Control Engineering Practice 138 (2023) 105600

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Optimal multi-robot path planning for cyclic tasks using Petri nets[☆]

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

[a] Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China
[b] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China

ARTICLE INFO

Keywords:
Petri nets
Linear temporal logic
Multi-robot systems
Task planning

ABSTRACT

In this paper, we investigate the problem of optimal multi-robot path planning for cyclic tasks 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, which is a compact representation of the reachability space of the PN. We demonstrate the computational efficiency and scalability of our method through illustrative examples. The proposed methods have also been implemented in real-world experiments.

1. Introduction

1.1. Motivation

Multi-robot systems are widely used in many applications such that autonomous warehouses (Basile, Chiacchio, & Di Marino, 2019; Cai, 2020; Fanti, Mangini, Pedroncelli, & Ukovich, 2018; Tatsumoto, Shiraishi, Cai, & Lin, 2018), manufacturing systems (Kovalenko, Tilbury, & Barton, 2019), environment surveillance (Mansouri, Kanellakis, Fresk, Kominiak, & Nikolakopoulos, 2018; Shin, Kwak, & Lee, 2020) and information gathering (Guo & Zavlanos, 2018a). 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 physical dynamics of robots. In the recent years, considerable attention has been paid on multi-robot planning for *high-level complex tasks* (Fainekos, Girard, Kress-Gazit, & Pappas, 2009; Kantaros & Zavlanos, 2016; Kress-Gazit, Fainekos, & Pappas, 2009; Yang, Yin, Li, & Zamani, 2020; Yu, Yin, Li, & Li, 2022).

In order to describe the high-level task requirement of multi-robot, linear temporal logic (LTL) is widely used as the formal specification language. It can describe high-level requirements such that "*visit a target region infinitely often while avoid reaching some unsafe regions*". Although LTL can provide formal guarantee for the behavior of the robot, solving the planning problem is usually costly especially for multi-robot systems due to the curse of dimensionality. This is because the overall state-space grows exponentially fast when the number of robots increases. How to design optimal plans satisfying LTL tasks for multi-robot systems is a practical but very challenging problem.

1.2. Related works

In the past years, considerable attentions have been paid on robot planning for LTL tasks. For example, Guo and Dimarogonas (2015), Smith, Tmová, Belta, and Rus (2011), Tian, Fang, Yang, and Wei (2021), Ulusoy, Smith, and Belta (2014) consider the problem of finding an optimal infinite path in the prefix-suffix form that satisfies an LTL formula. In Ding, Smith, Belta, and Rus (2014), Guo and Zavlanos (2018b), Lahijanian, Andersson, and Belta (2011), the authors consider the planning problem for probabilistic satisfaction of temporal logic requirements under uncertainties. However, these works mainly focus on single-robot systems and the algorithms usually do not scale for multi-robot systems.

For multi-robot LTL planning problems, 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, Bürger, & Dimarogonas, 2018; Yu & Dimarogonas,

☆ 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).

* Corresponding author.
E-mail addresses: lv-peng@sjtu.edu.cn (P. Lv), luogq11@sjtu.edu.cn (G. Luo), maziyue@xidian.edu.cn (Z. Ma), syli@sjtu.edu.cn (S. Li), yinxiang@sjtu.edu.cn (X. Yin).

https://doi.org/10.1016/j.conengprac.2023.105600
Received 26 November 2022; Received in revised form 6 June 2023; Accepted 8 June 2023
Available online xxxx

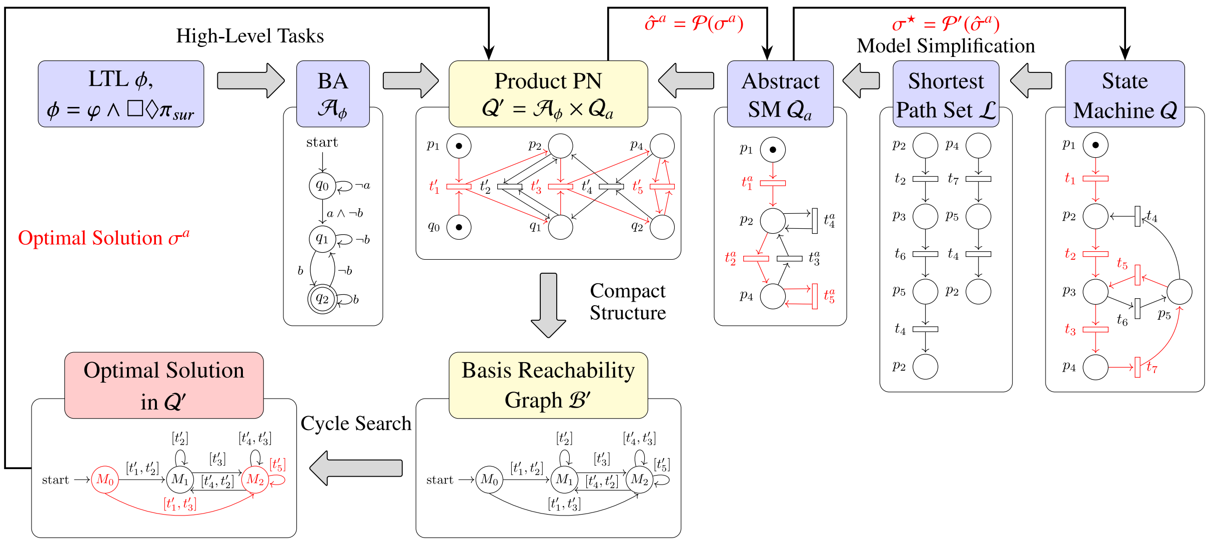


Fig. 1. The overall framework of our synthesis procedure.

2021). However, for these approaches, optimality of the plan is not always guaranteed.

The aforementioned works are mainly based on the automata model of the robots. In contrast, Petri nets (PNs) are a more efficient model for representing concurrent systems (Seatzu, Silva, & Van Schuppen, 2013), and are particularly suitable for modeling multi-robot systems (He, Dong, Ren, Gu, & Li, 2020; He, Zhang, Ran, & Gu, 2022; Lacerda & Lima, 2019; Mahulea & Kloetzer, 2017; Shi, He, Tang, Liu, & Ma, 2022). In the context of LTL planning, PN-based approaches have been investigated in the literature recently, which provide a promising way to mitigate the computation complexity (Mahulea, Kloetzer, & González, 2020). For example, in Mahulea and Kloetzer (2018), the authors investigate multi-robot path planning for Boolean tasks 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 still not guaranteed.

Finally, in the context of Petri nets, several approaches have been developed for efficient computation of optimal sequences (Lefebvre, 2018; Lefebvre & Leclercq, 2015; Ma, Zou, Zhang, & Li, 2022). Particularly, based on the structure of *basis reachability graph* (BRG) (Ma, Tong, Li, & Giua, 2016; Ma, Yin, & Li, 2021; Tong, Li, Seatzu, & Giua, 2017), the authors in Ma et al. (2022) provide an efficient approach for computing optimal *finite* sequences for the purpose of reachability without enumerating the entire state space. However, these approach cannot handle the LTL requirement since LTL requires to design an infinite sequence under different optimality criteria.

1.3. Our contributions

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 robots need 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, Yin, Ji, & Li, 2021). Our objective is to find a plan such that (i) the LTL task is satisfied; and (ii) the cost for each task cycle is minimized.

Compared with existing works, the contributions of this paper are summarized as follows:

- We provide a new PN-based framework for computing optimal plans for multi-robot systems under LTL specifications. The overall framework of our synthesis procedure is shown in Fig. 1. Our approach is based on the construction of the BRG of the product of the abstracted system and the Büchi automaton representing the LTL specification. Previous frameworks for optimal planning of multi-robot systems under LTL tasks are generally not feasible for the average cost per task optimality criterion.
- We provide an explicit algorithm for computing optimal cyclic sequence for the purpose of infinite surveillance based on the BRG. Specifically, we show that the synthesized plan based on the compact structure can be effectively mapped back to the original system while preserving the optimality guarantee. This result is different from the existing works such as (Ma et al., 2022), where only finite sequence is considered for the purpose of reachability.
- We demonstrate the scalability of the proposed approach when the number of robots increases. Specifically, a set of simulation and hardware experiments are performed. Notably, 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.

1.4. Organization

The rest of this paper is organized as follows. In Section 2, we introduce some basic notions and problem formulation. In Section 3, we make a model reduction for planning. Our main synthesis procedure is provided in Section 4. Next, we present computational experiments and hardware demonstration in Section 5. Finally, we conclude the paper in Section 6. Preliminary and partial versions of some results in this paper were presented in Lv, Luo, Yin, Ma, and Li (2022). This article expands upon the conference version in threefold: (i) complete proofs for all results are presented; and (ii) detailed and expanded examples are presented; and (iii) richer experimental results, including scalability experiments, simulation experiments and hardware experiments, are also provided.

2. Preliminary and problem formulation

2.1. Petri net model of multi-robot system

Let X be a set. We denote by X^* (respectively, X^ω) the set of all finite (respectively, infinite) sequences over X. We use $|X|$ to denote the cardinality of X. For any sequence $\rho = x_0 x_1 \cdots x_n \in X^*$, we use ρ_i

to denote the ith element, $\rho_{[i,j]}$ to denote the sub-sequence from the ith element to the jth element and $|\rho| = n+1$ to denote its length.

We consider a team of identical robots moving in the same workspace that in consistent of a set of regions with connectivity constraints. In this work, the connectivity and properties of the workspace are modeled as a weighted *Petri net* (PN).

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

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 cost function that assigns each transition an integer. We also denote by $\mathbf{g} = [g(t_1), g(t_2), \ldots, g(t_n)]^{\top} \in \mathbb{N}^n$ the *cost vector*. The *incidence matrix* is defined by $C = Post - Pre \in \mathbb{N}^{m\times n}$. For a transition $t \in T$, we use ${}^{\bullet}t = \{p \in P \mid Pre(p,t) = 1\}$ and $t^{\bullet} = \{p \in P \mid Post(p,t) = 1\}$ to denote its *input places* and *output places*, respectively. Input transitions ${}^{\bullet}p$ and output transitions $p^{\bullet}$ are defined analogously. We say 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.

