Automatica 164 (2024) 111634

Contents lists available at ScienceDirect

Automatica

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

Optimal supervisory control of discrete event systems for cyclic tasks☆

Peng Lv [a], Zhangcong Xu [a], Yiding Ji [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] Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, 511458, Hong Kong, China

ARTICLE INFO

Article history:
Received 18 September 2022
Received in revised form 8 January 2024
Accepted 14 February 2024
Available online xxxx

Keywords:
Discrete event systems
Optimal supervisory control
Mean payoff games

ABSTRACT

In this paper, we investigate the problem of optimal supervisory control for cyclic tasks in the context of discrete-event systems (DES). We consider the completion of each single task as the visit of a marked state, and overall control objective is to complete tasks cyclically in the sense that marked states are visited *infinitely often*. Following the standard optimal supervisory control framework, two types of costs, disable cost and occurrence cost, are considered. However, instead of considering the standard accumulated total cost or the average cost per event, we consider the measure for the control performance using the *average cost per task*. We show that such an optimality measure is more suitable for tasks that need to be completed cyclically. Our goal is to design a live and non-blocking supervisor such that the average cost per task in the worst-case is minimized. To solve the problem, we propose a game-theoretical approach by converting the optimal control problem as a two-player graph game. Structural properties of the converted game are discussed. In particular, we show that this game can be solved by a set of mean payoff decision problems, for which effective algorithms exist. Our problem can be considered as a special instance of the general ratio-game in the literature. However, by exploring new structural property for this problem, we achieve superior computational efficiency when compared to the conventional solution designed for more general problem formulations. Illustrative examples are provided to demonstrate the proposed algorithm.

1. Introduction

1.1. Motivations

Discrete event systems (DES) are dynamic systems with discrete state–spaces and event-driven dynamics, which are widely used in the modeling and analysis of man-made engineering cyber–physical systems such as manufacturing systems, transportation systems and communication networks (Cassandras & Lafortune, 2009). In the context of DES, the supervisory control theory (SCT) initiated by Ramadge and Wonham is a powerful formal methodology that aims to synthesize a feedback supervisor such that the closed-loop system under control satisfies some desired specifications, such as safety, liveness and non-blockingness, in the presence of uncontrollable events; see, e.g., the textbooks (Seatzu, Silva, & Van Schuppen, 2013; Wonham & Cai, 2019) and some recent works (Li & Takai, 2022; Ma & Cai, 2022; Sakakibara, Urabe, & Ushio, 2021; Takai, 2021; Yin & Lafortune, 2016a, 2016b).

One important problem in the SCT is to synthesize supervisors optimally in terms of some performance measures. This is referred to as the *optimal supervisory control problem* and has drawn considerable attentions in the literature; see, e.g., Alves, Pena, and Takahashi (2021), Fu, Ray, and Lagoa (2004), Hill and Lafortune (2016), Kumar and Garg (1995), Ma and Zhang (2020), Pena, Vilela, Alves, and Rafael (2021), Sakakibara and Ushio (2020), Sengupta and Lafortune (1998), Su, Van Schuppen, and Rooda (2011), Ware and Su (2016). Particularly, an optimal supervisory control framework was proposed in Sengupta and Lafortune (1998) by considering both the occurrence cost and the disable cost. The objective of the supervisor is to reach marked states with the smallest worst-case *accumulated total cost*. This framework has been extended subsequently to several different settings,

☆ This work was supported in part by the National Natural Science Foundation of China (62173226, 62061136004, 62303389), in part by the Guangdong Basic and Applied Basic Research Funding, China, grant 2022A151511076, in part by the Guangzhou Basic and Applied Basic Research Scheme, China, grant 2023A04J1067, and in part by the Guangzhou Municipality-University Joint Funding, China, grant 2023A03J0678. The material in this paper was partially presented at the 60th IEEE Conference on Decision and Control, December 13–15, 2021, Austin, Texas, USA. This paper was recommended for publication in revised form by Associate Editor Michel Reniers under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: lv-peng@sjtu.edu.cn (P. Lv), randomx200@sjtu.edu.cn (Z. Xu), jiyiding@ust.hk (Y. Ji), syli@sjtu.edu.cn (S. Li), yinxiang@sjtu.edu.cn (X. Yin).

https://doi.org/10.1016/j.automatica.2024.111634
0005-1098/

including, e.g., multiple goals (Marchand, Boivineau, & Lafortune, 2000), partial observations (Marchand, Boivineau, & Lafortune, 2002), online control (Grigorov & Rudie, 2006) and probabilistic systems (Pantelic & Lawford, 2011).

The original framework of Sengupta and Lafortune (1998) essentially considers finite languages, which is more suitable for a single non-repetitive task. In terms of optimal control of infinite behaviors, a common approach is to use the *average cost per event* as the optimality measure; see, e.g., Ji, Yin, and Lafortune (2021a, 2021b), Pruekprasert and Ushio (2016). However, as we will argue later in the paper, such an optimality measure is not suitable when cyclic tasks are considered, because the optimal solution may keep executing useless behaviors to minimize its cost. Still following the framework of Sengupta and Lafortune (1998), the authors in Schmidt (2015) consider the optimal supervisory control of cyclic tasks, where each task cycle is pre-specified as the reset to its initial state. This setting cannot handle the scenario where tasks are modeled by multiple marked states without a pre-specified visiting order.

1.2. Our results

In this paper, we formulate and solve a class of optimal supervisory control problem for cyclic tasks. We also follow the basic setting in Sengupta and Lafortune (1998), where uncontrollable events are taken into account, and both the occurrence costs and the control costs are considered. The tasks are modeled by a set of marked states and each completion of the task is captured by the visit of a marked state. The task is cyclic in the sense that the system needs to visit marked states infinitely often. To formulate the optimal control problem, we consider a performance measure called the *average cost per task*. We argue that such a performance measure is more suitable for infinite cyclic behaviors than the standard accumulated total cost or the average cost per event.

In order to solve the proposed optimal control problem, a game-theoretical approach is developed. Specifically, we show two important structural properties of the control problem: (i) the optimal strategy is state-based; and (ii) the optimal value belongs to a finite space. Then we show that the associated threshold decision problem can be reduced to a mean payoff value decision problem, for which effective algorithms exist in the literature. By leveraging the structural properties as well as the reduction, a (pseudo) polynomial-time algorithm is proposed for solving this type of optimal supervisory control problem.

The proposed optimal supervisory control problem has various potential applications in different engineering systems. For example, in manufacturing lines, machines need to be reconfigured to make products repeatedly, and one always wants to minimize the overall production cost for each product. Also, for example, in multi-robot surveillance, the team of robots needs to visit some checkpoints regularly in order to gather or upload data. In this case, it is meaningful to minimize the long-run average distance between each time the checkpoint is visited, which can be captured by a per task cost.

1.3. Related works

Our work is closely related to several existing works in the literature. Specifically, our solution methodology is motivated by the two-player graph games (Gradel & Thomas, 2002) and game-based methodologies have been used in Ji et al. (2021a), Pruekprasert and Ushio (2016) for solving optimal supervisory control problems. However, our per task optimality metric is different from those considered therein. The formulated optimal control problem is motivated by the standard mean payoff games (Brim, Chaloupka, Doyen, Gentilini, & Raskin, 2011; Ehrenfeucht & Mycielski, 1979). However, in mean payoff games, the cost is averaged per event not per task. Furthermore, no infinite visit requirement is imposed therein. Our problem is also related to the problems studied in Chatterjee, Henzinger, and Jurdzinski (2005), Ding, Smith, Belta, and Rus (2014). However, Chatterjee et al. (2005) study the mean-payoff parity game, where the cost is still averaged per event not per task. The optimality measure in Ding et al. (2014) is more similar to our setting. However, it considers a stochastic setting in terms of Markov decision processes for the expected cost, while our work considers a non-stochastic supervisory control problem for the worst-case cost, and therefore, the solution methodologies are completely different.

In Bloem et al. (2014), a general type of game called *ratio-game* is investigated. Instead of considering a single cost function, the ratio-game aims to minimize the ratio value of two different costs. Our investigated mean payoff Büchi game per task can be considered as a special instance of the ratio-game. Specifically, for each edge achieving the task, one can assign a unit cost to the denominator and zero otherwise. However, using existing ratio-game algorithm to solve our problem requires a cubic complexity on the number of vertices, while by leveraging structural property of the per task game, our algorithm can solve the problem with a quadratic complexity on the number of vertices. In the context of DES, ratio-games have been first applied in van der Sanden (2018) to optimize the throughput performance of supervisors, which is captured by the ratio of two different costs. However, there is no target state considered therein and the synthesized optimal supervisor may not be non-blocking.

1.4. Organization

The rest of this paper is organized as follows. Section 2 reviews some basic notions in the optimal supervisory control theory. Section 3 formulates the optimal supervisory control problem for average cost per task (OSCP-AT) that we solve in this work. Section 4 discusses how to transform the OSCP-AT into mean payoff Büchi game per task (MPBG-PT) and show the equivalence between the two problems. Our main synthesis procedure is provided in Section 5. Case studies as well as numerical experiments are provided in Section 6. Finally, we conclude the paper in Section 7. Preliminary and partial versions of some of the results in this paper are presented in Lv, Yin, Ji, and Li (2021). However, the algorithm in Lv et al. (2021) needs to enumerate the entire strategy space which leads to an exponential complexity. Here, by leveraging new structural properties of the problem, we develop a new polynomial-time synthesis algorithm, which significantly improves the result in Lv et al. (2021).

2. Preliminary on optimal supervisory control

2.1. Supervisory control theory

Let Σ be a finite set of events. A string over Σ is a finite sequence of events of form $s = \sigma_1 \dots \sigma_n, \sigma_i \in \Sigma$. We denote by Σ^* the set of all finite strings over Σ including the empty string ε. The set of all infinite strings over Σ is denoted by Σ^ω. We denote by $|s|$ the length of s and by $s_{[i]}$ the ith event in s. Also, $s_{[i,j]} = \sigma_i \dots \sigma_j$ denote the sequence from the ith event to the jth event in s. A language $L \subseteq \Sigma^*$ is a set of strings. The prefix-closure of L is defined by $\bar{L} = \{s \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } sw \in L\}$.

We consider a DES modeled as a deterministic finite-state automaton (DFA)

$$G = (Q, \Sigma, \delta, q_0, Q_m),$$

where Q is a finite set of states, Σ is a finite set of events, $\delta : Q \times \Sigma \to Q$ is a partial transition function, $q_0 \in Q$ is the initial state and $Q_m \subseteq Q$ is a set of marked states. The transition function is also extended to $\delta : Q \times \Sigma^* \to Q$ in the usual manner (Cassandras & Lafortune, 2009). The language *generated* by G is $\mathcal{L}(G) = \{s \in \Sigma^* : \delta(q_0, s)!\}$, where "!" means "is defined". We also denote by $\mathcal{L}^\omega(G)$ the set of infinite strings generated by G. The language *marked* by G is $\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) : \delta(q_0, s) \in Q_m\}$. Marked states are usually used to model the goal/task of a system. For any $q \in Q$, we define $\Delta_G(q) = \{\sigma \in \Sigma : \delta(q, \sigma)!\}$ as the set of active events at q; we also define $\Delta_G(s) = \Delta_G(\delta(q_0, s))$. For simplicity, we write $\delta(q, s)$ as $\delta(s)$ when $q = q_0$. For technical reason that will be clear later, we assume that the transition from one state to the other is unique, i.e., for any $q, q' \in Q$, we have $|\{\sigma \in \Sigma : \delta(q, \sigma) = q'\}| = 1$, and we denote by $\sigma_{q,q'}$ the unique event from q to q'. Note that, this assumption is without loss of generality since we can always refine the state space such that the assumption holds.

