sigma = \sigma_{pre}(\sigma_{sur})^{\omega} \in \Sigma(\mathcal{Q}')$, we define the *average cost per cycle* of σ as follows:

$$\mathsf{Cost}_{\mathcal{Q}'}^{AveT}(\sigma) = \frac{\boldsymbol{g}^T \cdot \boldsymbol{y}_{\sigma_{sur}}}{N'(\sigma_{sur})} \tag{4}$$

where $N'(\sigma_{sur}) = \sum_{t \in \mathbb{T}'_{sur}} \boldsymbol{y}_{\sigma_{sur}}(t)$ represents the number of visiting π_{sur} along σ_{sur}, where $\mathbb{T}'_{sur}$ is the π_{sur} *input transition set in* $\mathcal{Q}'$ defined as follows: $\mathbb{T}'_{sur} = \{t \in T' | \pi_{sur} \in h'(t)\}$.

Besides, we define $\mathbb{T}_F$ as the Q_F *input transition set*: $\mathbb{T}_F = \{t \in T' | t^\bullet \cap Q_F \neq \emptyset\}$. We call σ an *accepting sequence* if $\sum_{t \in \mathbb{T}_F} \boldsymbol{y}_{\sigma_{sur}}(t) \neq 0$, and use $\Sigma_\phi(\mathcal{Q}')$ to denote the set containing all the feasible accepting sequences in $\mathcal{Q}'$.

The following lemma relates the prefix-suffix accepting transition sequences in $\mathcal{Q}$ and $\mathcal{Q}'$.

Lemma 1. Given a BA $\mathcal{A}_\phi$, a PN $\mathcal{Q}$ and their product PN $\mathcal{Q}'$, we have

- $\forall \sigma = \sigma_{pre}(\sigma_{sur})^{\omega} \in \Sigma_\phi(\mathcal{Q}'), [\mathcal{P}(\sigma) = \sigma'_{pre}(\sigma'_{sur})^{\omega} \in \Sigma_\phi(\mathcal{Q})] \wedge [\mathsf{Cost}_{\mathcal{Q}'}^{AveT}(\sigma) = \mathsf{Cost}_{\mathcal{Q}}^{AveT}(\mathcal{P}(\sigma))]$;
- $\forall \sigma = \sigma_{pre}(\sigma_{sur})^{\omega} \in \Sigma_\phi(\mathcal{Q}), \exists \sigma' = \sigma'_{pre}(\sigma'_{sur})^{\omega} \in \Sigma_\phi(\mathcal{Q}'), [\mathcal{P}(\sigma') = \sigma] \wedge [\mathsf{Cost}_{\mathcal{Q}}^{AveT}(\sigma) = \mathsf{Cost}_{\mathcal{Q}'}^{AveT}(\sigma')]$

The following lemma shows that a minimum average cost per cycle sequence $\sigma^\star_{\mathcal{Q}'}$ can be found by searching over finite-memory sequences of the prefix-suffix form in $\mathcal{Q}'$.

Lemma 2. There exists at least one accepting feasible transition sequence $\sigma^\star_{\mathcal{Q}'}$ in $\mathcal{Q}'$ satisfying

- $\forall \sigma \in \Sigma_\phi(\mathcal{Q}'), \mathsf{Cost}_{\mathcal{Q}'}^{AveT}(\sigma^\star_{\mathcal{Q}'}) \le \mathsf{Cost}_{\mathcal{Q}'}^{AveT}(\sigma)$;
- $\sigma^\star_{\mathcal{Q}'}$ is in prefix-suffix form, i.e., $\sigma^\star_{\mathcal{Q}'} = \sigma^\star_{pre'}(\sigma^\star_{suf'})^{\omega}$.

Besides, it is trivial that, when $\mathcal{Q}$ is a SM, and we use $\mathfrak{Q}$ to construct the product PN $\mathfrak{Q}'$, the above three lemmas are also valid. This leads to the following lemma 3.