Intuitively, each place represents a region in the workspace and each transition represents 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 its outgoing place. Therefore, we require that, for any place $p \in P$ and two transitions $t_1, t_2 \in T$ leading to this place, i.e., $t_1, t_2 \in {}^{\bullet}p$, we have $h(t_1) = h(t_2)$. In other words, given any place, all its input transitions have the same atomic propositions. We denote by P_{Π} the set of places whose input transitions have 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 $R(Q)$ the set of all reachable markings from M_0. Given a sequence $\sigma = t_0 t_1 \cdots$, we call the resulting marking sequence $\rho_{\sigma} = M_0 M_1 \cdots \in (R(Q))^{\omega}$ a *run* of Q, where $M_{i+1} = M_i + C(\cdot, t_i), \forall i = 0, 1, \ldots$. Also, given a feasible run ρ, we use σ_{ρ} to denote the sequence generating ρ. We denote by $\Sigma(Q)$ the set of all feasible sequences in Q. We use $\mathbf{y}_{\sigma} \in \mathbb{N}^{|T|}$ to denote the firing counting vector, i.e., $\mathbf{y}_{\sigma}(t) = k$ if transition t occurs k times in σ. Therefore, we have the following state function: $M' = M + C \cdot \mathbf{y}_{\sigma}$, which is a necessary condition for the reachability of a marking. Moreover, we consider a limited number of robots and the number of robots participating in the planning remains unchanged, which can be further expressed by the following two behavioral properties that any SM has (Seatzu et al., 2013), *boundedness* and *conservativeness*:

- bounded: If $\exists K \in \mathbb{N}, \forall M \in R(Q), p \in P, M(p) \leq K$;
- conservative: If $\forall M \in R(Q), \Sigma_{p\in P} M(p) = \Sigma_{p\in P} M_0(p)$;

We denote by P_0 the set of all places p such that $M_0(p) > 0$. Without loss of generality, we assume that each robot starts from a region with empty proposition, i.e., $P_0 \cap P_{\Pi} = \emptyset$.

2.2. Cyclic task

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

$$\phi = \text{true} \mid \pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U} \phi_2,$$

where $\pi \in \Pi$ is an atomic proposition, $\neg$ (negation) and $\wedge$ (conjunction) are standard Boolean operators, and $\mathcal{U}$ (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 propositions, i.e., $\Box$ is not allowed.

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 more details about the semantics of LTL formulae. Given a run ρ of Q and LTL formula ϕ, we denote by $\rho \vDash \phi$ if $h(\sigma_{\rho})$ satisfies ϕ and we use $\Sigma_{\phi}(Q)$ 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

$$\phi = \varphi \wedge \Box\Diamond\pi_{sur}, \tag{1}$$

where φ is an scLTL formula over 2^{Π} without next representing a finite-horizon task, which describes the transient part of ϕ, and $\pi_{sur} \in \Pi$ is a special proposition representing a cyclic task that should be satisfied *infinitely often*, which describes the steady part of ϕ. We denote by $\mathbf{1}_{sur} \in \{0,1\}^n$ the indicator vector for transitions satisfying π_{sur}, i.e., $\forall t \in T : [\mathbf{1}_{sur}(t) = 1] \Leftrightarrow [\pi_{sur} \in h(t)]$.

The objective LTL ϕ of form (1) can be translated into a *Büchi automaton* (BA) $\mathcal{A}_{\phi} = (\mathcal{Q}, q_0, 2^{\Pi}, \delta, \mathcal{Q}_F)$, where $\mathcal{Q}$ is a finite set of states, q_0 is the initial state, 2^{Π} is the power set of Π, $\delta : \mathcal{Q} \times 2^{\Pi} \to 2^{\mathcal{Q}}$ is the non-deterministic transition function and $\mathcal{Q}_F$ is the set of all accepting states. Without loss of generality, we assume that for ϕ, its corresponding BA $\mathcal{A}_{\phi}$ has only one initial state. Then the BA $\mathcal{A}_{\phi}$ exactly accepts all infinite sequences σ such that $\sigma \in \Sigma_{\phi}(Q)$, and there exists a path from q_0 and visits $\mathcal{Q}_F$ infinitely under $h(\sigma)$ in $\mathcal{A}_{\phi}$.

2.3. Problem formulation

Regarding the optimality condition, since our objective is to satisfy φ while achieving π_{sur} infinitely often, we refer to each satisfaction of π_{sur} as a *task cycle*. Then for any finite sequence $\sigma \in T^*$,

- the *total cost* incurred is $\mathbf{g}^{\top} \cdot \mathbf{y}_{\sigma}$; and
- the *number of task cycles* achieved is $N(\sigma) := \mathbf{1}_{sur}^{\top} \cdot \mathbf{y}_{\sigma}$.

Our objective is to minimize the average cost for each cycle as the sequence goes to infinity. Therefore, when given a PN Q and an LTL specification ϕ in form (1), for an infinite sequence $\sigma \in \Sigma(Q)$ such that $\rho_{\sigma} \vDash \phi$, we define the *average cost per task* of σ in Q by

$$\mathsf{Cost}_{Q}^{AveT}(\sigma) = \limsup_{n\to\infty} \frac{\mathbf{g}^{\top} \cdot \mathbf{y}_{\sigma_{[0,n]}}}{N(\sigma_{[0,n]})}. \tag{2}$$

It is known that the cost function in Eq. (2) can be minimized by an infinite sequence in Q of the following prefix-suffix form (Lv et al., 2021):

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

where $\sigma_{pre} \in T^*$ is a finite *prefix* representing the transient behavior of the system, while $\sigma_{sur} \in T^*$ is a finite *suffix* that needs to be executed for infinite number of times. Here $(\sigma_{sur})^{\omega}$ denotes the infinite repetition of finite sequence σ_{sur}. Our objective is to find such an optimal sequence. This leads to the *Multi-Robot Optimal Path Planning Problem for Average Cost Per Task* (MOPP-AT) that we solve in this work.

Problem 1. (MOPP-AT) Given a SM Q representing a team of identical robots moving in an environment and a temporal logic specification ϕ in form (1) for the team, where π_{sur} is the atomic proposition representing some regions needed to be surveilled infinitely often, find an optimal sequence $\sigma^{\star}$ for the team such that

- $\sigma^{\star} \in \Sigma_{\phi}(Q)$;
- $\sigma^{\star}$ is in the prefix-suffix form, namely $\sigma^{\star} = \sigma_{pre}^{\star}(\sigma_{sur}^{\star})^{\omega}$;

- $\forall \sigma \in \Sigma_\phi(Q) : \mathrm{Cost}_Q^{AveT}(\sigma^\star) \le \mathrm{Cost}_Q^{AveT}(\sigma)$.

Remark 1. As shown in Problem 1, part of our work is to plan a trajectory for the agent team meeting the specification ϕ. In the previous work (Kloetzer & Mahulea, 2020), this problem has been solved in the framework of PN, but they do not consider the optimal control problem. Here, we consider both qualitative and quantitative requirements.

3. Model reduction for planning

Before we provide our planning algorithm, we first provide a model reduction method for system model Q. Our approach is to simplify Q by abstracting those transitions with non-empty propositions, and get an abstracted SM Q_a, which has less places compared with Q but does not lose any information with respect to atomic propositions.

First, we recall some basic concepts from the graph theory. A *path* $l = v_1e_1v_2e_2\cdots e_{n-1}v_n$ in a *directed multi-graph* $\mathcal{G} = (V, E)$ is a sequence of vertices and edges such that $v_i \in V, e_i \in \langle v_i, v_{i+1}\rangle \subseteq E, \forall i \ge 1$, where $\langle v_i, v_{i+1}\rangle$ is the set of all edges from v_i to v_{i+1}. We use $\mathcal{E}_\mathcal{G}$ to denote the number of all multi-edges in $\mathcal{G}$ such that $\mathcal{E}_\mathcal{G} = \sum_{\langle v_i,v_{i+1}\rangle\subseteq E}(|\langle v_i, v_{i+1}\rangle|-1)$. A path is said to be a *cycle* if $v_1 = v_n$. For a cycle, if $\forall 1 < i < j < n, v_i \ne v_j, v_i \ne v_1$ and $v_j \ne v_1$, then we call it a *simple cycle*. Otherwise, it is a *compound cycle*. We use $\mathsf{L}_\mathcal{G}, \mathsf{C}_\mathcal{G}, \mathsf{SC}_\mathcal{G}$ and $\mathsf{CC}_\mathcal{G}$ to denote the sets of all paths, cycles, simple cycles and compound cycles in $\mathcal{G}$, respectively. Graph $\mathcal{G}$ is called a *finite graph* if $|V| \ne \infty$ and $|E| \ne \infty$. Also, $\mathcal{G}$ is called a *directed simple graph* if

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

And in this case, we use $l = e_1e_2\cdots e_n$ to denote a path for simplicity.

The main idea of the model reduction is the observation that, in many cases, the atomic propositions in Q are sparse, which means that $|P_\Pi| < |P|$. Therefore, given a temporal logic formula ϕ, the transition sequence satisfying ϕ in Q can be divided into infinite sub-sequences and each of them satisfies the following two conditions:

- the last transition of the sub-sequence is with atomic propositions;
- the length of the sub-sequence is one or all the remaining transitions are with empty propositions.

However, in every sub-sequence, we usually do not pay attention to those intermediate transitions with empty proposition. Therefore we can abstract every sub-sequence into its last transition with atomic propositions and let the weight value of the transition be the total weight of the sub-sequence. By this method, the original transition sequence can be simplified into a sequence, which only consists of transitions with atomic propositions.

The above heuristic discussion is the main principle in the model reduction of Q. We require that the place set and transition set of the simplified model should only contain $P_0 \cup P_\Pi$ and the abstracted transitions with atomic propositions, respectively, which can reduce the scale of the system model and further reduce the complexity of searching optimal sequence later. Next we formalize the above discussion and give the complete reduction method by the concepts of graph theory.