In the supervisory control theory, the event set is partitioned as

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc},$$

where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. Then a *supervisor* is a mapping

$$S : \mathcal{L}(G) \to \Gamma$$

that enables events dynamically based on the executed string, where $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ is the set of control patterns. The language generated by the closed-loop system under control, denoted by $\mathcal{L}(S/G)$, is defined recursively by

- $\varepsilon \in \mathcal{L}(S/G)$;
- For any $s \in \Sigma^*$ and $\sigma \in \Sigma$, we have $s\sigma \in \mathcal{L}(S/G)$ iff $s \in \mathcal{L}(S/G)$, $s\sigma \in \mathcal{L}(G)$ and $\sigma \in S(s)$.

The language marked by S/G is defined by $\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$. The infinite language generated by the closed-loop is denoted by $\mathcal{L}^\omega(S/G)$ such that, for any $s \in \Sigma^\omega$, we have $s \in \mathcal{L}^\omega(S/G)$ iff $\forall i \geq 1 : s_{[1,i]} \in \mathcal{L}(S/G)$.

A supervisor S is said to be:

- *live*: if $\forall s \in \mathcal{L}(S/G), \exists \sigma \in \Sigma : s\sigma \in \mathcal{L}(S/G)$;
- *non-blocking*: if $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$;
- *state-based*: if the decisions are only based on the current state, i.e., $\forall s, t \in \mathcal{L}(G) : \delta(s) = \delta(t) \Rightarrow S(s) = S(t)$.

Note that liveness and non-blockingness are incomparable, and in this work, we require the synthesized supervisor satisfying both properties.

2.2. Cost functions

Following the standard framework of optimal supervisory control (Sengupta & Lafortune, 1998), we consider the following two types of costs:

- occurrence cost: $c_e : \Sigma \to \mathbb{N}$; and
- disable cost: $c_d : \Sigma_c \to \mathbb{N}$.

That is, for each $\sigma \in \Sigma$, $c_e(\sigma)$ denotes the cost incurred when σ is executed. The occurrence cost can model, for example, the energy consumption for each event execution. On the other hand, the disable cost $c_d(\sigma)$ describes the cost incurred when the supervisor tries to prevent a feasible and controllable event σ from happening. The disable cost also provides a quantitative measure for the permissiveness of the supervisor since the supervisor will incur large disable cost if it interferes the behavior of the system too much. Therefore, the disable cost incurred using control decision $\gamma \in \Gamma$ at state $q \in Q$, denoted by $c'_d(\gamma, q)$, is defined by

$$c'_d(\gamma, q) = \sum_{\sigma \in (\Delta_G(q) \cap \Sigma_c) \setminus \gamma} c_d(\sigma),$$

which is the summation of disable costs for all feasible but disabled events. With a slight abuse of notion and for the sake of simplicity, hereafter, we still use $c_d(\gamma, q)$ to denote $c'_d(\gamma, q)$.

Then given a supervisor, and for any finite string $s = \sigma_1 \ldots \sigma_n \in \mathcal{L}(S/G)$, the *total cost* incurred along s is defined by

$$\mathsf{Cost}_S(s) = \sum_{i=1,\ldots,n} c_e(\sigma_i) + \sum_{s' \in \overline{\{s\}}} c_d(S(s'), \delta(s')),$$

where the first component represents the total occurrence cost for all events in string s and the second component represents the total disable cost for all decisions along s.

However, for an infinite string $s \in \mathcal{L}^\omega(S/G)$, it does not make sense to talk about its total cost as it usually goes to infinity. An alternative optimality metric is to consider the *average cost* (or, the mean payoff), which is defined by

$$\mathsf{Cost}_S^{Ave}(s) = \limsup_{n \to \infty} \left\{ \frac{1}{n} \mathsf{Cost}_S(s_{[1,n]}) \right\},$$

where $s_{[1,n]}$ is the prefix of s with length n.

In the DES literature, different types of optimal control problems have been investigated, including, e.g.,

(1) *Total Cost Control for Reachability* (Sengupta & Lafortune, 1998): This problem requires to reach marked states optimally; hence, the supervisor needs to be *non-blocking*. Furthermore, the supervisor needs to minimize the worst-case total cost $\mathsf{Cost}_S(s)$ for string s that reaches the marked states Q_m *for the first-time*. Note that here it is meaningful to discuss the total cost since once a marked state is reached, the entire task is completed, i.e., the optimal solution should be in finite horizon.

(2) *Average Cost Control for Liveness*(Ji et al., 2021b; Pruekprasert & Ushio, 2016): This problem requires to find a *live* supervisor such that the system can execute indefinitely. Since the horizon is infinite, it makes sense to consider the average cost and the supervisor needs to minimize $\mathsf{Cost}_S^{Ave}(s)$ for the worst-case.

The first problem is useful to describe the scenario, where a *single task*, modeled by marked states, needs to be achieved optimally and for only once. The second problem is useful to describe the scenario, where the supervisor needs to ensure the non-termination of the entire process and to minimize the average cost during the indefinite process.

3. Optimal control for cyclic tasks

3.1. Motivating example

In the optimal control problem for cyclic tasks, the supervisor wants to make sure that marked states can be visited *infinitely often* so that tasks can be completed repeatedly. Although such a solution involves infinite strings, the following simple example shows that the standard average cost is not a suitable performance metric for optimality.

Example 1. Let us consider system G shown in Fig. 1, where all events are controllable and double circle denotes a marked state. For each state, we assume that there is no disable cost and the occurrence cost is given as the number associated to each transition. If one wants to visit marked state q_0 infinitely often, while minimizing the average cost, a possible optimal solution is as follows:

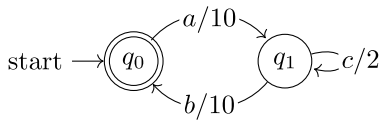


Fig. 1. System G in Example 1.

- upon the kth occurrence of event a, the supervisor repeats control decision $\{c\}$ for k-times and then changes to control decision $\{b\}$.

This is because that c has a lower cost and enabling c few times can help to reduce the cost averaged by events, i.e., $\mathsf{Cost}_S^{Ave}(s)$ goes to 2. Furthermore, marked state q_0 is still visited infinitely often, which satisfies the cyclicity requirement.

However, this optimal solution is not of practical interest, because the optimal average cost is achieved by increasing the percentage of event c, which is useless for completing the task. A simple and practical solution is to alternate between control decisions $\{a\}$ and $\{b\}$ without allowing event c for even once. This simple example suggests that, when tasks modeled by marked states are considered, it makes more sense to average the total cost by the number of completions of tasks, rather than the number of event occurrences.

3.2. Problem formulation

Motivated by the above discussions, for each string $s \in \mathcal{L}(G)$, we denote by $I_m(s)$ the number of visits of marked states Q_m along s, i.e.,

$$I_m(s) = |\{s' \in \overline{\{s\}} : \delta(q_0, s') \in Q_m\}|.$$

Essentially, $I_m(s)$ represents the number of tasks the system has completed. Let $s \in \mathcal{L}^\omega(S/G)$ be an infinite string generated by the system. Then the *average cost per task* of s under supervisor S is

$$\mathsf{Cost}_S^{AveT}(s) = \limsup_{n\to\infty} \left\{ \frac{1}{I_m(s_{[1,n]})} \mathsf{Cost}_S(s_{[1,n]}) \right\}. \tag{1}$$

This leads to the *Optimal Supervisory Control Problem for Average Cost Per Task* (OSCP-AT) that we solve in this paper.

Problem 1 (*OSCP-AT*)**.** Given system G with Σ_c, find an optimal supervisor S^* such that

(1) S^* is live and non-blocking; and
(2) for any $s \in \mathcal{L}^\omega(S^*/G)$, $\mathsf{Cost}_{S^*}^{AveT}(s) < \infty$; and
(3) for any S' satisfying (1) and (2), we have

$$\sup_{s\in\mathcal{L}^\omega(S^*/G)} \mathsf{Cost}_S^{*AveT}(s) \le \sup_{s\in\mathcal{L}^\omega(S'/G)} \mathsf{Cost}_{S'}^{AveT}(s).$$

Remark 1. It is important to note that, the average cost per task considered in our problem can be treated as a special instance of the ratio value cost originally studied in Bloem et al. (2014). Specifically, for any finite string $s_{[1,n]}$, we consider two different costs $c_1(s_{[1,n]})$ and $c_2(s_{[1,n]})$. Then for any infinite string s, its ratio value cost is defined as $\mathsf{Ratio}^{(c_1,c_2)}(s) = \limsup_{n\to\infty} \frac{c_1(s_{[1,n]})}{c_2(s_{[1,n]})}$. Clearly, by setting $c_1 = \mathsf{Cost}_S$ and $c_2 = I_m$, this ratio value cost reduces to our average cost per task. In this work, instead of solving our problem directly using the general algorithm in Bloem et al. (2014), we will explore new structural properties for the average cost per task measure, which yield a more efficient and customized algorithm.

4. Game-based formulation of SCT

Pruekprasert, Ushio, and Kanazawa (2015), van der Sanden (2018), a graph-game-based approach was developed for solving an optimal supervisory control problem for average cost per event. In this paper, to solve the optimal supervisory control problem for average cost per task, we also convert the DES model as a two-player game over a weighted graph.

4.1. Two-player graph games

A two-player *game graph* is a bipartite graph

$$\mathcal{A} = (V = V_0 \dot\cup V_1, E, v_0),$$

where V is a set of vertices and V_0 and V_1 form a partition of V denoting, respectively, the set of vertices of *Player 0* and *Player 1*; $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ is a set of edges; and $v_0 \in V_0$ is the initial vertex of the game. A *play* in $\mathcal{A}$ is an infinite sequence $\rho \in V^\omega$ such that $\langle \rho_{[i]}, \rho_{[i+1]} \rangle \in E, \forall i \ge 0$. The set of all plays starting from v is denoted by $\mathsf{Plays}(\mathcal{A}, v)$. For any play $\rho = v_0 v_1 \cdots v_n \in \mathsf{Plays}(\mathcal{A}, v)$, ρ is called a *cycle* if $v_0 = v_n$. For a cycle, if $\forall 0 < i < j < n, v_i \ne v_j, v_i \ne v_0$ and $v_j \ne v_0$, we call it a *simple cycle*; otherwise, it is a *compound cycle*.

A *strategy* for Player $i \in \{0, 1\}$ is a function $\theta_i : V^* V_i \to V$ such that $\forall w \in V^*, v \in V_i : \theta_i(wv) = v' \Rightarrow \langle v, v' \rangle \in E$, where V^* is the set of all finite sequences over V. We denote by $\tilde\Theta_i$ the set of all strategies for Player $i \in \{0, 1\}$. Then given a strategy θ_i of Player $i \in \{0, 1\}$, we say a play $\rho \in \mathsf{Play}(\mathcal{A}, v)$ from $v \in V$ is *consistent* with strategy θ_i if $\forall n \ge 0 : \rho_{[n]} \in V_i \Rightarrow \rho_{[n+1]} = \theta_i(\rho_{[1,n]})$. We denote by $\mathsf{Play}(\mathcal{A}, v, \theta_i)$ the set of all plays consistent with θ_i from $v \in V$. If the strategies of both players are given, i.e., $\boldsymbol{\theta} = (\theta_0, \theta_1)$, then the play from $v \in V$ can be uniquely determined, which is $\rho(v, \boldsymbol{\theta}) := \cap_{i=0,1} \mathsf{Play}(\mathcal{A}, v, \theta_i)$.