Lemma 3. When $\mathcal{Q}$ is a SM, there exists at least one accepting feasible transition sequence $\sigma^\star_{\mathfrak{Q}'}$ of $\mathfrak{Q}'$ satisfying

- $\forall \sigma \in \Sigma_\phi(\mathfrak{Q}'), \mathsf{Cost}_{\mathfrak{Q}'}^{AveT}(\sigma^\star_{\mathfrak{Q}'}) \le \mathsf{Cost}_{\mathfrak{Q}'}^{AveT}(\sigma)$;

- $\sigma^\star_{\mathfrak{Q}'}$ is in prefix-suffix form, i.e., $\sigma^\star_{\mathfrak{Q}'} = \sigma^\star_{pre^\star}(\sigma^\star_{suf^\star})^\omega$;
- $\mathsf{Cost}^{AveT}_{\mathfrak{Q}'}(\sigma^\star_{\mathfrak{Q}'}) = \mathsf{Cost}^{AveT}_{\mathcal{Q}'}(\sigma^\star_{\mathcal{Q}'})$.

Lemma 3 shows us the equivalence of the results when we use $\mathcal{Q}$ or $\mathfrak{Q}$ to solve problem 1 when $\mathcal{Q}$ is a SM. Therefore, from now on, we will not distinguish $\mathcal{Q}$ and $\mathfrak{Q}$ unless otherwise specified and we will use $\mathcal{Q}$ to represent the two models for the convenience of narration. This leads to the formulation of the problem MOTP-AT in $\mathcal{Q}'$.

Problem 2. (MOTP-AT-$\mathcal{Q}'$) Given a BA $\mathcal{A}_\phi$, a PN $\mathcal{Q}$ and their product PN $\mathcal{Q}'$, find a optimal sequence $\sigma^\star_{\mathcal{Q}'}$ for $\mathcal{Q}'$ such that

- $\sigma^\star_{\mathcal{Q}'} \in \Sigma_\phi(\mathcal{Q}')$;
- $\sigma^\star_{\mathcal{Q}'}$ is in prefix-suffix form;
- $\forall \sigma \in \Sigma_\phi(\mathcal{Q}') : \mathsf{Cost}^{AveT}_{\mathcal{Q}'}(\sigma^\star_{\mathcal{Q}'}) \leq \mathsf{Cost}^{AveT}_{\mathcal{Q}'}(\sigma)$.

Proposition 1. Let $\sigma^\star_{\mathcal{Q}'}$ be a solution to problem 2, then $Pro(\sigma^\star_{\mathcal{Q}'})$ is a solution to problem 1.

3.3 Prefix-suffix form solution in BRG representation of $\mathcal{Q}'$

In (Ma et al. (2016)), a compact way, called *basis reachability graph* (BRG), to represent the reachability graph of a PN is proposed to solve the finite sequence reachability problem. However, in this paper, we use BRG to reduce planning complexity in the infinite sequence cyclic task planning problem. Before we give the definition of BRG, we recall some basic concepts from (Ma et al. (2016)).

Given a PN $\mathcal{Q} = (P, T, Pre, Post, \Pi, h, g)$, the pair $\varpi = (T_E, T_I)$ is called a *basis partition* of T if

- $T_I \subseteq T, T_E = T \setminus T_I$;
- the T_I-induced subnet is acyclic

where the sets T_E and T_I are called the *explicit transition set* and the *implicit transition set*, respectively.

Definition 3. (Ma et al. (2016)) Given a bounded PN $\mathcal{Q}$ and a basis partition $\varpi = (T_E, T_I)$, its basis reachability graph (BRG) is a four-tuple $\mathcal{B} = (\mathcal{M}, Tr, \Delta, M_0)$, where

- $\mathcal{M}$ is the set of basis markings;
- Tr is the set of pairs $(t, \boldsymbol{y}) \in T_E \times \mathbb{N}^{n_I}$;
- $\Delta : \mathcal{M} \times Tr \to \mathcal{M}$ is a transition relation such that $\Delta[(M_1, (t, \boldsymbol{y})] = M_2$ if (i) $t \in T_E$; (ii) $\boldsymbol{y} \in Y_{min}(M_1, t)$; and (iii) $M_2 = M_1 + C_I \cdot \boldsymbol{y} + C(\cdot, t)$.
- M_0 is the initial marking.

From definition 3, we know that $\mathcal{B}$ is a finite directed multi-graph with $\mathcal{M}$ being the vertex set and Δ being the edge set. Given a basis partition $\varpi = (T_E, T_I)$ and a basis marking $M_b \in \mathcal{M}$, we define $R_I(M_b)$ as the implicit reach of M_b by $R_I(M_b) = \{M \in \mathbb{N}^m | \exists \sigma \in T_I^*, M = M_b + C \cdot \boldsymbol{y}_\sigma\}$.