Given any $p \in P_0 \cup P_\Pi$ and $p' \in P_\Pi$, we denote by $L_{pp'}$ the set of all sequences starting from p to p' in Q without passing through any other places in P_Π. Formally, we have

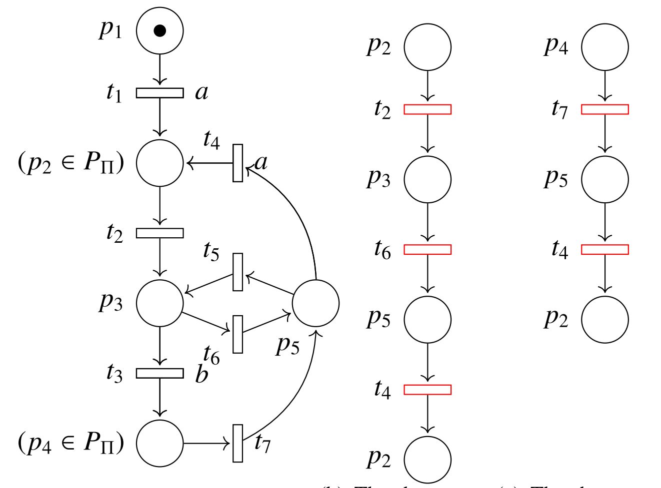
$$L_{pp'} = \left\{ \sigma \in T^* \,\middle|\, \begin{array}{c} {}^\bullet\sigma_0 = p \wedge \sigma_{|\sigma|}^\bullet = p' \wedge \\ (\forall 0 \le i < |\sigma|)[\sigma_i^\bullet \notin P_\Pi \wedge \sigma_i^\bullet = {}^\bullet\sigma_{i+1}] \end{array} \right\}.$$

We denote by $\hat{g}_{pp'}$ the minimum cost of sequences in $L_{pp'}$, i.e.,

$$\hat{g}_{pp'} = \min_{\sigma\in L_{pp'}} \mathbf{g}^\top \cdot \mathbf{y}_\sigma.$$

Note that Q can be considered as a simple graph with P as the vertex set and T as the edge set. Therefore, $\hat{g}_{pp'}$ can be easily calculated by the existing shortest path algorithm, such as the well-known Dijkstra's algorithm, and if $p' \in P_\Pi$ is not reachable from p, we define $\hat{g}_{pp'} = \infty$. Without loss of generality, when $\hat{g}_{pp'} \ne \infty$, we assume that there exists an unique sequence in $L_{pp'}$, denoted by $\hat{l}_{pp'}$, that achieves the minimum cost $\hat{g}_{pp'}$. Then we denote by $\mathfrak{L}$ the set of all the shortest sequences from $P_0 \cup P_\Pi$ to P_Π in Q, i.e.,

$$\mathfrak{L} = \{\hat{l}_{pp'} \in T^* \mid \exists p \in P_0 \cup P_\Pi, p' \in P_\Pi \text{ s.t. } \hat{g}_{pp'} \ne \infty\}.$$



(a) State machine Q (b) The shortest path from p_2 to p_2 (c) The shortest path from p_4 to p_2

Fig. 2. Graphs for Example 1, where $\forall t \in T, g(t) = 1$.

Remark 2. Although all the information about the atomic propositions can be summarized by the shortest sequences between the places in P_Π, we still need to acquire the shortest sequences from P_0 to P_Π to integrate the initial position information of the robots.

Example 1. Consider Petri net Q as shown in Fig. 2(a) with one token, where $P_0 = \{p_1\}$ and $P_\Pi = \{p_2, p_4\}$ with $\Pi = \{a, b\}$. Moreover, $\forall t \in T, g(t) = 1, h(t_1) = h(t_4) = a, h(t_3) = b$ and $\forall t \in T\backslash\{t_1, t_3, t_4\}, h(t) = \epsilon$. Here, we show the shortest paths from p_2 to p_2 and p_4 to p_2 in Figs. 2(b) and 2(c), respectively. Note that $\hat{l}_{p_2p_2} = t_2t_6t_4$ with $\hat{g}_{p_2p_2} = 3$ and $\hat{l}_{p_4p_2} = t_7t_4$ with $\hat{g}_{p_4p_2} = 2$. Furthermore, we have that $\mathfrak{L} = \{\hat{l}_{p_1p_2} = t_1, \hat{l}_{p_2p_2} = t_2t_6t_4, \hat{l}_{p_2p_4} = t_2t_3, \hat{l}_{p_4p_2} = t_7t_4, \hat{l}_{p_4p_4} = t_7t_5t_3\}$.

For every sequence in $\mathfrak{L}$, we abstract it as a new transition and we denote T^a as the set of all the abstracted transitions. We denote by $o^a : T^a \to \mathfrak{L}$ the bijection mapping between T^a to $\mathfrak{L}$. Then, based on the set T^a and mapping o^a, given a SM Q, we can given the definition of *abstracted SM* Q_a, which is a reduced model of Q.

Definition 1 (*Abstracted System*)**.** Given a SM $Q = (P, T, Pre, Post, M_0, \Pi, h, g)$, the abstracted transition set T^a and the projection function o^a, the abstracted SM is defined as an eight-tuple $Q_a = (P^a, T^a, Pre^a, Post^a, M_0^a, \Pi, h^a, g^a)$, where

- $P^a = P_0 \cup P_\Pi$ is the set of places;
- T^a is the set of transitions;
- Pre^a (resp., $Post^a$): $P^a \times T^a \to \{0, 1\}$ is the pre (resp., post)-incidence function defined by ${}^\bullet t$ and $t^\bullet$ as follows: $\forall t \in T^a$,
  - $|{}^\bullet t| = |t^\bullet| = 1$;
  - $[(p_1 = {}^\bullet t) \wedge (p_2 = t^\bullet)] \Rightarrow [o^a(t) = \hat{l}_{p_1p_2}]$;
- M_0^a is the initial marking defined as follows:
  - $\forall p \in P_0, M_0^a(p) = M_0(p)$;

– $\forall p \in P_\Pi, M_0^a(p) = 0$;

- Π is the set of all the atomic propositions;
- $h^a : T^a \to 2^\Pi$ is the labeling function that specifies a subset of atomic propositions $\Pi' \subseteq \Pi$ for every transition $t \in T^a$ defined by: $\forall t \in T^a, [(h^a(t) = h(\sigma_{|\sigma|})) \wedge (\sigma = o^a(t))]$, where $\sigma_{|\sigma|}$ denotes the last transition in σ.
- $g^a : T^a \to \mathbb{N}^+$ is the cost function that specifies a weight for every transition $t \in T^a$ defined by: $\forall t \in T^a, [(g^a(t) = \hat{g}_{p_1p_2}) \wedge (p_1 = {}^\bullet t) \wedge (p_2 = t^\bullet)]$.

To construct Q_a from SM Q, we need to calculate the shortest path between each pair of two places in P_Π, as well as from any place in P_0 to any place in P_Π. To this end, we can utilize the standard Dijkstra algorithm whose complexity is $\mathcal{O}((|T|+|P|)\cdot \log |P|)$. Therefore, the total computational complexity of constructing Q_a is $\mathcal{O}(P_\Pi \cdot (|P_0|+P_\Pi)\cdot(|T|+|P|)\cdot \log |P|)$.

Then, we define the projection function $\mathcal{P}' : \Sigma(Q_a) \to \Sigma(Q)$ as follows: $\forall \sigma = t_0t_1t_2\cdots \in \Sigma(Q_a)$, $\mathcal{P}'(\sigma) = o^a(t_0)o^a(t_1)\ o^a(t_2)\cdots \in \Sigma(Q)$. From the definition above, we can see that $\nexists t \in T^a, h^a(t) = \epsilon$ and Q_a is also a bounded and conservative SM. However, compared with Q, Q_a only needs $|P_0| + |P_\Pi|$ places to model the same system.

The following lemma shows that the optimal solution to Problem 1 can be found by the projection of the optimal sequence in Q^a with respect to (2).

Lemma 1. *There is a solution $\sigma^\star = \sigma^\star_{pre}(\sigma^\star_{sur})^\omega$ to Problem 1 if and only if there exists a prefix-suffix form sequence $\hat{\sigma}^a = \hat{\sigma}^a_{pre}(\hat{\sigma}^a_{suf})^\omega$ in Q_a such that*

- $\hat{\sigma}^a \in \Sigma_\phi(Q_a)$;
- $\forall \sigma \in \Sigma_\phi(Q_a) : \mathrm{Cost}^{AveT}_{Q_a}(\hat{\sigma}^a) \le \mathrm{Cost}^{AveT}_{Q_a}(\sigma)$.

Moreover, $\mathrm{Cost}^{AveT}_{Q_a}(\hat{\sigma}^a) = \mathrm{Cost}^{AveT}_{Q}(\mathcal{P}'(\hat{\sigma}^a)) = \mathrm{Cost}^{AveT}_{Q}(\sigma^\star)$.

Proof. This proof follows directly from the definition of the transition set T^a and the cost function g^a. □

Therefore, hereafter, we can use the abstracted SM Q_a to perform the optimal path synthesis procedure, which has less scale compared with Q.

4. Synthesis procedure

Since our PN model is bounded and conservative, we can construct its reachability graph to search for the optimal sequence. However, the state space of the reachability graph is usually exponential in the place set and transition set, which makes this method having no advantages on planning complexity compared with traditional automata based method. Therefore, we take an alternative method here based on a compact representation of the reachability graph called *basic reachability graph* (BRG) to perform the search.

Our synthesis procedure can be mainly divided into three parts. In Section 4.1, we first provide the definition of product PN between a SM and a BA. Then for any sequence in the product PN, we define the average cost per task measure. We use Q_a to construct the product PN Q'_a and reformulate the MOPP-AT in Q'_a as a new problem over the product system. In Section 4.2, we briefly introduce the theory related to BRG and construct the BRG $\mathcal{B}'$ to represent the reachability graph of Q'_a, which can further compress the solution space. Then, we prove that the search of the optimal sequence can be simplified to search a prefix-suffix form trajectory between these basic markings in $\mathcal{B}'$. Finally, in Section 4.3, we provide the complete planning algorithm based on the above three parts to get the solution.

4.1. Product Petri nets

Given a BA $\mathcal{A}$ and a SM Q, in order to find those sequences in Q satisfying ϕ, we define the *product PN* Q' between Q and $\mathcal{A}$. To this end, we first define a transition set T' and two projection functions o and o' as follows.

For any BA $\mathcal{A} = (Q, q_0, 2^\Pi, \delta, Q_F)$, we denote $E_\mathcal{A} = \{(q,q') : \exists \sigma \in 2^\Pi, q' \in \delta(q,\sigma)\}$ as the set of edges in $\mathcal{A}$. Given any BA $\mathcal{A}$ and a SM $Q = (P,T,Pre,Post,\Pi,h,g)$, we define a new transition set $T' \subseteq (T\times E_\mathcal{A})\cup T$ as follows: $\forall t \in T$, we have

- if $h(t) = \epsilon$, then $t \in T'$;
- if $h(t) \neq \epsilon$, then $\forall (q,q') \in E_\mathcal{A}$, $[q' \in \delta(q,h(t))] \Rightarrow [(t,(q,q')) \in T']$.