The goal of each player is to achieve some *objective*. Most game objectives investigated in the literature can be categorized as qualitative objectives or quantitative objectives.

(1) Qualitative Objective: A qualitative objective can be expressed as a *winning condition* $\mathsf{Win} \subseteq V^\omega$, which is a set of infinite sequences. Specifically, we say that strategy θ_i achieves Win for Player i, if $\mathsf{Play}(\mathcal{A}, v_0, \theta_i) \subseteq \mathsf{Win}$. In this paper, we focus on finding winning strategies for Player 0; hence Win is always considered for Player 0. Given a set of vertices $V_m \subseteq V$, one can define different types of winning conditions, e.g.,

- safety: $\mathsf{Win}_S(V_m) = \{\rho \in V^\omega : \mathsf{Occ}(\rho) \cap V_m = \emptyset\}$;
- reachability: $\mathsf{Win}_R(V_m) = \{\rho \in V^\omega : \mathsf{Occ}(\rho) \cap V_m \ne \emptyset\}$;
- Büchi: $\mathsf{Win}_B(V_m) = \{\rho \in V^\omega : \mathsf{Inf}(\rho) \cap V_m \ne \emptyset\}$,

where $\mathsf{Occ}(\rho)$ and $\mathsf{Inf}(\rho)$ denote, respectively, the set of vertices that occur at least once and infinite number of times in ρ. Game graphs associated with winning conditions $\mathsf{Win}_S(V_m)$, $\mathsf{Win}_R(V_m)$ and $\mathsf{Win}_B(V_m)$ are referred to as the *safety game*, *reachability game* and *Büchi game*, respectively. Effective algorithms have been proposed for solving each of the above games; see, e.g., Gradel and Thomas (2002).

(2) Quantitative Objective: Quantitative objectives are investigated for a weighted game $(\mathcal{A}, w)$, where $\mathcal{A}$ is an arena and $w : E \to \mathbb{N}^+$ is a weight function assigning each edge a weight (or payoff). Later in this work, we will leverage an important type of quantitative game called the *mean payoff game* to solve our problem. In the mean payoff game, Player 0 aims to minimize the mean payoff of a play $\rho \in V^\omega$, i.e.,

$$\mathsf{MP}^{(w)}(\rho) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i<n} w(\rho_{[i]}, \rho_{[i+1]}).$$

The *value secured* by strategy θ_0 of Player 0 at vertex $v \in V$ is

$$\mathsf{val}^{(w)}_{\mathsf{MP}}(v;\theta_0) = \sup_{\theta_1\in\tilde{\Theta}_1} \mathsf{MP}^{(w)}(\rho(v,\theta_0,\theta_1)),$$

which is the worst-case payoff it can ensure. The optimal value for Player 0 at vertex $v \in V$ in the game $(\mathcal{A},\omega)$ is $\mathsf{val}^{(w)}_{\mathsf{MP}}(v) = \inf_{\theta_0\in\tilde{\Theta}_0} \mathsf{val}^{(w)}_{\mathsf{MP}}(v;\theta_0)$. It was shown by Zwick and Paterson (1996) that Player 0 has an optimal strategy θ^*_0 to secure this optimal value; furthermore this strategy is positional, i.e., it only depends on the current vertex. Note that, here we use superscript (w) to emphasize that the payoff is w.r.t. weight function w. We will also omit the superscript (w) when there is no ambiguity.

4.2. Supervisory control as a game

As we mentioned earlier, our approach is to transform the supervisory control problem as a game. Specifically, given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, we construct a new game arena

$$\mathcal{A}^G = (V^G = V^G_0 \cup V^G_1, E^G, v^G_0)$$

where

- $V^G_0 \subseteq Q \cup \{v_{D,0}\}$ are Player 0's vertices;
- $V^G_1 \subseteq (Q \times \Gamma) \cup \{v_{D,1}\}$ are Player 1's vertices;
- $E^G \subseteq (V^G_0 \times V^G_1) \cup (V^G_1 \times V^G_0)$ are edges defined by:
  - for any $q \in Q \subseteq V^G_0$, $\gamma \in \Gamma$, we have $\langle q,(q,\gamma)\rangle \in E^G$
  - for any $(q,\gamma) \in V^G_1$, if $\Delta_G(q) \cap \gamma \neq \emptyset$, then for each $\sigma \in \Delta_G(q) \cap \gamma$, we have

    $\langle (q,\gamma), \delta(q,\sigma)\rangle \in E^G$

    otherwise, when $\Delta_G(q) \cap \gamma = \emptyset$, we have

    $\langle (q,\gamma), v_{D,0}\rangle \in E^G$
  - for $v_{D,0} \in V^G_0$ and $v_{D,1} \in V^G_1$, we have

    $\langle v_{D,0}, v_{D,1}\rangle, \langle v_{D,1}, v_{D,0}\rangle \in E^G$
- $v^G_0 = q_0$ is the initial vertex.

Intuitively, game graph $\mathcal{A}^G$ explicitly distinguishes between the supervisor's decision stage V^G_0 from which a control pattern is chosen and the environment's decision stage V^G_1 from which an event occurs. Furthermore, each state in V^G_1 is of form (q,γ), where q is the current state, while $\gamma \in \Gamma$ represents the current control pattern applied. Note that (q,γ) can move to some state $q' \in V^G_0$ only when there exists some feasible events enabled at q under γ. For the case that $\Delta_G(q) \cap \gamma = \emptyset$, we introduce two new vertices $v_{D,0} \in V^G_0$ and $v_{D,1} \in V^G_1$, where "D" represents "deadlock". Therefore, the graph has at least one outgoing edge for each vertex, but may loop in the deadlock vertices forever. We will refer to $\mathcal{A}^G$ as the *supervisory control (SC) game graph* and for the sake of simplicity, hereafter we will omit all superscripts G in $\mathcal{A}^G$ and just write it as $\mathcal{A}$.

To describe the qualitative winning condition, we define $V_m := Q_m \subseteq V_0$ as the set of "target" vertices that should be visited infinitely often, i.e., we consider Büchi winning condition w.r.t. V_m. Furthermore, to introduce the quantitative objective, we define a weight function $w : E \to \mathbb{N} \cup \{\infty\}$ by:

- $w((q,\gamma),q') = c_e(\sigma_{q,q'})$;
- $w(q,(q,\gamma)) = c_d(\gamma,q)$;
- $w((q,\gamma),v_{D,0}) = w(v_{D,0},v_{D,1}) = w(v_{D,1},v_{D,0}) = \infty$.

The above construction is sufficient to capture all information needed in the synthesis problem since we have assumed that there is at most one event from a state to another. If this assumption does not hold, then function w cannot capture the occurrence costs correctly since $\sigma_{q,q'}$ is not unique. In this case, we need to pre-process plant G such that the assumption holds.

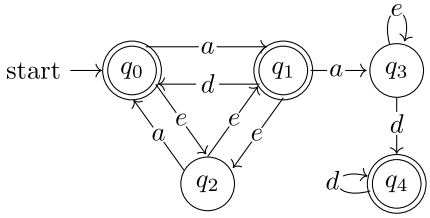


Fig. 2. DES G for Example 2.

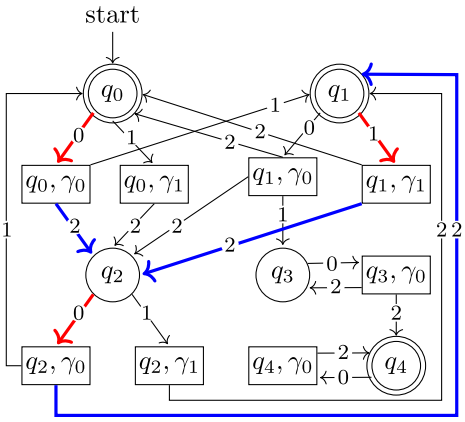


Fig. 3. The SC game graph $\mathcal{A}$, where $\gamma_0 = \{a,d,e\}$ and $\gamma_1 = \{d,e\}$.

Example 2. Consider system G shown in Fig. 2, where $\Sigma_c = \{a\}$ and $Q_m = \{q_0,q_1,q_4\}$. Suppose the occurrence costs of events are given by $c_e(a) = 1$ and $c_e(d) = c_e(e) = 2$ and the disable cost for $a \in \Sigma_c$ is given by $c_d(a) = 1$. Then, the corresponding SC game graph $\mathcal{A}$ is shown in Fig. 3, where we use circles and squares to denote Player 0's vertices and Player 1's vertices respectively and the numbers on the edges are the cost values. For the sake of simplicity, at each state, we only consider control patterns in which all disabled events are defined at this state. That is, those undefined events are always included in the control pattern. For example, in state q_3, event a is not feasible; therefore, it suffices to only consider control decision $\gamma_0 = \{a,d,e\}$, and $\gamma_1 = \{d,e\}$ is redundant.

To capture the average cost per task requirement as formulated in Problem 1, similarly to the mean payoff game, for any finite sequence ρ, we denote by $N_m(\rho)$ as the number of occurrences of Q_m in ρ. Then the *mean payoff per task* of a play $\rho \in V^\omega$ is

$$\mathsf{PT}^{(w)}(\rho) = \limsup_{n\to\infty} \frac{\mathsf{Cost}(\rho_{[0,n]})}{N_m(\rho_{[0,n]})}, \tag{2}$$

where $\mathsf{Cost}(\rho_{[0,n]}) = \sum_{i<n} w(\rho_{[i]},\rho_{[i+1]})$. Then the *mean payoff per task value* secured by strategy θ_0 of Player 0 at vertex $v \in V$ w.r.t. weight function w is defined by

$$\mathsf{val}^{(w)}_{\mathsf{PT}}(v;\theta_0) = \sup_{\theta_1\in\tilde{\Theta}_1} \mathsf{PT}^{(w)}(\rho(v,\theta_0,\theta_1)).$$

Similarly, we define the optimal mean payoff per task value for Player 0 at vertex $v \in V$ in the game $(\mathcal{A},\omega)$ as $\mathsf{val}^{(w)}_{\mathsf{PT}}(v) = \inf_{\theta_0\in\tilde{\Theta}_0} \mathsf{val}^{(w)}_{\mathsf{PT}}(v;\theta_0)$.

Example 3. We still consider the SC game in Fig. 3 and consider positional strategies θ_0 and θ_1 showed by red and blue lines, respectively. Then we have $\rho(q_0,\theta_0,\theta_1) = \rho_0\rho_1^\omega$ with $\rho_0 = q_0(q_0,\gamma_0)$ and $\rho_1 = q_2(q_2,\gamma_0)q_1(q_1,\gamma_1)$. Since there exists only one target vertex q_1 in ρ_2 and the total cost of ρ_2 is 5, we have $\mathsf{val}_{\mathsf{PT}}^{(w)}(q_0) = \mathsf{val}_{\mathsf{PT}}^{(w)}(q_0;\theta_0) = \mathsf{PT}^{(w)}(\rho(q_0,\theta_0,\theta_1)) = 5$.