Proposition 2. (Ma et al. (2016)) Given a PN $\mathcal{Q}$, a basis partition $\varpi = (T_E, T_I)$ and a marking $M \in R(\mathcal{Q})$, let $\mathcal{B} = (\mathcal{M}, Tr, \Delta, M_0)$ be the corresponding BRG. Then the following two statements are equivalent:

- $(\exists \sigma = \sigma_1 t_1 \cdots \sigma_n t_n \sigma_{n+1} \in \Sigma(\mathcal{Q}))$ $M_0[\sigma\rangle M$, where $\sigma_j \in T_I^*, t_j \in T_E, \forall 1 \leq j \leq n+1$;
- $(\exists l \in L_\mathcal{B}, M_{b,n} \in \mathcal{M})$ $M \in R_I(M_{b,n})$, where
$$l = M_0 \xrightarrow{t_1, \boldsymbol{y}_1} M_{b,1} \xrightarrow{t_2, \boldsymbol{y}_2} \cdots \xrightarrow{t_n, \boldsymbol{y}_n} M_{b,n}$$
and $\boldsymbol{y}_i \in \mathbb{N}^{T_I}, \forall 1 \leq i \leq n$.

Given product PN $\mathcal{Q}'$ and a basis partition $\varpi = (T_E, T_I)$, we construct the BRG $\mathcal{B}' = (\mathcal{M}', Tr', \Delta', M_0')$. As $\mathcal{B}'$ is also a graph, we use $cyc_{\mathcal{B}'}$ and $Scyc_{\mathcal{B}'}$ to denote the set containing all cycles and simple cycles in $\mathcal{B}'$ respectively.

Let $\sigma^\star_{\mathcal{Q}'} = \sigma^\star_{pre'}(\sigma^\star_{sur'})^\omega$ be the prefix-suffix form solution that solves problem 2. As $\mathcal{Q}'$ is bounded and conservative, if $M_0'[\sigma^\star_{pre'}\rangle M'_{n+1}$, we have $M_0'[\sigma^\star_{pre'}(\sigma^\star_{sur'})^\omega\rangle M'_{n+1}$. From proposition 2, we know that there is a path l' in $\mathcal{B}'$ in the following form:

$$M_0' \xrightarrow{t_1, \boldsymbol{y}_1} M_{b,1} \xrightarrow{t_2, \boldsymbol{y}_2} \cdots \xrightarrow{t_\omega, \boldsymbol{y}_\omega} M_{b,\omega}$$

such that $M'_{n+1} \in R_I(M_{b,\omega})$. Let $M_{b,\omega}[\sigma'_\omega\rangle M'_{n+1}$, where $\sigma'_\omega \in T_I^*$, then we have the following lemma for σ'_ω:

Lemma 4. Given σ'_ω, we have $\mathbf{1}^T \cdot \boldsymbol{y}_{\sigma'_\omega} < \infty$, where $\boldsymbol{y}_{\sigma'_\omega} \in \mathbb{N}^{T_I}$ and $\mathbf{1} \in \mathbb{N}^{T_I}$ is a full 1 vector.

As $\mathcal{M}'$ is a finite set and l' is a infinite sequence which contains infinite basis markings, at least one basis marking appears infinitely in l'. Let $\Xi(l')$ be the set containing all the basis markings which appears infinitely in l'. For any $M_i \in \Xi(l')$, we can divide l' into infinite continuous cycles which start and end in M_i. Certainly, there might be a finite transition sequence σ_{0-i} from M_0 to M_i and a finite sequence $\sigma_{i-\omega}$ from M_i to $M_{b,\omega}$.

Given a cycle $l \in cyc_{\mathcal{B}'}$, let l be the following form:

$$M_0 \xrightarrow{t_1, \boldsymbol{y}_1} M_1 \xrightarrow{t_2, \boldsymbol{y}_2} \cdots \xrightarrow{t_n, \boldsymbol{y}_n} M_n,$$

where $M_0 = M_n$. We use σ_l to denote the transition sequence including both explicit transitions and implicit transitions such as: $M_0[\sigma_l\rangle M_n$ and $l_{[i]}$ to denote the ith basis marking. Besides, we use $\mathbb{T}_l$ to denote the set containing all the transitions appearing in the cycle which is defined as follows: $\mathbb{T}_l = \{t \in T' | t \in \mathbb{T}_1 \cup \mathbb{T}_2\}$, where $\mathbb{T}_1 = \{t \in T_E | t \in \bigcup_{j=1}^n \{t_j\}\}$ and $\mathbb{T}_2 = \{t \in T_I | \sum_{i=1}^n \boldsymbol{y}_i(t) \neq 0\}$.