Then, we use $o : T' \to T$ to denote the projection function from T' to T, which is defined as follows: $\forall t' \in T'$, we have

- if $t' \in T$, then $o(t') = t'$;
- if $t' = (t,(q,q')) \in T \times E_\mathcal{A}$, then $o(t') = t$.

Moreover, we use $o' : T' \cap (T\times E_\mathcal{A}) \to E_\mathcal{A}$ to denote the projection function from $T' \cap (T\times E_\mathcal{A})$ to $E_\mathcal{A}$, which is defined as follows: $\forall t' = (t,(q,q')) \in T', o'(t') = (q,q')$.

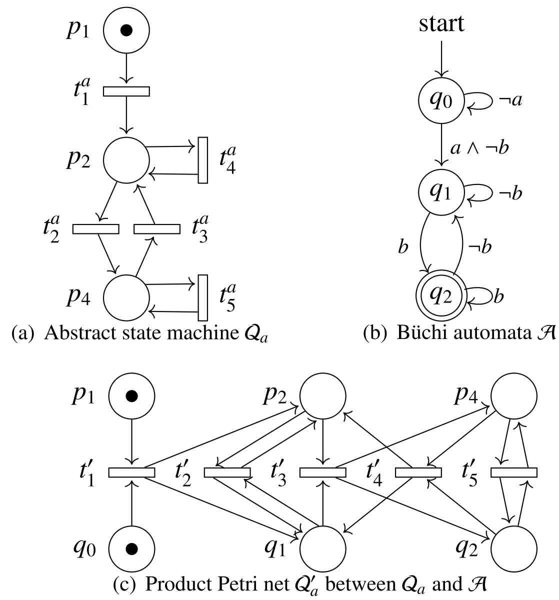
Based on the above transition set and projection functions, we can now give the definition of product PN Q'.

Definition 2 (*Product PN*)**.** Given a SM $Q = (P,T,Pre,Post,\Pi,h,g)$ with initial marking M_0 and BA $\mathcal{A} = (Q,q_0,2^\Pi,\delta,Q_F)$, we define the product of Q and $\mathcal{A}$ as a new PN $Q' = (P',T',Pre',Post',M'_0,\Pi,h',g')$, where

- $P' = P \cup Q$ is the set of places;
- T' is the set of 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: $\forall t \in T'$, we have
  – $[h(o(t)) = \epsilon] \Rightarrow [({}^\bullet t = {}^\bullet o(t)) \wedge (t^\bullet = o(t)^\bullet)]$;
  – $[h(o(t)) \neq \epsilon] \Rightarrow [({}^\bullet t \cap P = {}^\bullet o(t)) \wedge (t^\bullet \cap P = o(t)^\bullet) \wedge ({}^\bullet t \cap Q = q) \wedge (t^\bullet \cap Q = q')]$, where $o'(t) = (q,q')$;
- M'_0 is the initial marking defined by:
  – $\forall p \in P, M'_0(p) = M_0(p)$;
  – $M'_0(q_0) = 1$;
  – $\forall q \in Q \setminus q_0, M'_0(q) = 0$;
- Π is the set of atomic propositions;
- $h' : T' \to 2^\Pi$ is the labeling 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 the cost function that specifies a weight for every transition $t \in T'$ defined by: $\forall t \in T', g'(t) = g(o(t))$.

Since Q is bounded and conservative, from Definition 2, we know that Q' is also bounded and conservative. Moreover, we know that $\forall M \in R(Q'), \Sigma_{p'\in P'} M(p') = \Sigma_{p\in P} M_0(p) + 1$. Compared with Q, there is one more token in Q' at q_0 in M'_0. This token will be used to track the completion of the temporal logic task. In order to map sequence in the product system to the original system, we define the projection function $\mathcal{P} : \Sigma(Q') \to \Sigma(Q)$ by: $\forall \sigma = t_0t_1t_2\cdots \in \Sigma(Q')$, $\mathcal{P}(\sigma) = o(t_0)o(t_1)o(t_2)\cdots \in \Sigma(Q)$.

Example 1 (*Continued*)**.** Based on Definition 1, we construct the abstracted SM Q_a from Q as shown in Fig. 3(a) with $\Pi = \{a,b\}$, where $o^a(t_1^a) = \hat{l}_{p_1p_2} = t_1, o^a(t_2^a) = \hat{l}_{p_2p_4} = t_2t_3, o^a(t_3^a) = \hat{l}_{p_4p_2} = t_7t_4, o^a(t_4^a) = \hat{l}_{p_2p_2} = t_2t_6t_4$ and $o^a(t_5^a) = \hat{l}_{p_4p_4} = t_7t_5t_3$. Therefore, $g^a(t_1^a) = 1, g^a(t_2^a) = 2, g^a(t_3^a) = 2, g^a(t_4^a) = 3, g^a(t_5^a) = 3$ and $h^a(t_1^a) = h^a(t_3^a) = h^a(t_4^a) = a, h^a(t_2^a) = h^a(t_5^a) = b$. Then, a temporal task is given as $\phi = \Diamond a \wedge \Box\Diamond b$ and the corresponding Büchi automata is shown in Fig. 3(b). As we only consider one token in Q, which means that only one robot participates in the planning, and it cannot enable two transitions with atomic proposition a and b at the same time, therefore, we remove the edge from q_0 to q_2 in Fig. 3(b), which requires that a and b must be enabled at the same



(a) Abstract state machine Q_a (b) Büchi automata $\mathcal{A}$

(c) Product Petri net Q'_a between Q_a and $\mathcal{A}$

Fig. 3. Graphs for Example 1 (Continued).

time. Then, we construct the product PN Q'_a between Q_a and $\mathcal{A}$ as shown in Fig. 3(c), where $o(t'_1) = t_1^a, o(t'_2) = t_4^a, o(t'_3) = t_2^a, o(t'_4) = t_3^a$ and $o(t'_5) = t_5^a$. Therefore, $g'(t'_1) = 1, g'(t'_2) = 3, g'(t'_3) = 2, g'(t'_4) = 2, g'(t'_5) = 3$ and $h'(t'_1) = h'(t'_2) = h'(t'_4) = a, h'(t'_3) = h'(t'_5) = b$. Note that we can also construct the product PN between Q and $\mathcal{A}$, which has 8 places and 21 transitions, however Fig. 3(c) is already enough to illustrate the principle of Definition 2 and we will not repeat it here. Moreover, for the convenience of figure presentation, we omit the inaccessible transitions in Fig. 3(c) such that we can also incorporate transition t_1^a with edge $\delta(q_2, \neg b) = q_1$ into a new transition in Fig. 3(c) according to Definition 2. However, it will never be enabled due to the transition constrains.

Similar to the original PN Q, for the product PN Q', we also define $\mathbf{g}' \in \mathbb{N}^{|T'|}$ as the cost vector, and for any finite sequence $\sigma \in (T')^*$, still denote by $N'(\sigma)$ the number of task cycles. Then for finite sequence $\sigma \in (T')^*$, we define the *finite average cost per task* by

$$\mathrm{Cost}_{Q'}^{FT}(\sigma) = \frac{(\mathbf{g}')^\top \cdot \mathbf{y}_\sigma}{N'(\sigma)}. \tag{3}$$

When $\sigma \in \Sigma(Q')$ is an infinite sequence in Q', we also define the *average cost per task* by

$$\mathrm{Cost}_{Q'}^{AveT}(\sigma) = \limsup_{n\to\infty} \mathrm{Cost}_{Q'}^{FT}(\sigma_{[0,n]}). \tag{4}$$

Definition 3 (*Accepting Sequence*)**.** For any infinite sequence σ in Q', we say σ is an *accepting sequence* if accepting states are involved infinite number of times, i.e., $\mathbf{1}_F^\top \cdot \mathbf{y}_\sigma = \infty$, where $\mathbf{1}_F \in \{0,1\}^n$ is the indicator vector such that $\forall t \in T' : [\mathbf{1}_F(t) = 1] \Leftrightarrow [t^\bullet \cap Q_F \neq \emptyset]$. We denote by $\Sigma_\phi(Q')$ the set of all the accepting sequences in Q'.

The following lemma shows that an accepting sequence $\tilde{\sigma}$ with the minimal average cost per task with respect to (4) can be found by searching over all the prefix-suffix form sequences in Q'.

Lemma 2. *There exists at least one accepting sequence $\tilde{\sigma}$ in Q' satisfying*

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

Proof. Since Q' is bounded and conservative, we can construct its reachability graph $\mathcal{R}$. Since the set of all the transitions $t \in T'$ such that $\mathbf{1}_F(t) = 1$ is finite and an accepting sequence is in infinite length, at least one transition in the above set must be enabled infinite times. We use $\mathrm{SC}_F = \{\sigma \in \mathrm{SC}_\mathcal{R} \mid \exists i \in \mathbb{N}, \mathbf{1}_F(\sigma_i) = 1\}$ to denote the set of all the simple cycles in $\mathcal{R}$ containing edges $t \in T'$ in the above transition set. Then let $\tilde{\sigma}_{suf}$ be the simple cycle with the minimum average cost per task over all cycles such that

$$\tilde{\sigma}_{suf} = \mathrm{argmin}_{\sigma \in \mathrm{SC}_F} \mathrm{Cost}_{Q'}^{FT}(\sigma).$$

Let $\tilde{\sigma}_{pre}$ be any finite sequence from M'_0 to $\tilde{\sigma}_{suf}$. Let $\tilde{\sigma} = \tilde{\sigma}_{pre}(\tilde{\sigma}_{suf})^\omega$, and the claim follows. □

Note that, any accepting sequence in Q' can be projected back to a sequence in Q, which satisfies the given formula, and vice versa. This result is summarized by the following lemma, which makes a connection between the accepting transition sequences in Q and Q'.