We consider a game in which Player 0 aims to minimize the per task payoff. This leads to the formulation of the following problem of *Mean Payoff Büchi Game Per Task* (MPBG-PT).

Problem 2 (*MPBG-PT*)**.** Given a game graph $\mathcal{A}$, target vertices Q_m and weight function $w: V \to \mathbb{N}\cup\{\infty\}$, find an optimal strategy $\theta_0^* \in \tilde{\Theta}_0$ for Player 0 such that

(1) $\mathsf{Play}(\mathcal{A}, v_0, \theta_0^*) \subseteq \mathsf{Win}_B(V_m)$;
(2) for any strategy $\theta_0' \in \tilde{\Theta}_0$ satisfying (1), we have $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0;\theta_0^*) \le \mathsf{val}_{\mathsf{PT}}^{(w)}(v_0;\theta_0')$.

Note that, if strategy θ_0^* ensures infinite visits of target vertices, then its per-task value has to be finite, and vice versa. Therefore, in the above problem formulation, the satisfaction of the Büchi condition is equivalent to the boundedness of $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0;\theta_0^*)$.

4.3. Correctness of the transformation

Note that game graph $\mathcal{A}$ is constructed from G. Therefore, a Player 0's strategy in $\mathcal{A}$ and a supervisor for G can be mapped from one to the other as follows:

- Given strategy θ_0, it induces a supervisor, denoted by S_{θ_0}, inductively by: for any play $\rho = q_0(q_0,\gamma_0)q_1(q_1,\gamma_1)\dots q_n(q_n,\gamma_n) \in \mathsf{Play}(\mathcal{A},q_0,\theta_0)$, we have $S_{\theta_0}(\sigma_0\sigma_1\dots\sigma_n) = \gamma_n$, where $\sigma_0 = \epsilon$ and $\delta(q_i,\sigma_{i+1}) = q_{i+1}, \forall i = 0,1,\dots,n-1$.
- Given supervisor S, it also induces a strategy for Player 0, denoted by θ_S, inductively by: for any $s = \sigma_1\cdots\sigma_n \in \mathcal{L}(S/G)$, we have $\theta_0(\rho) = (q_n, S(s))$, where ρ is the unique play of form $\rho = q_0(q_0,\gamma_0)q_1(q_1,\gamma_1)\dots q_n \in \mathsf{Play}(\mathcal{A},q_0,\theta_S)$, and $\delta(q_i,\sigma_{i+1}) = q_{i+1}, \forall i = 0,1,\dots,n-1$.

We note that Problem 2 requires the Büchi winning condition, which seems to be stronger than the liveness and non-blockingness requirements in the original problem, because non-blockingness only requires the existence of path to marked states. However, these two conditions are essentially equivalent under the assumption that each event has a non-zero occurrence cost and the requirement that $\mathsf{Cost}_{S^*}^{AveT}(s) < \infty$. This is because, if the system is live and non-blocking but cannot visit marked states infinitely often, then it must loop somewhere which yields infinite cost per task. For example, for system G in Fig. 1, the system itself is already live and non-blocking, but its cost per task is infinite due to the cycle at state q_1. Therefore, it does not satisfy the Büchi winning condition.

The following theorem shows that, to solve the original OSCP-AT as formulated in Problem 1, it is equivalent to solve the game as defined in Problem 2. Therefore, our later developments can only focus on the game-based formulation.

1. *A strategy θ_0^* solves Problem 2 if and only if its induced supervisor $S_{\theta_0^*}$ solves Problem 1.*

Proof. See the Appendix.

Hereafter, we assume without loss of generality that, for the construction game $\mathcal{A}$, there exists a strategy of Player 0 θ_0 such that $\mathsf{Play}(\mathcal{A},v_0,\theta_0) \subseteq \mathsf{Win}_B(V_m)$. The existence of such a strategy can be determined by checking the non-emptyness of the Büchi winning region in polynomial-time; see, e.g., Gradel and Thomas (2002). Otherwise, we can conclude immediately that Problem 1 has no solution.

5. Optimal supervisor synthesis procedure

In this section, we present our main synthesis procedure for solving Problem 2 (and hence solves Problem 1). Our approach consists of the following steps:

- First, we show two key structural properties of MPBG-PT: (i) the optimal strategy is positional; and (ii) the value of the game is within a finite set;
- Then, we further consider a threshold decision problem for the mean payoff per task value, and show that this decision problem can be reduced to the threshold decision problem for the standard mean payoff value;
- Finally, by leveraging the structural properties of the game and by iteratively applying the threshold decision problems, we can effectively solve MPBG-PT.

5.1. Structure property of MPBG-PT

In this subsection, we first show two key structural properties of MPBG-PT. First, we show that the value of MPBG-PT can be achieved by a *positional strategy*.

In the literature of two-player graph games, it is well-known that (e.g., Corollary 7 in Gimbert and Zielonka (2005)), given a payoff function, the optimal value of the game can be achieved by a pair of *positional strategies* if each player has a positional optimal strategy for the same payoff function by assuming that it can control all vertices in the graph. In many cases, the later condition is much easier to test. The reader is referred to Gimbert and Zielonka (2005) for a more rigorous statement of this result. Here, by applying this result to our MPBG-PT, we have the following proposition.

Proposition 1. *If Problem 2 has a solution, then there exists a pair of positional strategies (θ_0^*,θ_1^*) such that θ_0^* solves Problem 2 and $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0) = \mathsf{PT}^{(w)}(\rho^*(v_0,\theta_0^*,\theta_1^*))$.*

Proof. For the original weighted game $(\mathcal{A} = (V = V_0\cup V_1, E, v_0), w)$, we assume that Player 0's vertices are $V_0\cup V_1$ and there is no Player 1's vertex and consider (2) as the payoff function for Player 0. For any pair of positional strategies (θ_0,θ_1), $\rho(v_0,\theta_0,\theta_1) \in \mathsf{Win}_B(V_m)$ must be a play of the *prefix–suffix form* $\rho(v_0,\theta_0,\theta_1) = \rho_{pre}(\rho_{sur})^\omega$, where $\rho_{pre} = v_0\cdots v_n$ is a finite path from v_0 to ρ_{sur} and $\rho_{sur} = v_{n+1}\cdots v_m$ is a cycle. Furthermore, we know that

$$\begin{aligned}\mathsf{PT}^{(w)}(\rho) &= \limsup_{i\to\infty}\frac{\mathsf{Cost}(\rho_{[0,i]})}{N_m(\rho_{[0,i]})}\\ &= \limsup_{k\to\infty}\frac{\mathsf{Cost}(\rho_{pre})+k\cdot\mathsf{Cost}(\rho_{sur})}{N_m(\rho_{pre})+k\cdot N_m(\rho_{sur})}.\end{aligned}$$

As ρ_{pre} is a finite sequence, we have

$$\begin{aligned}\mathsf{PT}^{(w)}(\rho) &= \limsup_{k\to\infty}\frac{\mathsf{Cost}(\rho_{pre})+k\cdot\mathsf{Cost}(\rho_{sur})}{N_m(\rho_{pre})+k\cdot N_m(\rho_{sur})}\\ &= \limsup_{k\to\infty}\frac{k\cdot\mathsf{Cost}(\rho_{sur})}{k\cdot N_m(\rho_{sur})}\\ &= \frac{\mathsf{Cost}(\rho_{sur})}{N_m(\rho_{sur})},\end{aligned}$$

which means that $\mathsf{PT}^{(w)}(\rho)$ is only determined by the mean payoff value per task of ρ_{sur}. Also, given two cycles ρ_1 and ρ_2 with $\mathsf{Cost}(\rho_1) = n_1, \mathsf{Cost}(\rho_2) = n_2, N_m(\rho_1) = d_1 \neq 0$ and $N_m(\rho_2) = d_2 \neq 0$, suppose $\frac{n_1}{d_1} < \frac{n_2}{d_2}$. Clearly, we have $\frac{n_1}{d_1} < \frac{n_1+n_2}{d_1+d_2}$. Therefore, let ρ_{sur}^* be the simple cycle with the minimum value with respect to (2) that is reachable from v_0. Then there must exist a positional strategy for Player 0 to achieve the following

path $\rho^* = \rho^*_{pre}(\rho^*_{sur})^\omega$, where ρ^*_{pre} is a finite path from v_0 to ρ^*_{sur}. Therefore, Player 0 has an optimal positional strategy in any weighted game $(\mathcal{A}, \omega)$ with $V_1 = \emptyset$ with respect to (2). Similarly, we can also argue that Player 1 has an optimal positional strategy in any weighted game $(\mathcal{A}, \omega)$ with $V_0 = \emptyset$ with respect to (2) by letting ρ^*_{sur} be the simple cycle with the maximum value with respect to (2) that is reachable from v_0. Since both Players have a positional optimal strategy when assuming it can control all vertices, the proposition is proved according to Corollary 7 of Gimbert and Zielonka (2005). ■

Based on the above result, hereafter, we will only consider positional strategies for Player 0 to solve Problem 2 and denote by Θ_i the set of all positional strategies for Player $i \in \{0, 1\}$. In fact, the following result further shows that, for any optimal positional $\theta_0 \in \Theta_0$ of Player 0, the worst-case value can be achieved also by a positional strategy $\theta_1 \in \Theta_1$ of Player 1 (either for the mean payoff value or the MPBG-PT value).

Lemma 1. *Given an optimal positional strategy $\theta_0 \in \Theta_0$ such that $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$, we have*

$$\mathsf{val}^{(w)}_{\mathsf{PT}}(v; \theta_0) = \sup_{\theta_1 \in \Theta_1} \mathsf{PT}^{(w)}(\rho(v, \theta_0, \theta_1)) \tag{3}$$

$$\mathsf{val}^{(w)}_{\mathsf{MP}}(v; \theta_0) = \sup_{\theta_1 \in \Theta_1} \mathsf{MP}^{(w)}(\rho(v, \theta_0, \theta_1)). \tag{4}$$

Proof. We first prove Eq. (3). Let $p = \sup_{\theta_1 \in \Theta_1} \mathsf{PT}^{(w)}(\rho(v, \theta_0, \theta_1))$. Note that since θ_0 is a positional strategy, then for any positional strategy $\theta \in \Theta_1$, $\rho(v_0, \theta_0, \theta)$ must be a play of prefix–suffix form such that $\rho(v_0, \theta_0, \theta) = \rho_{pre}(\rho_{sur})^\omega$ with ρ_{sur} being a simple cycle. Now we assume Eq. (3) does not hold, which means that $\exists \theta' \in \tilde{\Theta}_1, \mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta')) > p$. As $\mathcal{A}$ is a finite graph with a finite vertex set V, there must exist at least one vertex $v \in V$ occurring infinite times in $\rho(v_0, \theta_0, \theta')$. Then, we can divide $\rho(v_0, \theta_0, \theta')$ into infinite number of cycles according to v, which may be either simple cycles or compound cycles, and a finite play from v_0 to these cycles. If $\mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta')) > p$, then there must exist infinite number of cycles in $\rho(v_0, \theta_0, \theta')$ with the mean payoff values per task being greater than p, which leads to a contradiction. Therefore, Eq. (3) holds.