Then, for $Scyc_{\mathcal{B}'}$, we have the following lemma:

Lemma 5. $\exists l \in Scyc_{\mathcal{B}'}, \mathbb{T}_l \cap \mathbb{T}_F \neq \emptyset$.

Based on lemma 5, we denote $Scyc^F_{\mathcal{B}'}$ as the *Q_F-simple cycle set in* $\mathcal{B}'$, which is defined as follows: $Scyc^F_{\mathcal{B}'} = \{l \in Scyc'_\mathcal{B} | \mathbb{T}_l \cap \mathbb{T}_F \neq \emptyset\}$. Then, for any $l \in Scyc^F_{\mathcal{B}'}$, we define its *average cost per cycle* of l in $\mathcal{B}'$ as follows:

$$\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_l) = \frac{\boldsymbol{g}^T \cdot \boldsymbol{y}_{\sigma_l}}{N'(\sigma_l)}. \quad (5)$$

From lemma 5, we know that if there exists a solution for problem 2, there exists at least one simple cycle l in $Scyc^F_{\mathcal{B}'}$ such that $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_l) \neq \infty$.

For the simplification of narration, we make the following mild assumption, whose absence will not influence the conclusion: $\forall l_1, l_2 \in Scyc^F_{\mathcal{B}'}, [\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l_1}) \neq \infty, \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l_2}) \neq \infty] \Rightarrow [\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l_1}) \neq \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l_2})]$.

Based on the above assumption, we know that there exists a simple cycle $l^\star \in Scyc^F_{\mathcal{B}'}$ such that:

- $\forall l \in Scyc^F_{\mathcal{B}'}, \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) \leq \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_l)$.

Besides, it follows from (5) that in the worst case, $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star})$ can be computed in $O((|\mathcal{M}'| + \mathfrak{N}_{\mathcal{B}'})^3 \cdot 2^{|\mathcal{M}'| + \mathfrak{N}_{\mathcal{B}'}})$ operations at most, where $\mathfrak{N}_{\mathcal{B}'}$ is the number of multi edges of $\mathcal{B}'$. However, in most cases, the actual complexity is much lower than the upper bound.

Finally, we can get the following proposition:

Proposition 3. There is a solution $\sigma^\star_{\mathcal{Q}'}$ to problem 2, if and only if $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) \neq \infty$ and

- $\sigma^\star_{\mathcal{Q}'} = \sigma^\star_{pre'}(\sigma_{l^\star})^\omega$;
- $\mathsf{Cost}^{AveT}_{\mathcal{Q}'}(\sigma^\star_{\mathcal{Q}'}) = \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star})$.

where σ'_{pre} is any sequence from M'_0 to $l^\star$.

3.4 Computation of Solution to Problem 1

In this part, we propose the overall algorithm for problem 1 as shown in algorithm 1.