Lemma 3. *Given BA $\mathcal{A}_\phi$, SM Q and their product PN Q', then*

- *$\forall \sigma' \in \Sigma_\phi(Q')$, $\mathcal{P}(\sigma') \in \Sigma_\phi(Q)$ and $\mathrm{Cost}_{Q'}^{AveT}(\sigma') = \mathrm{Cost}_{Q}^{AveT}(\mathcal{P}(\sigma'))$. If σ' is in prefix-suffix form, then $\mathcal{P}(\sigma')$ is also in prefix-suffix form.*
- *$\forall \sigma \in \Sigma_\phi(Q)$, $\exists \sigma' \in \Sigma_\phi(Q'), \mathcal{P}(\sigma') = \sigma$ and $\mathrm{Cost}_{Q}^{AveT}(\sigma) = \mathrm{Cost}_{Q'}^{AveT}(\sigma')$. If σ is in prefix-suffix form, then σ' is also in prefix-suffix form.*

Proof. For the first item, since $g'(t) = g(o(t))$ holds for all $t \in T'$, we have $\forall \sigma' \in \Sigma_\phi(Q'), \mathrm{Cost}_{Q'}^{AveT}(\sigma') = \mathrm{Cost}_{Q}^{AveT}(\mathcal{P}(\sigma'))$ with $\mathcal{P}(\sigma') \in \Sigma_\phi(Q)$. When σ' is in the prefix-suffix form, $\mathcal{P}(\sigma')$ is also in the prefix-suffix form, which follows directly from the definition of projection functions o and $\mathcal{P}$.

For the second item, under any sequence $\sigma \in \Sigma_\phi(Q)$, there exists an infinite path l in $\mathcal{A}_\phi$ from q_0 to Q_F and visits Q_F infinite times. Based on the definition of transition set T' of the product PN, we can always find the corresponding sequence σ' in Q' with $\sigma' \in \Sigma_\phi(Q'), \mathcal{P}(\sigma') = \sigma$ and $\mathrm{Cost}_{Q}^{AveT}(\sigma) = \mathrm{Cost}_{Q'}^{AveT}(\sigma')$. If σ is in prefix-suffix form, then l must also be in prefix-suffix form. Therefore, σ' is also in prefix-suffix form. □

In Section 3, we have reduced Q to an abstracted SM Q_a and demonstrated the equivalence of the two models to solve Problem 1. Therefore, we can construct the product PN Q'_a with the abstracted SM Q_a instead of Q to search the optimal sequence for Problem 1 with lower complexity. This result is summarized by the following corollary.

Corollary 1. *Let $\sigma^a = \sigma_{pre}^a(\sigma_{suf}^a)^\omega$ be the optimal prefix-suffix form sequence with respect to (4) in Q'_a, then $\mathcal{P}'(\mathcal{P}(\sigma^a)) \in \Sigma_\phi(Q)$ is the solution to Problem 1.*

Proof. From Lemmas 1–3, this corollary follows directly. □

This leads to the reformulation of MOPP-AT in Q'_a.

Problem 2 (*MOPP-AT-Q'_a*)**.** Given a BA $\mathcal{A}$, an abstracted SM Q_a and their product PN Q'_a, find an optimal sequence σ^a in Q'_a such that

- $\sigma^a \in \Sigma_\phi(Q'_a)$;
- σ^a is in prefix-suffix form, namely $\sigma^a = \sigma_{pre}^a(\sigma_{suf}^a)^\omega$;
- $\forall \sigma \in \Sigma_\phi(Q'_a) : \mathrm{Cost}_{Q'_a}^{AveT}(\sigma^a) \le \mathrm{Cost}_{Q'_a}^{AveT}(\sigma)$.

4.2. Basic reachability graphs for planning

In Ma et al. (2016), a compact structure, called *basis reachability graph* (BRG), to represent the reachability graph of a PN is proposed, based on which the author can solve the finite sequence reachability problem more efficiently compared with constructing the complete reachability graph. In this paper, we also use BRG to reduce trajectory synthesis complexity. But the difference is that we concentrate on the infinite sequence cyclic task planning problem. Before giving the definition of BRG, we recall some related concepts briefly and the interested reader can refer to Ma et al. (2016).

Definition 4 (*Basis Partition of Transitions*)**.** Given a PN $Q = (P, T, Pre, Post, \Pi, h, g)$, 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 5 (*Explanations*)**.** Given a basis partition $\varpi = (T_E, T_I)$, a marking M, and a transition $t \in T_E$,

- the set of *explanations* of t at M is defined by

  $\Sigma(M,t) = \{\sigma \in T_I^* \mid M[\sigma\rangle M', M' \geq Pre(\cdot,t)\}$

- the set of *minimal explanations* of t at M is defined by

  $\Sigma_{min}(M,t) = \{\sigma \in \Sigma(M,t) \mid \nexists \sigma' \in \Sigma(M,t), \mathbf{y}_{\sigma'} \lneq \mathbf{y}_\sigma\}$

  and we define $Y_{min}(M,t) = \{\mathbf{y}_\sigma \in \mathbb{N}^{|T_I|} \mid \sigma \in \Sigma_{min}(M,t)\}$ as the set of *minimal explanation vectors*.

Intuitively, $\Sigma(M,t)$ means that from M if we want to enable the explicit transition t by firing only implicit transitions, then some sequence $\sigma \in \Sigma(M,t)$ must fire. Further, $\Sigma_{min}(M,t)$ is the set of sequences in $\Sigma(M,t)$ with minimal firing sequences and $Y_{min}(M,t)$ is the set of these minimal firing vectors. Based on the above notions, we can give the definition of BRG as follows.

Definition 6 (*Basic Reachability Graph* Ma et al., 2016)**.** Given a bounded PN Q and a basis partition $\varpi = (T_E, T_I)$, its basis reachability graph is a four-tuple $\mathcal{B} = (\mathcal{M}, Tr, \Delta, M_0)$, such that

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

From Definition 6, we know that $\mathcal{B}$ is a finite directed multi-graph with $\mathcal{M}$ being the vertex set and Δ being the edge set. It is worth remarking that the basis partition $\varpi = (T_E, T_I)$ may not be unique and different basis partitions correspond to different BRGs. The reader is referred to Ma et al. (2016) for how to select a good basis partition such that the BRG is reasonably small. Note that every edge in $\mathcal{B}$ involves explicit transitions, but the system may also move implicitly. This leads to the following definition.

Definition 7 (*Implicit Reach*)**.** Given a basis partition $\varpi = (T_E, T_I)$ and a basis marking $M_b \in \mathcal{M}$, we define

$$R_I(M_b) = \{M \in \mathbb{N}^m \mid \exists \sigma \in T_I^*, M = M_b + C \cdot \mathbf{y}_\sigma\} \quad (5)$$

as the *implicit reach* of M_b in BRG $\mathcal{B}$.

Then, we prove that the transition sequence σ in the above definition is a finite sequence by the following lemma.

Lemma 4. *Given any basis marking $M_b \in \mathcal{M}$ and $M \in R_I(M_b)$ such that $M_b[\sigma\rangle M$ with $\sigma \in T_I^*$, we have that $\mathbf{1}^\top \cdot \mathbf{y}_\sigma < \infty$, where $\mathbf{1} \in \mathbb{N}^{|T|}$ is a vector with all elements being one.*

Proof. We prove it by contradiction. Suppose $\mathbf{1}^\top \cdot \mathbf{y}_\sigma = \infty$, which means some transition $t \in T_I$ occurs infinite times in σ. There are two reasons for this case: either we have infinite number of robots or there exists a cycle in σ. However, from the definition of T_I and the boundedness property of Q'_a, neither of these two reasons holds, which contradicts the assumption. □

With the above contents ready, we now introduce the following main property of BRG.

Proposition 1 (Ma et al., 2016)**.** *Given a PN Q, a basis partition $\varpi = (T_E, T_I)$ and a marking $M \in R(Q)$, let $\mathcal{B} = (\mathcal{M}, Tr, \Delta, M_0)$ be the BRG. Then the following two statements are equivalent:*

- *There exists a sequence $\sigma = \sigma_1 t_1 \cdots \sigma_n t_n \sigma_{n+1}$ in Q, where $\sigma_i \in T_I^*$ and $t_i \in T_E$, such that $M_0[\sigma\rangle M$;*
- *There exists a path $l = M_0 \xrightarrow{t_1,\mathbf{y}_1} M_{b,1} \xrightarrow{t_2,\mathbf{y}_2} \cdots \xrightarrow{t_n,\mathbf{y}_n} M_{b,n}$ in $\mathcal{B}$ such that $M \in R_I(M_{b,n})$.*

Note that in order to track a reachable marking in Q, we usually need to construct the complete reachability graph of Q and this approach is in general costly. However, the above proposition shows that any reachable marking $M \in R(Q)$ can also be tracked in $\mathcal{B}$, whose state space is generally much smaller than that of the reachability graph, by following a path l in $\mathcal{B}$ and a finite sub-sequence starting from a basis marking which consists only of implicit transitions. Therefore, we propose an algorithm based on the above advantages of BRG to synthesize trajectories for robots, which brings significant advantages from the point of view of the computational effort. Given product PN Q'_a and a basis partition $\varpi = (T_E, T_I)$, we construct the basis reachability graph (BRG) $\mathcal{B}' = (\mathcal{M}', Tr', \Delta', M'_0)$ as in Definition 6.

Let $\sigma^a = \sigma^a_{pre}(\sigma^a_{suf})^\omega$ be the prefix-suffix form solution that solves Problem 2. From Proposition 1 and Lemma 4, there is an infinite path l_a in $\mathcal{B}'$ in the following form:

$$M'_0 \xrightarrow{t_1,\mathbf{y}_1} M_{b,1} \xrightarrow{t_2,\mathbf{y}_2} \cdots \xrightarrow{t_n,\mathbf{y}_n} M_{b,n} \xrightarrow{t_{n+1},\mathbf{y}_{n+1}} \cdots.$$

As $\mathcal{M}'$ is a finite set and l_a is an infinite sequence which contains infinite basis markings, at least one basis marking appears infinitely in l_a. For any marking M_i that appears infinitely in l_a, we can divide l_a into infinite cycles which start and end at M_i. Note that, there might be a finite transition sequence σ_f from M'_0 to M_i in $\mathcal{B}'$.

For any finite path $l = M_0 \xrightarrow{t_1,\mathbf{y}_1} M_{b,1} \xrightarrow{t_2,\mathbf{y}_2} \cdots \xrightarrow{t_n,\mathbf{y}_n} M_{b,n}$ in $\mathcal{B}'$, we use σ_l to denote the transition sequence in l such as: $M_0[\sigma_l\rangle M_{b,n}$. Note that the firing of explicit transitions are captured by $t_1, \ldots, t_n$, while the firing of implicit transitions are captured by $\mathbf{y}_1, \ldots, \mathbf{y}_n$. Therefore, we define the *firing counting vector* of path l by

$$\mathbf{y}_l = \mathbf{y}_{t_1 t_2 \ldots t_n} + \sum_{i=1}^{n} \mathbf{y}_i.$$

With the above concepts, we introduce the following definition.