Regarding Eq. (4), similarly, we can still choose $p = \sup_{\theta_1 \in \Theta_1} \mathsf{MP}^{(w)}(\rho(v, \theta_0, \theta_1))$. Note that $\rho(v_0, \theta_0, \theta)$ is also a play of prefix–suffix form. Therefore, following the same contradictive argument, we know that there exists no $\theta' \in \tilde{\Theta}_1$ such that $\mathsf{MP}^{(w)}(\rho(v_0, \theta_0, \theta')) > p$. ■

With the help of the above lemma, now we are ready to show the second key structural property of MPBG-PT. That is, the value of MPBG-PT $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$ belongs to a finite set of rational numbers.

2. *Let $(\mathcal{A}, w)$ be a weighted game, where $\mathcal{A} = (V = V_0 \cup V_1, E, v_0)$, $W = \max_{e \in E} w(e)$ and $W' = \min_{e \in E} w(e)$ be the maximum and minimum weights in $\mathcal{A}$, respectively. Then the optimal mean payoff per task value for Player 0, i.e., $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$, is a rational number $\frac{n}{d}$, where $n, d \in \mathbb{N}$, such that $1 \le d \le |V_m|$ and $2d \cdot W' \le n \le |V| \cdot W$.*

Proof. By Proposition 1 and Lemma 1, we know that $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$ can be achieved by a pair of positional strategies (θ_0, θ_1), which yields a unique play in the prefix–suffix form $\rho(v_0, \theta_0, \theta_1) = \rho_{pre}(\rho_{sur})^\omega$. As we have argued before, the per task value of the play $\mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta_1))$ is only determined by the suffix part ρ_{sur} and we have $\mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta_1)) = \frac{\mathsf{Cost}(\rho_{sur})}{N_m(\rho_{sur})}$. Therefore, $\mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta_1))$ is a fractional number, where the numerator representing the total cost value incurred along ρ_{sur} and the denominator representing the number of occurrences of V_m in ρ_{sur}. Since each edge is assigned with a non-negative integer weight, $\mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta_1))$ must be a rational number, which means that the optimal mean payoff value per task $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$ is also a rational number. Furthermore, for any simple cycle ρ_{sur} in $\mathcal{A}$, the number of vertices in V_m that ρ_{sur} passes through can only be an integer between 1 and V_m, and therefore we have $1 \le d \le |V_m|$.

Then for the numerator $\mathsf{Cost}(\rho_{sur})$, since $\mathcal{A}$ is a bipartite graph and V_m belongs to V_0 only, for a given d, the minimum total cost is $2d \cdot W'$. On the other hand, since $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$ can be achieved by two optimal positional strategies (θ_0, θ_1), the maximal edge number in ρ_{sur} is $|V|$. Therefore, we have $\mathsf{Cost}(\rho_{sur}) \le |V| \cdot W$. This proves the theorem. ■

Based on the above theorem, we define

$$\Xi = \left\{ \frac{n}{d} : \begin{array}{c} n, d \in \mathbb{N}, 1 \le d \le |V_m|, \\ 2d \cdot W' \le n \le |V| \cdot W \end{array} \right\}$$

as the set of all possible optimal values, or the value space, of Player 0.

For example, let us consider the SC game $(\mathcal{A}, w)$ in Fig. 3, which is constructed based on DES G in Fig. 2. Since $W = 2$, $W' = 0$, $|V| = 13$ and $|V_m| = 3$, the space of the optimal per task value is

$$\Xi = \left\{ \frac{n}{d} : d = 1, 2, 3, n = 0, 1, 2, \ldots, 26 \right\}. \tag{5}$$

Recall that the objective of MPBG-PT is to find an optimal strategy that achieves the minimum value in Ξ. Since we have already shown that Ξ is a finite set, the optimal value can be found by solving a set of threshold decision problems defined as follows.

Problem 3 (*Per Task Value Decision Problem*)**.** Let $(\mathcal{A}, w)$ be a weighted game. Given a rational number $p \in \mathbb{Q}^+$, decide whether there exists a strategy $\theta_0 \in \Theta_0$ such that $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0; \theta_0) \le p$. If so, find such a strategy.

Clearly, if we can solve the above value decision problem, then we can start from the largest p in Ξ and decrease the threshold value until the answer to the decision problem becomes negative. Then the strategy achieving the minimum $p \in \Xi$ is the optimal strategy θ^*_0 achieving the optimal value $\mathsf{val}^{(w)}_{\mathsf{PT}}(v_0)$. Now, the question comes how to solve Problem 3.

5.2. Synthesis of p-shifted game

In order to solve the per task value decision problem, our approach is to reduce it to the standard mean payoff value decision problem defined as follows.

Problem 4 (*Mean Payoff Value Decision Problem*)**.** Let $(\mathcal{A}, w)$ be a weighted game. Given a rational number $p \in \mathbb{Q}^+$, decide whether or not we can find a strategy $\theta_0 \in \Theta_0$ such that $\mathsf{val}^{(w)}_{\mathsf{MP}}(v_0; \theta_0) \le p$. If so, find such a strategy.

Compared with Problem 3, Problem 4 simply replaces the per task criterion by the mean payoff criterion. Clearly, this decision problem can be solved by the standard mean payoff game algorithm. Specifically, for each $(\mathcal{A}, w)$, one can find the optimal positional strategy $\theta^*_{\mathsf{MP}} \in \Theta_0$. If $\mathsf{val}^{(w)}_{\mathsf{MP}}(v_0; \theta^*_{\mathsf{MP}}) \le p$, then it is the solution; otherwise, no solution exists.

Now we present our main construction for connecting the per task value decision problem with the standard mean payoff value decision problem. Specifically, for any $(\mathcal{A}, w)$ and p as the instance of Problem 3, we construct an instance of Problem 4 as follows.

Definition 1 (*p-Shifted Game*)**.** Let $(\mathcal{A}, w)$ be a weighted game, where $\mathcal{A} = (V = V_0 \dot{\cup} V_1, E, v_0)$ with accepting vertices V_m, and $p \in \mathbb{Q}^+$ be a rational number. We define $E_m = \{\langle (q, \gamma), q' \rangle \in E : q' \in V_m\}$ as the set of edges leading to target vertices. The p-shifted game of $(\mathcal{A}, w)$ is a new weighted game $(\mathcal{A}, \hat{w})$, where the arena $\mathcal{A}$ is the same and the weight function $\hat{w}$ is defined by:

- $\forall e \in E_m$, $\hat{w}(e) = w(e)$;
- $\forall e \in E \setminus E_m$, $\hat{w}(e) = w(e) + p$.

The above construction is motivated by the reduction from mean payoff games to energy games in Brim et al. (2011). However, here we need to further take the issue of target vertices into account. Furthermore, as we will show later, this construction is used to connect (not precisely reduce) our per task game with mean payoff game. Intuitively, for a weighted game, its p-shifted game simply adds value p to the weight of each edge leading to non-task vertices. Then the per task value decision problem and the mean payoff value decision problem can be connected using the p-shifted game. Note that, since $(\mathcal{A}, w)$ and $(\mathcal{A}, \hat{w})$ share the same arena $\mathcal{A}$, it makes sense to apply the same strategy $\theta_0 \in \Theta_0$ of Player 0 to both games. Now let us consider a play $\rho \in \mathsf{Play}(\mathcal{A}, v, \theta_0)$ and suppose that $\mathsf{MP}^{(\hat{w})}(\rho) \leq p$, i.e., the play satisfies the mean payoff threshold in game $(\mathcal{A}, \hat{w})$. Now, let us analyze the mean payoff per task value of the same run but in game $(\mathcal{A}, w)$, i.e., $\mathsf{PT}^{(w)}(\rho)$. By the construction of $\hat{w}$, we know that $\mathsf{MP}^{(\hat{w})}(\rho)$ consists of two parts:

- the payoff collected from w with $\mathsf{MP}^{(\hat{w})}(\rho)$;
- the added payoff p in each non-task edges.

Since $\mathsf{MP}^{(\hat{w})}(\rho) \leq p$, the (average) total payoff collected from w each time visiting task edges must be no larger than p, i.e., $\mathsf{PT}^{(w)}(\rho) \leq p$. Otherwise, by moving these payoff to each task edge in $(\mathcal{A}, \hat{w})$, the mean payoff will be larger than p. Similarly, we can argue that if $\mathsf{PT}^{(w)}(\rho) \leq p$, then we have $\mathsf{MP}^{(\hat{w})}(\rho) \leq p$.

Now, we formalize the above heuristic discussion and show that the two decision problems are related by the p-shifted game. Note that, according to Proposition 1 and Lemma 1, for both the mean payoff value and the per task value, one can focus on positional strategies for both players, which yield a play of prefix–suffix form.

Proposition 2. *Let $(\mathcal{A}, w)$ be a weighted game and $(\mathcal{A}, \hat{w})$ be its p-shifted game for some given $p \in \mathbb{Q}^+$. Let $\theta_0 \in \Theta_0$ be a positional strategy of Player 0 such that $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$ on arena $\mathcal{A}$. Then for any prefix–suffix form play $\rho = \rho_1(\rho_2)^\omega \in \mathsf{Play}(\mathcal{A}, v_0, \theta_0)$, we have*

$$\mathsf{PT}^{(w)}(\rho) \leq p \Leftrightarrow \mathsf{MP}^{(\hat{w})}(\rho) \leq p. \tag{6}$$

Proof. ($\Rightarrow$) Suppose $\mathsf{MP}^{(\hat{w})}(\rho) > p$ and let $\rho_2 = v_1 v_2 \cdots v_n v_{n+1}$. Then we have

$$\mathsf{MP}^{(\hat{w})}(\rho) = \limsup_{m \to \infty} \frac{\sum_{i<m} \hat{w}(\rho_{[i]}, \rho_{[i+1]})}{m} = \frac{\sum_{i=1}^{n} \hat{w}(v_i, v_{i+1})}{n}.$$

Also, since $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$, we have $N_m(\rho_2) \neq 0$. Then

$$\frac{\sum_{i=1}^{n} \hat{w}(v_i, v_{i+1})}{n} = \frac{\sum_{i=1}^{n} w(v_i, v_{i+1}) + (n - N_m(\rho_2)) \cdot p}{N_m(\rho_2) + (n - N_m(\rho_2))} > p,$$

which means that $\mathsf{PT}^{(w)}(\rho) = \frac{\sum_{i=1}^{n} w(v_i, v_{i+1})}{N_m(\rho_2)} > p$. However, this is a contradiction, which means that $\mathsf{MP}^{(\hat{w})}(\rho) \leq p$.

($\Leftarrow$) Suppose $\mathsf{PT}^{(w)}(\rho) > p$. Then we know that

$$\mathsf{PT}^{(w)}(\rho) = \limsup_{m \to \infty} \frac{\mathsf{Cost}(\rho_{[0,m]}(v_0, \theta_0^*, \theta_1))}{N_m(\rho_{[0,m]}(v_0, \theta_0^*, \theta_1))} = \frac{\sum_{i=1}^{n} w(v_i, v_{i+1})}{N_m(\rho_2)}.$$

As we consider non-negative weight function w, $N_m(\rho_2) \neq 0$. Otherwise, $\mathsf{MP}^{(\hat{w})}(\rho) > p$. Therefore,

$$\frac{\sum_{i=1}^{n} w(v_i, v_{i+1}) + (n - N_m(\rho_2)) \cdot p}{N_m(\rho_2) + (n - N_m(\rho_2))} = \frac{\sum_{i=1}^{n} \hat{w}(v_i, v_{i+1})}{n} > p,$$

which means that $\mathsf{MP}^{(\hat{w})}(\rho) = \frac{\sum_{i=1}^{n} \hat{w}(v_i, v_{i+1})}{n} > p$. However, this is a contradiction, which means that $\mathsf{PT}^{(w)}(\rho) \leq p$. ■

With the help of the above result, now we present the main theorem that establishes the relationship between Problem 3 and Problem 4.