Algorithm 1 Algorithm for MOTP-AT

```
Input: The weighted SM $\mathcal{Q}$ and a temporal logic formula $\phi$ in form (2)
Output: The prefix-suffix form solution $\sigma^\star$ to problem 1
 1: Construct the Büchi automata $\mathcal{A}_\phi$;
 2: if $|P_0| + |P_\Pi| < |P|$ then
 3:     Construct $\mathfrak{Q}$ based on $\mathcal{Q}$;
 4:     Construct the Product PN $\mathcal{Q}'$ based on $\mathfrak{Q}$ and $\mathcal{A}_\phi$;
 5: else
 6:     Construct the Product PN $\mathcal{Q}'$ based on $\mathcal{Q}$ and $\mathcal{A}_\phi$;
 7: Construct BRG $\mathcal{B}'$ based on $\mathcal{Q}'$;
 8: Let $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) = \infty$ and $l^\star = l$, where $l \in Scyc^F_{\mathcal{B}'}$;
 9: for $l_k \in Scyc^F_{\mathcal{B}'}, k = 1, 2, \cdots, |Scyc^F_{\mathcal{B}'}|$ do
10:     if $[Cost^{AveT}_{\mathcal{B}'}(\sigma_{l_k}) < Cost^{AveT}_{\mathcal{B}'}(\sigma_{l^\star})]$ then
11:         $l^\star = l_k$;
12:         $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) = \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l_k})$;
13:     else $l^\star = l^\star$;
          $\mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) = \mathsf{Cost}^{AveT}_{\mathcal{B}'}(\sigma_{l^\star})$;
14: $\sigma^\star_{sur'} = \sigma_{l^\star}$;
15: if $Cost^{AveT}_{\mathcal{B}'}(\sigma_{l^\star}) = \infty$ then
16:     Go to line 24;
17: else
18:     Let $\sigma^\star_{pre'}$ be a feasible sequence from $M'_0$ to $\sigma_{l^\star}$;
19:     if $\mathcal{Q}'$ is constructed based on $\mathcal{Q}$ and $\mathcal{A}$ then
20:         $\sigma^\star_{pre} = \mathcal{P}(\sigma^\star_{pre'}), \sigma^\star_{sur} = \mathcal{P}(\sigma^\star_{sur'})$;
21:     else
22:         $\sigma^\star_{pre} = \mathcal{P}'(\mathcal{P}(\sigma^\star_{pre'})), \sigma^\star_{sur} = \mathcal{P}'(\mathcal{P}(\sigma^\star_{sur'}))$;
23:     Go to line 25;
24: return There is no solution to problem 1;
25: return The solution to problem 1 is $\sigma^\star = \sigma^\star_{pre}(\sigma^\star_{sur})^\omega$;
```

This algorithm works as follows. Firstly, it constructs the Büchi automata $\mathcal{A}_\phi$ and judges whether the original PN $\mathcal{Q}$ can be simplified. If it can be simplified, it constructs the product PN $\mathcal{Q}'$ based on the simplified PN $\mathfrak{Q}$. Otherwise, it constructs $\mathcal{Q}'$ based on $\mathcal{Q}$. Then based on the BRG $\mathcal{B}'$ constructed from the product PN $\mathcal{Q}'$ (line 10), it computes the smallest average cost per cycle simple cycle $l^\star$ which visits Q_F to be the candidate suffix part of the solution to problem 2. Finally, it verifies whether there is a solution to problem 1 and computes it by projecting the prefix and suffix part we compute for problem 2 back to the original PN $\mathcal{Q}$ if the solution indeed exists.

4. NUMERICAL EXAMPLES

The algorithmic framework developed in this paper is implemented in MATLAB and Python 3.7, and here we describe a motion planning case for a team of identical robots to illustrate the advantages of our algorithm.

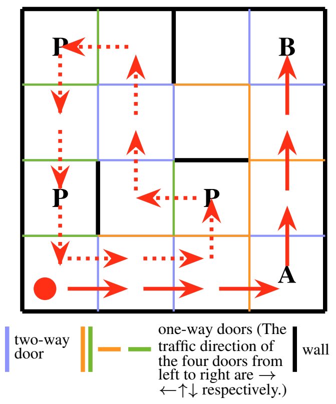
P
B
P
P
A
two-way door
one-way doors (The traffic direction of the four doors from left to right are → ←↑↓ respectively.)
wall

Fig. 1. PN model $\mathcal{Q}$ for the workspace.

We consider a 4×4 space as shown in Fig 1. We use red dot to denote the starting region of the robots. We assume that the robot can move from one region to another region by passing through a door. There are two kinds of door in the space: the two-way door and the one-way door which are marked at the bottom of Fig 1. Besides, there may be a wall between different regions and around the space which cannot be passed through. Suppose the cost of passing any door is identical and let it be 1. Consider the atomic proposition set $\Pi = \{A, B, D, P\}$ and some regions are marked with atomic proposition, while the other regions are with empty proposition. We consider P as the atomic proposition that we should visit infinitely and give the following temporal logic specification:

$$\phi_1 = (\neg B \mathsf{U} A) \wedge \Diamond B \wedge \Box\Diamond P.$$

which requests that we should visit P infinite often and visit region A and B in order. The temporal logic specification corresponds to a BA with 4 states, 1 accepting state and 9 transitions. The robots start from the region in the lower left corner. Therefore, p_1 is the starting place filled with red.