Definition 8 (*Accepting Cycles*)**.** Let $l \in \mathsf{SC}_{\mathcal{B}'}$ be a simple cycle in $\mathcal{B}'$. We call l an *accepting cycle* if $\mathbf{1}_F^\top \cdot \mathbf{y}_l > 0$ and we denote by $\mathsf{SC}^F_{\mathcal{B}'}$ the set of all simple accepting cycles.

Note that the set of all simple accepting cycles can be found. For the simplification of narration in the following Proposition 2, we make the following mild assumption: for any $l, l' \in \mathsf{SC}^F_{\mathcal{B}'}$, we have

$$[(\mathsf{Cost}^{FT}_{Q'_a}(\sigma_l) \neq \infty) \wedge (\mathsf{Cost}^{FT}_{Q'_a}(\sigma_{l'}) \neq \infty)]$$
$$\Rightarrow [\mathsf{Cost}^{FT}_{Q'_a}(\sigma_l) \neq \mathsf{Cost}^{FT}_{Q'_a}(\sigma_{l'})].$$

This assumption is not essential and only for the sake of technical development of Proposition 2, whose absence will not influence the conclusion. Based on this assumption, we know that there exists a simple cycle $l^\star \in \mathsf{SC}^F_{\mathcal{B}'}$ such that: $\forall l \in \mathsf{SC}^F_{\mathcal{B}'}, \mathsf{Cost}^{FT}_{Q'_a}(\sigma_{l^\star}) < \mathsf{Cost}^{FT}_{Q'_a}(\sigma_l)$.

The following result states that, the planning problem has a solution if and only if a simple accepting cycle always exists and the solution to the problem can be constructed from $l^\star$.

Algorithm 1: Algorithm for MOPP-AT

Input: Weighted SM $\mathcal{Q}$, a temporal logic formula ϕ in form (2)
Output: Prefix-suffix form solution $\sigma^\star$ for Problem 1

1 Construct the Büchi automata $\mathcal{A}_\phi$;
2 Construct $\mathcal{Q}_a$ based on $\mathcal{Q}$;
3 Construct the Product PN $\mathcal{Q}'_a$ based on $\mathcal{Q}_a$ and $\mathcal{A}_\phi$;
4 Construct BRG $\mathcal{B}'$ based on $\mathcal{Q}'_a$;
5 Compute the set of all simple accepting cycles $\mathsf{SC}^F_{\mathcal{B}'}$;
6 **if** $SC^F_{\mathcal{B}'} = \emptyset$ **then**
7 **return** There is no solution to Problem 1;
8 **else**
9 Find $l^\star \in \mathsf{SC}^F_{\mathcal{B}'}$ such that $\forall l \in \mathsf{SC}^F_{\mathcal{B}'}$: $\mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_{l^\star}) \leq \mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_l)$
10 Set $\sigma^a_{suf} \leftarrow \sigma_{l^\star}$ and let M' be the initial marking in $l^\star$;
11 Find an arbitrary feasible sequence from M'_0 to M' in $\mathcal{Q}'_a$, and define it as σ^a_{pre};
12 Let $\sigma^\star_{pre} \leftarrow \mathcal{P}'(\mathcal{P}(\sigma^a_{pre}))$ and $\sigma^\star_{sur} \leftarrow \mathcal{P}'(\mathcal{P}(\sigma^a_{suf}))$;
13 **return** The solution to Problem 1 is $\sigma^\star = \sigma^\star_{pre}(\sigma^\star_{sur})^\omega$.

Proposition 2. *Problem 2 has a solution if and only if $\mathsf{SC}^F_{\mathcal{B}'} \neq \emptyset$ and if the solution indeed exists, then $\sigma^a = \sigma^a_{pre}(\sigma_{l^\star})^\omega$ is a solution to Problem 2, where σ^a_{pre} is any finite sequence from M'_0 to $l^\star$ in $\mathcal{Q}'_a$.*

Proof. ($\Rightarrow$) We prove it by contradiction. Suppose $\mathsf{SC}^F_{\mathcal{B}'} = \emptyset$, which means that all the cycles divided in l_a do not pass any transition t with $\mathbf{1}_F(t) = 1$. Furthermore, as σ_f is a finite sequence, l_a fails to visit Q_F infinite times. This means that $\sigma^a = \sigma^a_{pre}(\sigma^a_{suf})^\omega$ is not a solution to Problem 2, thus causing a contradiction.

($\Leftarrow$) We also prove it by contradiction. Suppose σ^a is not a solution to Problem 2, which means there exists another sequence $\sigma' = \sigma'_{pre}(\sigma'_{sur})^\omega$ in $\mathcal{Q}'_a$ with

$$\mathsf{Cost}^{AveT}_{\mathcal{Q}'_a}(\sigma') < \mathsf{Cost}^{AveT}_{a}(\sigma^a). \tag{6}$$

From Proposition 1, let l_a be the infinite path in $\mathcal{B}'$ that corresponds to σ'. Note that as σ_f is a finite sequence, if (6) establishes, there must exist at least a simple cycle $l \in \mathsf{SC}^F_{\mathcal{B}'}$ with

$$\mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_l) < \mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_{l^\star})$$

that appears in l_a infinite times. However, as $l^\star$ is the cycle with the smallest average cost per task, we cannot find such l. Therefore, there exists no such sequence σ' and $\sigma^a = \sigma^a_{pre}(\sigma_{l^\star})^\omega$ is a solution to Problem 2. ☐

4.3. Computation of solution to Problem 1

With the above developments, we propose the overall algorithm for Problem 1 as shown in Algorithm 1. This algorithm works as follows. First, it constructs the Büchi automata $\mathcal{A}_\phi$ and constructs $\mathcal{Q}_a$ from $\mathcal{Q}$ (lines 1–2). Next, the product PN $\mathcal{Q}'_a$ is constructed from $\mathcal{A}_\phi$ and $\mathcal{Q}_a$ (line 3). Then based on the BRG $\mathcal{B}'$ constructed from the product PN $\mathcal{Q}'_a$ (line 4), it computes the smallest average cost per task simple cycle $l^\star$ as the candidate suffix part of the solution to Problem 2 (lines 5–10). Then we connect the initial marking with the optimal cycle by an arbitrary finite sequence since this transient part does not contributes to the overall cost (line 11). Finally, we project the sequences in $\mathcal{Q}'_a$ back to the original system in order to generate the solution path (lines 12–13).

Finally, we discuss the implementation and complexity issues in the above algorithm. Let $|\mathcal{M}'|$ and $|\mathcal{E}'|$ be the number of vertices and multi-edges in BRG $\mathcal{B}'$, respectively. To find the simple accepting cycle $l^\star \in \mathsf{SC}^F_{\mathcal{B}'}$ with minimal cost, our approach is to use the standard cycle search algorithm to find all simple cycles. Then we simply compare each of them and select the minimal cost one that is also accepting. In the worst-case, there are at most $(|\mathcal{M}'| + |\mathcal{E}'|)^3 \cdot 2^{|\mathcal{M}'|+|\mathcal{E}'|}$ simple cycles (Sedgewick & Wayne, 2007). However, in practice, the actual complexity is much lower as we will show later in the numerical experiments. To find the transient path σ^a_{pre}, we can just perform a reachability search such as DFS, whose complexity is linear in $\mathcal{B}'$.

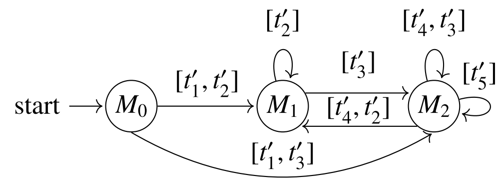


Fig. 4. BRG $\mathcal{B}'$ for $\mathcal{Q}'$.

Remark 3. In Section 2.2, we assume that $\mathcal{A}_\phi$ has only one initial state. For the case of multiple initial states, it is equivalent to perform Algorithm 1 on $\mathcal{A}_\phi$ for every initial state. In other words, when constructing the initial marking of the product PN, in addition to the token distribution of the SM, we consider another token in any one of the initial states of $\mathcal{A}_\phi$, while other initial states of $\mathcal{A}_\phi$ having zero, and continue with the subsequent analysis process to get the optimal solution with respect to the chosen initial state. Then, we repeat the above process for every initial state of $\mathcal{A}_\phi$. Finally, we compare the values of average cost per task of these sequences and the optimal solution to Problem 1 is just the sequence with the minimum value.

Example 1 (*Continued*). We construct the BRG $\mathcal{B}'$ for product PN $\mathcal{Q}'_a$ as shown in Fig. 4 with $T_E = \{t'_2, t'_3, t'_5\}$ and $T_I = \{t'_1, t'_4\}$. If we use $M = [p_1, p_2, p_4, q_0, q_1, q_2]$ to denote the token distribution vector, then $M_0 = [1, 0, 0, 1, 0, 0]$, $M_1 = [0, 1, 0, 0, 1, 0]$ and $M_2 = [0, 0, 1, 0, 0, 1]$. Further, we use $[t, t']$ to denote the mixed transition vector including both explicit and implicit transitions attached to the corresponding edge that should be enabled successively, such as $M_0[t'_1 t'_2\rangle M_1$, $M_1[t'_2\rangle M_1$ etc. Note that as $\mathbb{T}_F = \{t'_3, t'_5\}$, we only need to enumerate three cycles: $l_1 = M_1[t'_3]M_2[t'_4, t'_2]M_1$ with $\mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_{l_1}) = 7$, $l_2 = M_2[t'_4, t'_3]M_2$ with $\mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_{l_2}) = 4$ and $l_3 = M_2[t'_5]M_2$ with $\mathsf{Cost}^{FT}_{\mathcal{Q}'_a}(\sigma_{l_3}) = 3$. Therefore, we can get the optimal solution $\sigma^a = t'_1 t'_3 (t'_5)^\omega$ for Problem 2. Further, we have $\mathcal{P}(\sigma^a) = t^a_1 t^a_2 (t^a_5)^\omega$ and finally get the optimal solution $\sigma^\star = \mathcal{P}'(\mathcal{P}(\sigma^a)) = t_1 t_2 t_3 (t_7 t_5 t_3)^\omega$ with $\mathsf{Cost}^{AveT}_{\mathcal{Q}'_a}(\sigma^\star) = 3$ for Problem 1. Moreover, we can also construct the BRG for the product PN $\mathcal{Q}'$ constructed from $\mathcal{Q}$ and $\mathcal{A}$, and use this BRG to solve the solution for Problem 1. However, it has 7 states and 296 edges with obvious structural redundancy with respect to the states and edges compared with $\mathcal{B}'$, thus leading to the low computation efficiency.