3. *Given any rational number $p \in \Xi$ and weight function $\hat{w}$ defined with respect to p, for any positional strategy of Player 0 $\theta_0 \in \Theta_0$ such that $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$, we have*

$$\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0; \theta_0) \leq p \Leftrightarrow \mathsf{val}_{\mathsf{MP}}^{(\hat{w})}(v_0; \theta_0) \leq p.$$

Proof. We only prove the "$\Rightarrow$" direction since the other direction is analogous. Since $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0; \theta_0) \leq p$, by Eq. (3), we have $\forall \theta \in \Theta_1, \mathsf{PT}^{(w)}(\rho(v_0, \theta_0, \theta)) \leq p$. Clearly, this also holds when ρ is of the prefix–suffix form. Then by Proposition 2, for any play $\rho \in \mathsf{Play}(\mathcal{A}, v_0, \theta_0)$ of prefix–suffix form, we have $\mathsf{MP}^{(\hat{w})}(\rho) \leq p$, which also means that $\forall \theta \in \Theta_1, \mathsf{MP}^{(\hat{w})}(\rho(v_0, \theta_0, \theta)) \leq p$. Then by Eq. (4), we have $\mathsf{val}_{\mathsf{MP}}^{(\hat{w})}(v_0; \theta_0) \leq p$. ■

Therefore, given a rational number p, if we want to solve the per task value decision problem, we can only concentrate on the solution to the mean payoff value decision problem on the p-shifted game.

5.3. Synthesis algorithm

Algorithm 1: Optimal Strategy Search

Input: Weighted game $(\mathcal{A}, w)$ with V_m
Output: Optimal strategy θ_0^* for Problem 2

order $\Xi = \{p_1, p_2, \ldots, p_{|\Xi|}\}$ in increasing order;
$i_{min} \leftarrow 1$, $i_{max} \leftarrow |\Xi|$, $i = 0$;
while $i_{min} < i_{max}$ **do**
  $i \leftarrow \lfloor \frac{i_{min} + i_{max}}{2} \rfloor$;
  Construct p_i-shifted game $(\mathcal{A}, \hat{\omega})$;
  if $\exists \theta_0 \in \Theta_0$, $val_{MP}^{(\hat{w})}(v_0; \theta_0) \leq p_i$ **then**
    $i_{max} = i$;
    Update θ_0^* as θ_0;
  else
    $i_{min} = i + 1$;
return θ_0^*.

Based on the above discussions, we present Algorithm 1 as the main algorithm for solving MPBG-PT. Initially, we order all possible optimal values in Ξ in increasing order, i.e., $p_1 < p_2 < \cdots < p_{|\Xi|}$. To find the optimal value $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0)$, it suffices to find the smallest $p \in \Xi$ such that $\mathsf{val}_{\mathsf{PT}}^{(w)}(v_0) \leq p$ holds. In order to find such a value, we perform a binary search over the value space Ξ (the **while** loop). Specifically, for each value

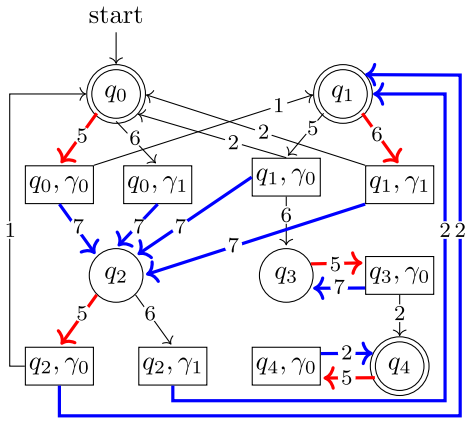


Fig. 4. The SC game $(\mathcal{A}, \hat{w}_p)$ with $p = 5$, where $\gamma_0 = \{a, d, e\}$ and $\gamma_1 = \{d, e\}$.

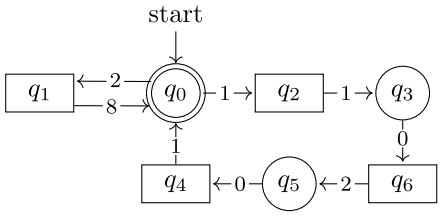


Fig. 5. Weighted game for Remark 2.

$p \in \Xi$ to be determined, we construct the corresponding p-shifted game $(\mathcal{A}, \hat{\omega})$ and solve the corresponding mean payoff value decision problem over the p-shifted game. If the answer to the decision problem is positive, then we keep the synthesized strategy θ as the current optimal strategy, and search for smaller p for later iterations; otherwise, we search for larger p for later iterations. Note that, we need to construct the p-shifted game for each p in the iteration. Once we obtain the optimal strategy θ^* to Problem 2, according to Theorem 1, its induced supervisor S_{θ^*} is the solution to the optimal supervisory control problem as formulated in Problem 1. Since we have shown that θ^* is positional, S_{θ^*} is state-based.

We illustrate the synthesis procedure by the following example.

Example 4. Let us still consider the SC game $(\mathcal{A}, w)$ in Fig. 3, where Ξ has been given in Eq. (5). By running Algorithm 1, we first consider value $p = 13 = \frac{0+26}{2} \in \Xi$. Then we construct the p-shifted game $(\mathcal{A}, \hat{w}_p)$ and compute the optimal mean payoff strategy θ_0^p for Player 0 on $(\mathcal{A}, \hat{w}_p)$, which is $\theta_0^p(q_0) = (q_0, \gamma_0)$, $\theta_0^p(q_1) = (q_1, \gamma_1)$, $\theta_0^p(q_2) = (q_2, \gamma_0)$, $\theta_0^p(q_3) = (q_3, \gamma_0)$ and $\theta_0^p(q_0) = (q_4, \gamma_0)$. For this strategy, we have $\mathsf{val}_{\mathsf{MP}}^{(\hat{w}_p)}(q_0; \theta_0^p) = 11 < 13$, which means that we should continue the binary search.

By iteratively repeating the above process, finally, the binary search process terminates with $p = 5 \in \Xi$ such that $\mathsf{val}_{\mathsf{MP}}^{(\hat{w}_p)}(q_0; \theta_0^p) = 5 \leq 5$. Therefore, we conclude that θ_0^p with $p = 5$ is the optimal strategy θ_0^* with $\mathsf{val}_{\mathsf{PT}}^{(w)}(q_0; \theta_0^*) = 5$. In particular, the p-shifted game $(\mathcal{A}, \hat{w}_p)$ with $p = 5$ is shown in Fig. 4. The red arrows in the figure denote the optimal positional strategy θ_0^* for the mean payoff game w.r.t. $\hat{w}_p$, which is also the optimal strategy for the per task game. Finally, we obtain optimal supervisor S^* for G based on θ_0^*, which works as follows:

- $S^*(s) = \gamma_0 = \{a, d, e\}$, when $\delta(q_0, s) \neq q_1$;
- $S^*(s) = \gamma_1 = \{d, e\}$, when $\delta(q_0, s) = q_1$.

Remark 2. Note that Algorithm 1 requires to iteratively construct p-shift games for different value p and solve the corresponding mean payoff games. In the above example, we see that the optimal strategy for each value of Player 0 keeps unchanged during the iteration process, which is equal to the optimal strategy θ_0^*. Therefore, one may ask if do we really need to solve mean payoff games over the same graph but for different values iteratively. In other words, once we obtain the optimal mean payoff strategy for some value p, is it possible that the mean payoff strategy is not optimal for different p. The answer is yes, which justifies why iterations are needed. To see this, let us consider another weight game shown in Fig. 5. Obviously, the optimal value is $\mathsf{val}_{\mathsf{PT}}^{(w)}(q_0) = 5$ and $\Xi = \{\frac{n}{d} : d = 1, n = 0, 1, 2, \ldots, 56\}$. However, at the initial stage of the iteration, when p is chosen between 12.5 and 56, the optimal mean payoff strategy over the p-shift game is to choose vertex q_1 at q_0. However, once the value of p is smaller than 12.5, the optimal mean payoff strategy over the p-shift game changes to choose vertex q_2 at q_0, which is the actual optimal strategy for the optimization problem. This observation also justifies why we cannot reduce the per task game to the standard mean payoff game directly since we do not know the optimal value p a priori. Therefore, in order to establish the reduction, we have to fix the decision value p first and then approach the optimal value gradually.

5.4. Complexity analysis

We conclude this section by discussing the complexity of the overall synthesis algorithm. Let V and E be the set of vertices and edges in $\mathcal{A}$, respectively, and let W be the largest weight in $\mathcal{A}$. The **while** loop in Algorithm 1 executes at most $\mathcal{O}(\log|\Xi|)$ times, where Ξ contains at most $W \cdot |V|^2$ elements (very roughly estimated). For each iteration of the **while** loop, the p-shifted game has exactly the same numbers of vertices and edges. To solve the mean payoff decision problem and strategy synthesis problem over $(\mathcal{A}, \hat{w})$, one can use the algorithm developed recently (Brim et al., 2011) with the complexity being $\mathcal{O}(|V|^2 \cdot W \cdot |E|)$. Therefore, the overall complexity of Algorithm 1 is $\mathcal{O}(|V|^2 \cdot W \cdot |E| \cdot \log(W \cdot |V|))$. Finally, we recall that the SC game is constructed from the original DES G and its size is linear in the number of states $|Q|$ and exponential in the number of controllable events $|\Sigma_c|$. In large systems, where the complexity is an important issue, the number of states is usually much bigger than the number of controllable actions. In such cases, the size of the state–space is the main parameter in the scalability of the synthesis algorithm.

Note that, MPBG-PT can also be solved by formulating it as an instance of a ratio-game, for which an effective algorithm is provided in Bloem et al. (2014). The reader is referred to Bloem et al. (2014) for more details about ratio-games. In brief, it considers a ratio value of two different costs and to capture our requirement, we can assign a unit cost to the denominator and zero otherwise. However, the complexity of using (Bloem et al., 2014) to solve our problem is $\mathcal{O}\left(|V|^3 \cdot W \cdot |E| \cdot \log(W \cdot |V|) \cdot \log \frac{|E|}{|V|}\right)$. Compared to our method, the algorithm in Bloem et al. (2014) requires an additional complexity of $\mathcal{O}\left(|V| \cdot \log \frac{|E|}{|V|}\right)$. The reason for this is that the general ratio-game algorithm needs to search the optimal value and optimal strategy separately and once the optimal ratio-value has been computed, iterative group-tests are performed to find the optimal ratio-game strategy, which introduces the additional complexity mentioned above. However, in Proposition 2 and Theorem 3, we have established a key structural property of MPBG-PT such that, based on the p-shifted game, when searching the possible value space, once this value converges to the optimum, the corresponding optimal strategy for the reduced p-shifted game is indeed the optimal strategy for the original

Table 1
Statistics for systems with fixed number of events $|\Sigma| = 10$.