Let $n = 2$, which means two robots participate in the planning. As $\mathcal{Q}$ is a SM and $|P_0| + |P_\Pi| < |P|$, we construct the BRG $\mathcal{B}^{\mathfrak{a}} = (\mathcal{M}^{\mathfrak{a}}, Tr^{\mathfrak{a}}, \Delta^{\mathfrak{a}}, M^{\mathfrak{a}}_0)$ from the abstract PN $\mathfrak{Q}$ with $|\mathcal{M}^{\mathfrak{a}}| = 24$, $|\Delta^{\mathfrak{a}}| = 235$ and $\mathfrak{N}_{\mathcal{B}^{\mathfrak{a}}} = 46$ only, where $\mathfrak{N}_{\mathcal{B}^{\mathfrak{a}}}$ is the number of multi edges of $\mathcal{B}^{\mathfrak{a}}$. Then, we get one of the feasible optimal transition sequence for $\mathcal{Q}$ starting from M_0 as follows:

- $\sigma^\star = t_1t_3t_5t_9t_{16}t_{26}(t_1t_3t_8t_{11}t_{14}t_{23}t_{28}t_{22}t_{13}t_6)^\omega$

with $\mathsf{Cost}^{AveT}_{\mathcal{Q}}(\sigma^\star) = \frac{10}{3}$. This sequence corresponds to two paths as shown in Fig 1 with solid line and dotted line for the two robots respectively.

We can also solve this problem by the baseline method of product automaton Wolff et al. (2012). We also use two robots and construct the product automaton $\mathcal{P}$ with $|S| = 134$ states and $|E| = 400$ edges. In $\mathcal{P}$, we need to search all cycles starting and ending at the accepting vertices to get the cycle with minimum average cost, which can be done in $O(|S|^3 \cdot 2^{|S|})$ operations. However, the search process in $\mathcal{B}^{\mathfrak{a}}$ can be finished in $O((|\mathcal{M}^{\mathfrak{a}}| + \mathfrak{N}_{\mathcal{B}^{\mathfrak{a}}})^3 \cdot 2^{|\mathcal{M}^{\mathfrak{a}}|+\mathfrak{N}_{\mathcal{B}^{\mathfrak{a}}}})$ operations at most.

Finally, as we can see, in this example, two robots are already enough to complete the specification and adding more robots can not reduce the cost further. However, to illustrate the adaptability of our method to the number of robots, we gradually

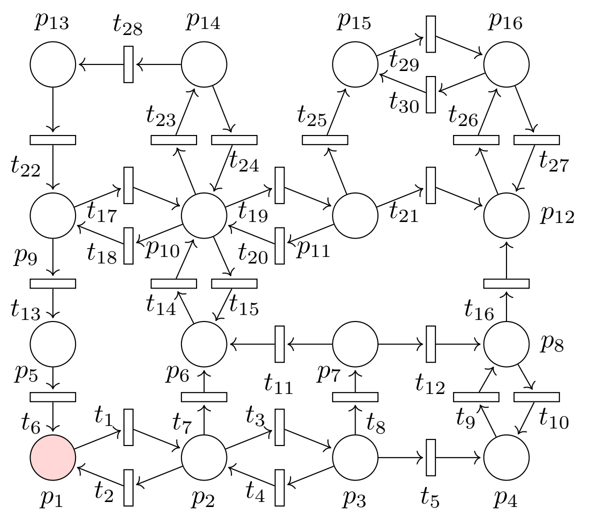

Fig. 2. PN model $\mathcal{Q}$ for the workspace.

Table 1. Numerical Results

N	$\mathcal{P}$			$\mathcal{B}^{\mathfrak{a}}$			
	$\lvert S\rvert$	$\lvert E\rvert$	T_1(sec)	$\lvert\mathcal{M}^{\mathfrak{a}}\rvert$	$\lvert\Delta^{\mathfrak{a}}\rvert$	$\mathfrak{N}_{\mathcal{B}^{\mathfrak{a}}}$	T_2(sec)
2	182	533	0.095	24	235	46	0.0011
3	1574	8475	2.85	71	1305	243	0.28
4	13202	133217	1752.7	172	4579	780	7.31
5	108590	2076603	-	363	12608	1955	105.3
6	881762	32179433	-	693	29671	4211	1052.2
7	-	-	-	1226	62435	8170	7740.9