Remark 4. In the above example, the difference between the two sets P_Π and P of $\mathcal{Q}$ is not significant since we choose a compact system for the purpose of illustration. When the atomic propositions in $\mathcal{Q}$ become more sparse as most of the practical cases, the structural redundancy of the BRG based on the non-abstract product PN with respect to the number of states and edges will be more significant compared with the one based on the abstract product PN. Therefore, in general case, we can directly use the abstract product PN to construct BRG to synthesize optimal solution.

5. Experiment Results

In this section, we provide a set of experiments to illustrate our results. In Section 4.1, we provide numerical experiments by increasing the size of the environment and the number of robots to illustrate

Table 1
Summary statistics for different number of robots on a fixed size of 7 × 7 grid environment.

N	TS⊗$\mathcal{A}$			$\mathcal{B}$			
	$\vert S\vert$	$\vert E\vert$	t_1 (s)	$\vert\mathcal{M}\vert$	$\vert\Delta\vert$	$\mathcal{E}_B$	t_2 (s)
1	97	333	1.76	6	20	1	0.09
2	2353	27729	48.53	17	119	15	0.35
3	57097	2309553	4636.96	36	347	54	1.29
4	–	–	–	65	754	130	2.577
5	–	–	–	106	1390	255	20.64
6	–	–	–	161	2305	441	123.88
7	–	–	–	232	3549	8700	598.93
8	–	–	–	321	5172	1044	2576.52
9	–	–	–	430	7224	1485	7838.39

Table 2
Summary statistics for different sizes of environments with a fixed number of 3 robots.

$\mathcal{W}$	TS⊗$\mathcal{A}$			$\mathcal{B}$			
	$\vert S\vert$	$\vert E\vert$	t_1 (s)	$\vert\mathcal{M}\vert$	$\vert\Delta\vert$	$\mathcal{E}_B$	t_2 (s)
4 × 4	1879	50733	23.56	27	152	31	0.11
5 × 5	7453	242653	895.36	30	192	31	0.24
6 × 6	22409	833193	2315.84	50	465	65	2.41
7 × 7	57097	2309553	4636.93	36	347	54	1.29
8 × 8	128095	5509813	8567.23	67	609	77	5.41
9 × 9	–	–	–	36	296	45	0.35

the computational efficiency and scalability of our method compared with the product automaton based method. In Section 4.2, we provide a simulation experiment on an 8 × 8 grid system with three robots. Finally, in Section 4.3, we perform a hardware experiment to assess the real-world feasibility of our method. All simulations are implemented by integrating `MATLAB`, `Python 3.7` and robot simulation platform `V-REP 4.2.0` on a PC with 64 cores with 3.30 GHz processors and 64 GB of RAM. The robots used in the hardware experiment are two `Turtlebot3-Burger` mobile robots.

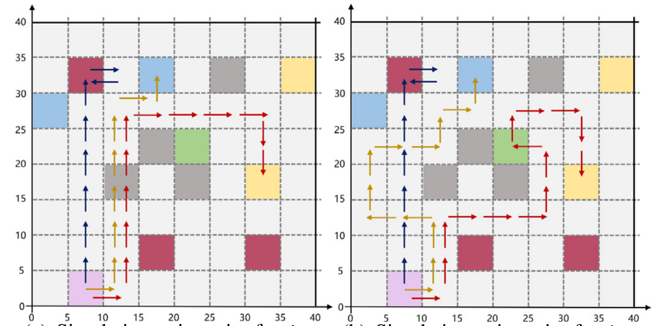
5.1. Scalability results

In this part, we increase the number of robots and the size of the environment respectively and compare the scalability of our BRG based method with the baseline method of product automaton (Wolff, Topcu, & Murray, 2012) with respect to the two mentioned parameters in the planning problem. Therefore, for each value of the two parameters, we conduct two groups of random experiments, each with 20 different grid environments, which are solved by our BRG based method and product automaton based method respectively. For every environment, the number of grids with atomic proposition is chosen uniformly between 2 and 12 and the distribution of grids with atomic proposition also obeys uniform distribution. Then, the LTL specification $\phi = \varphi \wedge \Box\Diamond\pi_{sur}$ is generated randomly such that φ contains at most twenty operators except *next*. We consider the edge weights the same and the initial grids for all robots the same in every environment.

We record the mean number of variables and the mean time to find the solution. For the convenience of statistic presentation, we collect all data and compile them into two tables as shown in Tables 1 and 2, without showing the trajectories. We use N to denote the number of robot and TS × $\mathcal{A}$ to denote the product automaton system with S being the state set, E being the edge set and t_1 being the solution time. Furthermore, we use $\mathcal{B}$ to denote the BRG with $\mathcal{M}$ being the basis marking set, Δ being the edge set, $\mathcal{E}_B$ being the multi-edge set and t_2 being the solution time. All computation times and number of variables in the two tables are presented as the mean values of all experiments and we use "–" for the entries which we fail to compute due to insufficient RAM (64 GB) or very high computation time (>10 000 s).

Scalability in the Number of Robots: In this numerical experiment, we fix the size of environment as 7 × 7 and increase the number of robots from 1 to 9. The experiment data indicate that N can have a significant effect on the computation time to obtain the optimal solution for both methods. For product automaton based method, it is known that we need to construct the complete state space and the numbers of states and edges are exponential with respect to N. Furthermore, in order to obtain the optimal solution, 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 at most. Therefore, from Table 1, we can see that the size of the product system grows almost exponentially fast when N increases and it is already infeasible to find solution by this method when there are more than 3 robots. However, for our BRG based method, although all synthesis parameters also increases when N increases, the growing speed is much slower compared with the former method. This is because we do not need to construct the complete state space for the purpose of searching optimal plans. The optimal solution is searched only in the compact representation of the state space. Therefore, the experiment result shows that our method has better scalability with respect to the number of robots compared with the baseline method.

Scalability in the Size of Environment: In this experiment, we maintain the number of robots as 3 and vary the size of environment from 4 × 4 to 9 × 9. The experimental results suggest that the size of the environment $\mathcal{W}$ has a relative small effect on the computation time to obtain the optimal solution for both methods compared with N. However, when $\mathcal{W}$ > 8, the automata-based approach still cannot find solution within the specified time. On the other hand, for the BRG based method, the experiment data shows that the numbers of states and edges and the time for solution are hardly affected by $\mathcal{W}$ and the values of all the four variables remain very stable across the process of chances in environment size. The main reason for this result is that our approach abstracts away those irrelevant information from the plant model before constructing the BRG. This explains why all synthesis parameters remain rather stable when the size of the grid map increases. Therefore, this experiment result shows that our method can achieve significant speed up with respect to computation efficiency and it has significantly better scalability with respect to the size of environment than the baseline method.



(a) Simulation trajectories for ϕ_1 (b) Simulation trajectories for ϕ_2

Fig. 5. Simulation results for Section 4.2, where blue, yellow and red trajectories are for R_1, R_2 and R_3 respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.2. Simulation experiments

In this section, we describe two task planning case studies for three identical robots to show the simulation trajectories. Consider a working space with 64 regions as shown in Fig. 5. Three robots R_1, R_2 and R_3 start from their initial region respectively (the pink region at the lower left quarter of the figure) to participate in the planning process. At each time instant, only one robot can move left/right/up/down to its adjacent region with cost equaling to one unit, which means that we do not allow multiple robots moving together. Consider the atomic proposition set $\Pi = \{B, R, Y, E, G\}$, where B, R, Y, E and G represent the blue, red, yellow, gray and green colors respectively. Some regions are marked with atomic propositions, while the other regions are with empty proposition. When a robot arrives a region, it can collect the color information here once. However, if it wants to collect the information again, it must leave the region and re-enter, which means that when it stays here, it cannot collect any information further.

The first temporal logic task we consider is given as follows:

$$\phi_1 = \Diamond Y \wedge \Diamond B \wedge \Box\Diamond R,$$

which requires that blue and yellow regions should be visited eventually and red regions should be visited infinitely often.

We use Algorithm 1 to synthesize plans and the simulation trajectories for this case are shown in Fig. 5(a), where R_1 is assigned to complete the cyclic task, while R_2 and R_3 are assigned for the visiting task. Note that both R_2 and R_3 arrive at the designated area and R_1 continuously goes back and forth between the red region and its adjacent region on the right. This task is finished with two units of average cost.

In order to verify the adaptability of our method to different task, we further give a more complicate specification as follows:

$$\phi_2 = (\neg Y \mathcal{U} G) \wedge \Diamond Y \wedge \Diamond B \wedge (\neg E \mathcal{U} Y) \wedge (\neg E \mathcal{U} B) \wedge \Box\Diamond R,$$

which requires that green and yellow regions should be visited in order, blue regions should be visited eventually, gray regions should not be visited before blue and yellow regions are visited and red regions should be visited infinitely often.

The synthesized trajectories are shown in Fig. 5(b). In this simulation, R_3 is assigned to finish the sequential access task, R_2 is assigned for the visiting task and R_1 is assigned for the cyclic task by following the same trajectory as the one solved in the previous simulation. Note that, the average cost is also two units in this case, as the cyclic task does not change. Moreover, as R_2 and R_3 are going to visit blue and yellow regions, both their trajectories avoid the gray regions compared with the trajectories in Fig. 5(a) as the extra avoid requirement in ϕ_2.

In the above two simulations, note that the trajectories for R_2 and R_3 might not be optimal in length. However, this will not influence the optimal average cost of the whole team trajectories. As except for $\Box\Diamond R$, all the remaining parts in the above two formulae are just the transient parts and we only just need to find a feasible trajectory to achieve these parts. The key part we need to optimize is the trajectory after arriving the red region, which is also the steady part in the whole team trajectories for cyclic task.