$\|Q\|$	50	100	200
t (s)	3.76	43.65	325.52

Table 2
Statistics for systems with fixed number of states $|Q| = 100$.

$\|\Sigma\|$	5	10	20
t (s)	8.63	38.31	1005.46

per-task game. With the help of this new structural property for the per-task game, we are able to combine the value search and the strategy search in the same iteration loop, which reduces the group-test complexity of $\mathcal{O}\left(|V|\cdot\log\frac{|E|}{|V|}\right)$ that is required in Bloem et al. (2014) after the binary search. In other words, although our problem setting is less expressive than existing works, we achieve better computational performance for this restrictive class.

6. Experimental results

In this section, we first present a set of numerical experiments to test the scalability of the proposed method. Then we illustrate the proposed optimal supervisory control problem by a case study of robot planning. All experiments are implemented by `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. All codes, the simulation video and other auxiliary materials are available in our project website.[1]

6.1. Numerical experiments

First, we show the scalability performance of the proposed approach with respect to the size of plant G. Specifically, for each size of the plant, we randomly generate 20 automata with the same numbers of states $|Q|$ and events $|\Sigma|$. Then we compute the average running time of our algorithm for solving the 20 different systems with the same size. For each random plant generation, the parameters are chosen as follows: $|Q_m| = \frac{1}{10}|Q|$, $|\Sigma_c| = \frac{3}{5}|\Sigma|$, the maximal number of transitions defined at each state is $\lceil\frac{1}{4}|\Sigma|\rceil$, $c_e(\sigma)$ is chosen randomly in $[1, 5]$ and $c_d(\sigma)$ is chosen randomly in $[0, 3]$.

In the first set of experiments, we investigate how the number of states in the plant affects the overall running time. To this end, we fix the number of events in G as $|\Sigma| = 10$. Then we increase the number of states $|Q|$ from 50 to 200. The averaged statistics for the running times are shown in Table 1. In the second set of experiments, we investigate how the number of events in the plant affects the overall running time. Here, we fix the number of states in G as $|Q| = 100$ and increase the number of events $|\Sigma|$ from 5 to 20. The results are shown in Table 2. Based on the above numerical experiments, clearly, we see that the running time of our algorithm is more sensitive to the number of events in the plant. This result is consistent with our theoretical analysis since the numbers of edges and vertices of the underlying game graph grows exponentially in the number of controllable event.

Next, we perform numerical experiments to show the computational efficiency of our approach compared with the standard ratio-game algorithm (Bloem et al., 2014). Specifically, we consider four batches of experiments, in which the number of states in the DES model is increased from 10 to 25,50 and 100. For each batch with the same number states, we randomly generate 20 DES models and apply both our synthesis algorithm and the algorithm in Bloem et al. (2014) to solve the MPBG-PT. We compute the average running time of the 20 randomly generated models as the statistic result for this batch. For each random model generation, the parameters are as follows: $|Q_m| = \frac{|Q|}{10}$, $|\Sigma_c| = 10$, $|\Sigma_{uc}| = 2$, the maximal number of transitions defined at each state is $\lceil\frac{1}{4}|\Sigma|\rceil$, $c_e(\sigma)$ is chosen randomly in $[1, 5]$ and $c_d(\sigma)$ is chosen randomly in $[0, 3]$. The experimental results are shown in Table 3. We use t and t' to denote the running time of our algorithm and that in Bloem et al. (2014) respectively. We use "—" to denote that the running time exceeds 10000 s Clearly, although the ratio-game algorithm is more general, in terms of the MPBG-PT under investigation, our algorithm is more efficient not only theoretically, but also in practice.

Table 3
Statistics for comparison between our algorithm and the standard ratio-game algorithm for solving MPBG-PT.

$\|Q\|$	10	25	50	100
t (s)	0.037	0.293	4.25	56.23
t' (s)	2.596	44.376	1252.43	—

6.2. Simulation experiment

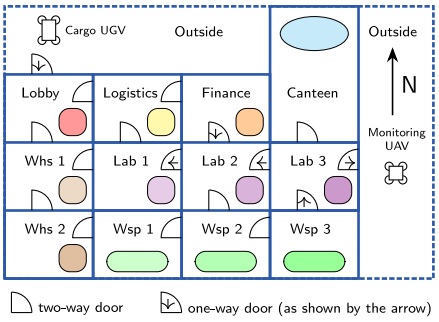
In this part, we illustrate the proposed optimal supervisory control problem by a case study of robot planning.

System Model: We consider a scenario where a cargo UGV moves in a factory as shown in Fig. 6(a). The factory has twelve regions of interest: a lobby, a logistics, a finance, a canteen, two warehouse (Whs 1-2), three laboratories (Lab 1-3) and three workshops (Wsp 1-3). The UGV can deliver cargoes from one region to another by passing through doors. Depending on the security law of the factory, some doors are one-way but some are two-way, as depicted in the figure. For simplicity, we consider four moving actions for the robot E, S, W and N, which represent advancing to east, south, west and north, respectively. The mobility of the UGV is abstracted as a DES shown in Fig. 6(b), where states q_0 to q_{12} correspond to regions outside, lobby, logistics, finance, canteen, Whs 1, Lab 1, Lab 2, Lab 3, Whs 2, Wsp 1, Wsp 2 and Wsp 3, respectively.

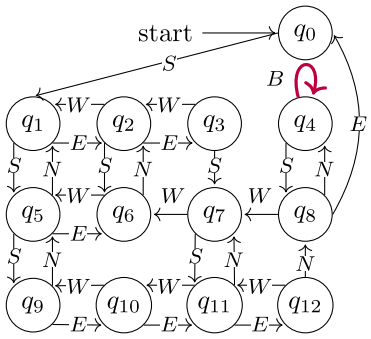
Control Capabilities: We assume that all doors in the factory can be controlled by the building manager. That is, if there is a door in any direction, we can disallow the UGV from moving in that direction by closing the door, which means that the UGV can only choose one of the remaining open doors to pass through. Therefore, all the moving actions, E, S, W and N, are controllable. The execution costs for the four actions are all assumed to be one unit and the cost to close any door (the disable cost for each action) is also assumed to be one unit.

Uncontrollable Events: In this example, we consider two different sources of uncontrollability. First, we assume that the UGV may break down in canteen due to the wet environment, which is captured as an uncontrollable action B in Fig. 6(b) with its executing cost being one unit. Furthermore, we consider another UAV that hovers over the factory to monitor the operation of equipment in the three laboratories, whose mobility can be modeled as a DES shown in Fig. 6(d). We assume that the aerial space in each region is interconnected, allowing unrestricted movement of the UAV without any constraints imposed by doors. At each instant, the UAV can randomly choose to continue monitoring the current region or go to adjacent regions through three actions H, $\overline{W}$ and $\overline{E}$. Therefore, the movement of the UAV is not controllable

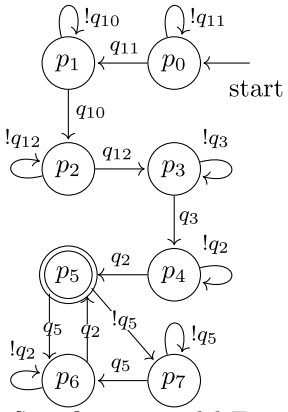
[1] https://github.com/Lv-Peng/Optimal-Supervisory-Control-for-Cyclic-Tasks



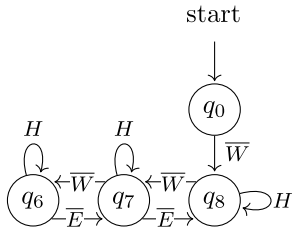
(a) The topology of the factory



(b) Move model G_0 for the UGV



(c) Specification model T



(d) Move model G_1 for the UAV

(e) A trajectory with higher cost.

(f) A trajectory with lower cost.

Fig. 6. Illustrative example of a mobile robot.

from our point of view. Note that, since we are only interested in controlling the UGV, the execution costs for uncontrollable events related to the UAV are all assumed to be zero.

Objectives: The objectives of the UGV are threefolds: *(1) safety constraint*: to avoid signal interference between each other, it cannot stay with the UAV in the same region at the same time. *(2) specifications*: it should achieve the task specification captured by the system shown in Fig. 6(c). Note that, each event in it means that the UGV moves to some region or not. For example, q_{10} means that the UGV is in Wsp 1 while $!q_{10}$ means that it is not in Wsp 1. Therefore, this specification essentially requires that the UGV first needs to deliver cargoes from Wsp 2 to Wsp 1, then deliver cargoes from Wsp 1 to Wsp 3, next deliver cargoes from Wsp 3 to finance, and then return to logistics to report. Finally, it needs to continuously deliver cargoes between Whs 1 and logistics. *(3) cost optimization*: each time that the UGV returns to logistics is considered as a completion of task and it should optimize the average cost per task.

DES Construction: In order to solve the optimal supervisory control problem, we need to build DES G, which is the product of G_0, G_1 and T. Specifically, since the movements of the UGV and UAV are independent, their synchronization $G_0 \times G_1$ is a pure shuffle. Moreover, since properties of interest are on states in $G_0 \times G_1$ while properties of interest are on edges in the specification model T, states in $G_0 \times G_1$ need to be synchronized with edges in T in order to obtain the overall DES $G = G_0 \times G_1 \times T$. We refer to Lacerda, Parker, and Hawes (2014) for details of this type of product. In particular, since there are many states that are either unreachable or unrelated to the synthesis of the optimal strategy, we use some pruning methods to remove redundant states and the resulting system for synthesis only has 40 states and 3 marked states with the controllable and uncontrollable events sets being $\Sigma_c = \{E, S, W, N\}$ and $\Sigma_{uc} = \{B, H, \overline{W}, \overline{E}\}$, respectively. The DES model of G as well as the optimal supervisor is available on our project website.[2]

Synthesis Result: With our synthesis algorithm, the optimal supervisor S^* for this example is found in 2.17 sec and we perform simulations on simulation platform `V-REP 4.2.0`. In Figs. 6(e) and 6(f), we show two different simulation trajectories under S^* upon the first time visiting marked states. The reason for the different trajectories is that, when the UGV begins to continuously deliver cargoes between Whs 1 and logistics, the UGV needs to adopt different strategies in response to the different behaviors of the UAV. Specifically:

- When the UGV arrives at logistics or Whs 1 and the UAV arrives at Lab 1 or Lab 2, since the UGV cannot determine whether the UAV will proceed to Lab 1 next step or not, supervisor S^* must disable E and S to avoid potential collisions. This corresponds to the trajectory shown in Fig. 6(e).
- On the other hand, when the UGV arrives at logistics or Whs 1, but the UAV arrives at Lab 3, since the UAV will definitely not reach Lab 1 next step, supervisor S^* can enable E and S. This corresponds to the trajectory shown in Fig. 6(f).

Note that the first trajectory incurs a larger cost to complete the task for the first time compared with the second one. Here, we only show two trajectories upon the first visit of marked states. In fact, the two robots work indefinitely, which results in a random combination of these two trajectories for different times of visiting marked states.

[2] This model essentially captures asynchronous movements between the two agents. In our simulation, to enforce synchronous movements, we further add a binary variable to encode turn-based decision of the two agents.