increase the number of robots from two to seven and use the above two methods to find the solution. The numerical results are shown in Tab 1. N is the number of robots. We compare the number of vertices and edges in $\mathcal{P}$ with $\mathcal{B}^{\mathfrak{a}}$. It is easy to see that with N increasing, the BRG $\mathfrak{B}^{\mathfrak{a}}$ is significantly smaller than the product automaton $\mathcal{P}$ no matter on the number of vertices or edges, and the size of these graphs is also a direct indicator of the memory requirement of the two algorithms. Furthermore, the growth rate of the number of vertices and edges of $\mathfrak{B}^{\mathfrak{a}}$ is also significantly lower than that of $\mathcal{P}$. Besides, we also list down the computation times T_1 and T_2 of solution by these two methods. It can be seen that there is no significant difference in the computation time of solution between these two methods when the number of robots is less than 4. However, when N is larger than 3, these two methods are not comparable in terms of computational efficiency, as we observe significant speed up that our method achieves over the baseline method. We use "–" to denote the case, where we fail to compute due to insufficient RAM (32GB) or very high computation time (> 10000 sec).

5. CONCLUSION

In this paper, we proposed a new approach for optimal multi-robot path planning with cyclic tasks. Our approach is based on the basis reachability graph of Petri nets without enumerating the entire concurrent state-space of the team of robots. We demonstrated the efficiency of the proposed approach by comparing with the standard product-automata-based approach. We showed that our method has better scalability on the number of robots. In our future work, we plan to extend our algorithm to deal with more general fragment of LTL specifications. Also, we plan to consider the concurrent execution of transitions in PN to deal with synchronous operations between robots.

REFERENCES

Baier, C. and Katoen, J. (2008). *Principles of Model Checking*.

Guo, M. and Dimarogonas, D.V. (2015). Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2), 218–235.

He, Z., Zhang, R., Ran, N., and Gu, C. (2022). Path planning of multi-type robot systems with time windows based on timed colored Petri nets. *Applied Sciences*, 12(14), 6878.

Kantaros, Y. and Zavlanos, M.M. (2018). Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. *IEEE Trans. Aut. Control*, 64(5), 1916–1931.

Kloetzer, M. and Mahulea, C. (2020). Path planning for robotic teams based on ltl specifications and petri net models. *Discrete Event Dynamic Systems*, 30(1), 55–79.

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

Liu, S., Trivedi, A., Yin, X., and Zamani, M. (2022). Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control*, 53, 30–50.

Lv, P., Yin, X., Ji, Y., and Li, S. (2021). A game-theoretical approach for optimal supervisory control of discrete event systems for cyclic tasks. In *60th IEEE CDC*, 324–330.

Ma, Z., Tong, Y., Li, Z., and Giua, A. (2016). Basis marking representation of petri net reachability spaces and its application to the reachability problem. *IEEE Trans. Automatic Control*, 62(3), 1078–1093.

Ma, Z., Yin, X., and Li, Z. (2021). Marking diagnosability verification in labeled petri nets. *Automatica*, 131, 109713.

Mahulea, C. and Kloetzer, M. (2018). Robot planning based on boolean specifications using petri net models. *IEEE Trans. Automatic Control*, 63(7), 2218–2225.

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

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

Shi, W., He, Z., Tang, W., Liu, W., and Ma, Z. (2022). Path planning of multi-robot systems with boolean specifications based on simulated annealing. *IEEE Robotics and Automation Letters*, 7(3), 6091–6098.

Smith, S.L., Tmová, J., Belta, C., and Rus, D. (2011). Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 30(14), 1695–1708.

Ulusoy, A., Smith, S.L., and Belta, C. (2014). Optimal multi-robot path planning with ltl constraints: guaranteeing correctness through synchronization. In *Distributed Autonomous Robotic Systems*, 337–351. Springer.

Wolff, E.M., Topcu, U., and Murray, R.M. (2012). Optimal control with weighted average costs and temporal logic specifications. *Proc. of Robotics: Science and Systems*, (2012).

Yang, S., Yin, X., Li, S., and Zamani, M. (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. In *59th IEEE CDC*, 4460–4466.

Yu, P. and Dimarogonas, D.V. (2021). Distributed motion coordination for multirobot systems under ltl specifications. *IEEE Trans. Robotics*.

Yu, X., Yin, X., Li, S., and Li, Z. (2022). Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, 123, 105130.