5.3. Hardware experiments

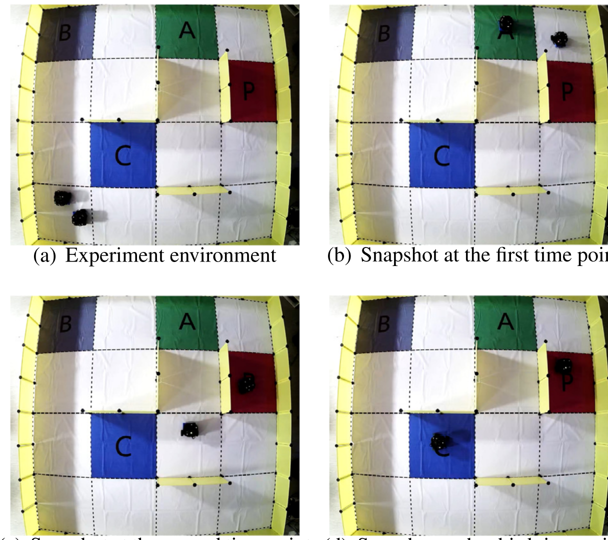
In this section, we perform a hardware experiment to verify the feasibility of our algorithm in real-world. The experiment is performed in an indoor 4×4 grid motion capture environment as shown in Fig. 6(a). We consider discrete, region-level trajectories and these trajectories can be translated into individual robot moving plans using the existing low level path planning algorithms.

We consider the following task

$$\phi = (\neg C \mathcal{U} A) \wedge \Diamond C \wedge (\neg B \mathcal{U} A) \wedge \Box\Diamond P,$$

which requires that the robots should visit regions A and C in turn, region B cannot be visited before A being visited and region P must be visited infinitely often.

The experimental video is available online.[1] Snapshots of the optimal solution for ϕ are shown in Figs. 6(b)–6(d). In Fig. 6(b), one of the robots proceeds to region A and as the avoid requirement $\neg B \mathcal{U} A$ in ϕ, it avoids region B before arriving A. At the meanwhile, another robot passes region P. Furthermore, in Fig. 6(c), the robot visiting region A continues to visit region C and the other robot comes back to visit region P. Finally, in Fig. 6(d), when the robot arrives C, it stays there forever, while another robot still continuously goes back and forth between region P and its adjacent region on the top even if its partner stops working. Therefore, ϕ has been finished by the synthesized trajectories.



(a) Experiment environment (b) Snapshot at the first time point

(c) Snapshot at the second time point (d) Snapshot at the third time point

Fig. 6. Experiment scene and snapshots of the solution. Note that in (*a*), two mobile robots with optical sensor are placed in the lower left corner.

6. 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-automaton-based approach. We showed that our method has better scalability on the number of robots and the size of environment. In our future work, we plan to extend our algorithm to deal with more general fragments of LTL specifications. Also, we plan to consider the concurrent execution of transitions in PN to deal with synchronous operations between robots.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Baier, Christel, & Katoen, Joost-Pieter (2008). *Principles of model checking*. MIT Press.

Basile, F., Chiacchio, P., & Di Marino, E. (2019). An auction-based approach to control automated warehouses using smart vehicles. *Control Engineering Practice*, *90*, 285–300.

Cai, Kai (2020). Warehouse automation by logistic robotic networks: a cyber-physical control approach. *Frontiers of Information Technology & Electronic Engineering*, *21*(5), 693–704.

Ding, Xuchu, Smith, Stephen L., Belta, Calin, & Rus, Daniela (2014). Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, *59*(5), 1244–1257.

Fainekos, Georgios E., Girard, Antoine, Kress-Gazit, Hadas, & Pappas, George J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, *45*(2), 343–352.

[1] https://github.com/Lv-Peng/Optimal-Multi-Robot-Path-Planning

Fanti, Maria Pia, Mangini, Agostino M., Pedroncelli, Giovanni, & Ukovich, Walter (2018). A decentralized control strategy for the coordination of AGV systems. *Control Engineering Practice*, *70*, 86–97.

Guo, Meng, & Dimarogonas, Dimos V. (2015). Multi-agent plan reconfiguration under local LTL specifications. *International Journal of Robotics Research*, *34*(2), 218–235.

Guo, Meng, & Zavlanos, Michael M. (2018a). Multirobot data gathering under buffer constraints and intermittent communication. *IEEE Transactions on Robotics*, *34*(4), 1082–1097.

Guo, Meng, & Zavlanos, Michael M. (2018b). Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Transactions on Automatic Control*, *63*(12), 4051–4066.

He, Zhou, Dong, Yuying, Ren, Gongchang, Gu, Chan, & Li, Zhiwu (2020). Path planning for automated guided vehicle systems with time constraints using timed Petri nets. *Measurement and Control*, *53*(9–10), 2030–2040.

He, Zhou, Zhang, Ruijie, Ran, Ning, & Gu, Chan (2022). Path planning of multi-type robot systems with time windows based on timed colored Petri nets. *Applied Sciences*, *12*(14), 6878.

Kantaros, Yiannis, & Zavlanos, Michael M. (2016). Distributed intermittent connectivity control of mobile robot networks. *IEEE Transactions on Automatic Control*, *62*(7), 3109–3121.

Kantaros, Yiannis, & Zavlanos, Michael M. (2018). Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. *IEEE Transactions on Automatic Control*, *64*(5), 1916–1931.

Kloetzer, Marius, & Mahulea, Cristian (2020). Path planning for robotic teams based on LTL specifications and Petri net models. *Discrete Event Dynamic Systems*, *30*(1), 55–79.

Kovalenko, Ilya, Tilbury, Dawn, & Barton, Kira (2019). The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, *86*, 105–117.

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

Lacerda, Bruno, & Lima, Pedro U. (2019). Petri net based multi-robot task coordination from temporal logic specifications. *Robotics and Autonomous Systems*, *122*, Article 103289.

Lahijanian, Morteza, Andersson, Sean B., & Belta, Calin (2011). Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics*, *28*(2), 396–409.

Lefebvre, D. (2018). Near-optimal scheduling for Petri net models with forbidden markings. *IEEE Transactions on Automatic Control*, *63*(8), 2550–2557.

Lefebvre, D., & Leclercq, E. (2015). Control design for trajectory tracking with untimed Petri nets. *IEEE Transactions on Automatic Control*, *60*(7), 1921–1926.

Lv, Peng, Luo, Guangqing, Yin, Xiang, Ma, Ziyue, & Li, Shaoyuan (2022). Optimal multi-robot path planning for cyclic tasks using Petri nets. In *16th IFAC international workshop on discrete event systems* (pp. 9–15).

Lv, Peng, Yin, Xiang, Ji, Yiding, & Li, Shaoyuan (2021). A game-theoretical approach for optimal supervisory control of discrete event systems for cyclic tasks. In *60th IEEE conference on decision and control* (pp. 324–330).

Ma, Ziyue, Tong, Yin, Li, Zhiwu, & Giua, Alessandro (2016). Basis marking representation of Petri net reachability spaces and its application to the reachability problem. *IEEE Transactions on Automatic Control*, *62*(3), 1078–1093.

Ma, Ziyue, Yin, Xiang, & Li, Zhiwu (2021). Marking diagnosability verification in labeled Petri nets. *Automatica*, *131*, Article 109713.

Ma, Ziyue, Zou, Minqiang, Zhang, Jiafeng, & Li, Zhiwu (2022). Design of optimal control sequences in petri nets using basis marking analysis. *IEEE Transactions on Automatic Control*.

Mahulea, Cristian, & Kloetzer, Marius (2017). Robot planning based on boolean specifications using Petri net models. *IEEE Transactions on Automatic Control*, *63*(7), 2218–2225.

Mahulea, Cristian, & Kloetzer, Marius (2018). Robot planning based on boolean specifications using Petri net models. *IEEE Transactions on Automatic Control*, *63*(7), 2218–2225.

Mahulea, Cristian, Kloetzer, Marius, & González, Ramón (2020). *Path planning of cooperative mobile robots using discrete event models*. John Wiley & Sons.

Mansouri, Sina Sharif, Kanellakis, Christoforos, Fresk, Emil, Kominiak, Dariusz, & Nikolakopoulos, George (2018). Cooperative coverage path planning for visual inspection. *Control Engineering Practice*, *74*, 118–131.

Schillinger, Philipp, Bürger, Mathias, & Dimarogonas, Dimos 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.

Seatzu, Carla, Silva, Manuel, & Van Schuppen, Jan H. (2013). *Vol. 433*, *Control of discrete-event systems*. Springer.

Sedgewick, Robert, & Wayne, Kevin (2007). *Vol. 226*, *Algorithms and data structures*. Princeton University, COS.

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

Shin, Jongho, Kwak, Dongjun, & Lee, Taehyung (2020). Robust path control for an autonomous ground vehicle in rough terrain. *Control Engineering Practice*, *98*, Article 104384.

Smith, Stephen L., Tmová, Jana, Belta, Calin, & Rus, Daniela (2011). Optimal path planning for surveillance with temporal-logic constraints. *International Journal of Robotics Research*, *30*(14), 1695–1708.

Tatsumoto, Yuta, Shiraishi, Masahiro, Cai, Kai, & Lin, Zhiyun (2018). Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. *Control Engineering Practice*, *81*, 97–104.

Tian, Daiying, Fang, Hao, Yang, Qingkai, & Wei, Yue (2021). Decentralized motion planning for multiagent collaboration under coupled LTL task specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Tong, Yin, Li, Zhiwu, Seatzu, Carla, & Giua, Alessandro (2017). Verification of state-based opacity using Petri nets. *IEEE Transactions on Automatic Control*, *62*(6), 2823–2837.

Ulusoy, Alphan, Smith, Stephen L., & Belta, Calin (2014). Optimal multi-robot path planning with LTL constraints: guaranteeing correctness through synchronization. In *Distributed autonomous robotic systems* (pp. 337–351). Springer.

Wolff, Eric M., Topcu, Ufuk, & Murray, Richard M. (2012). Optimal control with weighted average costs and temporal logic specifications. In *Proc. of robotics: science and systems*.

Yang, Shuo, Yin, Xiang, Li, Shaoyuan, & Zamani, Majid (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. In *2020 59th IEEE conference on decision and control* (pp. 4460–4466).

Yu, Pian, & Dimarogonas, Dimos V. (2021). Distributed motion coordination for multirobot systems under LTL specifications. *IEEE Transactions on Robotics*.

Yu, Xinyi, Yin, Xiang, Li, Shaoyuan, & Li, Zhaojian (2022). Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Engineering Practice*, *123*, Article 105130.