7. Conclusion

In this paper, we solve a class of optimal supervisory control problem for discrete-event systems under the average cost per task metric. This setting captures the scenario where tasks, modeled by marked states, need to be completed indefinitely in a cyclic manner. Although the optimality metric considered can be treated as a special instance of the ratio value cost, we provide a more efficient new algorithm which achieves the trade-off between computational efficiency and expressiveness of the optimality metric. In particular, we showed that the optimal strategy for this problem is positional and the optimal value is a rational number in a finite set. With the notion of p-shifted games, we reduce this problem to a set of mean payoff value decision problems so that the optimal strategy can be effectively synthesized. Our results extend the theory of optimal supervisory control of DES. In the future, we plan to further extend our result to the partial observation setting as well as stochastic systems modeled as Markov decision processes.

Appendix

In order to prove Theorem 1, we first prove some basic properties of the induced supervisors as well as the induced strategies satisfying some given conditions. Then, the proof for Theorem 1 follows directly.

Proposition 3. *For any strategy $\theta_0 \in \tilde{\Theta}_0$ such that $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$ and $\mathsf{val}^{(\omega)}_{\mathsf{PT}}(v_0; \theta_0) < \infty$, its induced supervisor S_{θ_0} is live, non-blocking and $\mathsf{Cost}^{AveT}_{S_{\theta_0}}(s) < \infty, \forall s \in \mathcal{L}^{\omega}(S_{\theta_0}/G)$.*

Proof. Suppose that S_{θ_0} is not live, then there exists a string $s = \sigma_1 \cdots \sigma_n \in \mathcal{L}(S_{\theta_0}/G)$, such that there exists no event $\sigma \in \Sigma : s\sigma \in \mathcal{L}(S_{\theta_0}/G)$. From the definition of $\mathcal{A}$, there exists a play $\rho = q_0(q_0, \gamma_0)q_1(q_1, \gamma_1)\cdots q_n \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$, where $\sigma_i \in \gamma_{i-1}$. As $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$, then there exists a play $\rho' = (q_n, \gamma_n)q_{n+1}$, $\rho\rho' \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$. Therefore, there exists a string $s = \sigma_1 \cdots \sigma_n\sigma_{n+1} \in \mathcal{L}(S_{\theta_0}/G)$, where $\sigma_{n+1} \in \gamma_n$, which is a contradiction.

Suppose that S_{θ_0} is blocking, then there exists a string $s = \sigma_1 \cdots \sigma_n \in \mathcal{L}(S_{\theta_0}/G)$, such that there exists no string $s' \in \Sigma^* : ss' \in \mathcal{L}_m(S_{\theta_0}/G)$. From the definition of $\mathcal{A}$, there exists a play $\rho = q_0(q_0, \gamma_0)q_1(q_1, \gamma_1)\ldots q_n \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$, where $\sigma_i \in \gamma_{i-1}$. As $\mathsf{Play}(\mathcal{A}, v_0, \theta_0) \subseteq \mathsf{Win}_B(V_m)$, then there exists a play $\rho' = (q_n, \gamma_n)q_{n+1}\ (q_{n+1}, \gamma_{n+1})\cdots q_m$, such that $\rho\rho' \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$ and $q_m \in V_m$. Therefore, there exists a string $s = \sigma_1 \cdots \sigma_n\sigma_{n+1}\cdots\sigma_m \in \mathcal{L}_m(S_{\theta_0}/G)$, where $\sigma_i \in \gamma_{i-1}$, which is a contradiction.

Suppose there exists a string $s = \sigma_1 \cdots \sigma_n \cdots \in \mathcal{L}^{\omega}(S_{\theta_0}/G)$, such that $\mathsf{Cost}^{AveT}_{S_{\theta_0}}(s) = \infty$. From the definition of $\mathcal{A}$, there exists a play $\rho = q_0(q_0, \gamma_0)\cdots q_n(q_n, \gamma_n)\cdots \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$, where $\sigma_i \in \gamma_{i-1}$. From the definition of I_m and N_m, we can easily get that

$$\limsup_{n\to\infty} I_m(s_{[1,n]}) = \limsup_{k\to\infty} N_m(\rho_{[1,k]}).$$

From the definition of weight function w for $\mathcal{A}$, we also get that

$$\limsup_{p\to\infty} \mathsf{Cost}_{S_{\theta_0}}(s_{[1,p]}) = \limsup_{q\to\infty} \mathsf{Cost}(\rho_{[1,q]}).$$

Then, we have

$$\begin{aligned}
&\mathsf{Cost}^{AveT}_{S_{\theta_0}}(s)\\
=&\limsup_{n\to\infty}\left\{\frac{1}{I_m(s_{[1,n]})}\mathsf{Cost}_{S_{\theta_0}}(s_{[1,n]})\right\}\\
=&\frac{\limsup_{n\to\infty}\mathsf{Cost}_{S_{\theta_0}}(s_{[1,n]})}{\limsup_{n\to\infty} I_m(s_{[1,n]})} = \frac{\limsup_{p\to\infty}\mathsf{Cost}(\rho_{[1,p]})}{\limsup_{p\to\infty} N_m(\rho_{[1,p]})}\\
=&\limsup_{p\to\infty}\left\{\frac{\mathsf{Cost}\rho_{[1,p]}}{N_m(\rho_{[1,p]})}\right\}\\
=&\mathsf{PT}^{(\omega)}(\rho) = \infty.
\end{aligned}$$

As $\mathsf{val}^{(\omega)}_{\mathsf{PT}}(v_0; \theta_0) = \sup_{\theta_1\in\Theta_1}\mathsf{PT}^{(\omega)}(\rho(v_0, \theta_0, \theta_1)) < \infty$, it is a contradiction. ■

Proposition 4. *For any live and non-blocking supervisor S such that $\mathsf{Cost}^{AveT}_S(s) < \infty, \forall s \in \mathcal{L}^{\omega}(S/G)$, its induced strategy θ_S satisfies $\mathsf{Play}(\mathcal{A}, v_0, \theta_S) \subseteq \mathsf{Win}_B(V_m)$ and $\mathsf{val}^{(\omega)}_{\mathsf{PT}}(v_0; \theta_S) < \infty$.*

Proof. Suppose $\mathsf{Play}(\mathcal{A}, v_0, \theta_S) \nsubseteq \mathsf{Win}_B(V_m)$, then there exists a play $\rho = q_0(q_0, \gamma_0)q_1\ldots q_n(q_n, \gamma_n)\cdots \in \mathsf{Play}(\mathcal{A}, v_0, \theta_S)$, such that $\mathrm{Inf}(\rho) \cap V_m = \emptyset$. From the definition of G, there exists a string $s = \sigma_1\cdots\sigma_n\cdots \in \mathcal{L}^{\omega}(S/G)$, where $\sigma_i \in \gamma_{i-1}$. Then, we know that $I_m(s) < \infty$. However, as S is live and non-blocking and $\mathsf{Cost}^{AveT}_S(s') < \infty, \forall s' \in \mathcal{L}^{\omega}(S/G)$, then it must hold that $I_m(s) = \infty$, which is a contradiction.

Next, we prove the second part. Suppose there exists a play $\rho = q_0(q_0, \gamma_0)\cdots q_n(q_n, \gamma_n)\cdots \in \mathsf{Play}(\mathcal{A}, q_0, \theta_0)$, such that $\mathsf{val}^{(\omega)}_{\mathsf{PT}}(v_0; \theta_S) = \mathsf{PT}^{(\omega)}(\rho) = \infty$. From the definition of G, there exists a string $s = \sigma_1\cdots\sigma_n\cdots \in \mathcal{L}^{\omega}(S/G)$, where $\sigma_i \in \gamma_{i-1}$. From the definition of I_m and N_m, we know that

$$\limsup_{n\to\infty} I_m(s_{[1,n]}) = \limsup_{k\to\infty} N_m(\rho_{[1,k]}).$$

Similarly, from the definition of weight function w for $\mathcal{A}$, we also have

$$\limsup_{p\to\infty} \mathsf{Cost}_S(s_{[1,p]}) = \limsup_{q\to\infty} \mathsf{Cost}(\rho_{[1,q]}).$$

Then, we have

$$\begin{aligned}
&\mathsf{val}^{(\omega)}_{\mathsf{PT}}(v_0; \theta_S)\\
=&\mathsf{PT}^{(\omega)}(\rho) = \limsup_{p\to\infty}\left\{\frac{\mathsf{Cost}\rho_{[1,p]}}{N_m(\rho_{[1,p]})}\right\}\\
=&\frac{\limsup_{p\to\infty}\mathsf{Cost}(\rho_{[1,p]})}{\limsup_{p\to\infty} N_m(\rho_{[1,p]})} = \frac{\limsup_{n\to\infty}\mathsf{Cost}_S(s_{[1,n]})}{\limsup_{n\to\infty} I_m(s_{[1,n]})}\\
=&\limsup_{n\to\infty}\left\{\frac{1}{I_m(s_{[1,n]})}\mathsf{Cost}_S(s_{[1,n]})\right\}\\
=&\mathsf{Cost}^{AveT}_S(s) = \infty.
\end{aligned}$$

However, we have $\mathsf{Cost}^{AveT}_S(s) < \infty, \forall s \in \mathcal{L}^{\omega}(S/G)$, which is a contradiction. ■

Based on the above two propositions, now we are ready to prove the main theorem.

Proof for Theorem 1. We prove the " $\Rightarrow$ " direction here and the other direction is analogous. As θ_0^* solves Problem 2, from Propositions 3 and 4, we know that $S_{\theta_0^*}$ is live and non-blocking. Suppose $S_{\theta_0^*}$ does not solve Problem 1, which means there exists another live and non-blocking supervisor $S^\star$, such that

$$\sup_{s\in\mathcal{L}^{\omega}(S^\star/G)}\mathsf{Cost}^{AveT}_{S^\star}(s) < \sup_{s\in\mathcal{L}^{\omega}(S^*_{\theta_0}/G)}\mathsf{Cost}^{AveT}_{S_{\theta_0^*}}(s),$$

$$\sup_{s\in\mathcal{L}^{\omega}(S^\star/G)}\mathsf{Cost}^{AveT}_{S^\star}(s) \le \sup_{s\in\mathcal{L}^{\omega}(S'/G)}\mathsf{Cost}^{AveT}_{S'}(s)$$

for any other supervisor S'. Let $\theta_0^\star$ be the strategy induced from $S^\star$. From the proof of Proposition 3, we also have

$$\mathsf{val}_{\mathsf{PT}}^{(\omega)}(v_0;\theta_0^\star) = \sup_{s\in\mathcal{L}^\omega(S^\star/G)} \mathsf{Cost}_{S^\star}^{AveT}(s),$$

$$\mathsf{val}_{\mathsf{PT}}^{(\omega)}(v_0;\theta_0^*) = \sup_{s\in\mathcal{L}^\omega(S^*_{\theta_0^*}/G)} \mathsf{Cost}_{S^*_{\theta_0^*}}^{AveT}(s).$$

Therefore, we have $\mathsf{val}_{\mathsf{PT}}^{(\omega)}(v_0;\theta_0^\star) < \mathsf{val}_{\mathsf{PT}}^{\omega}(v_0;\theta_0^*)$, which means that $\theta_0^\star$ solves Problem 2, which is a contradiction. ■

References

Alves, Lucas V. R., Pena, Patrícia N., & Takahashi, Ricardo H. C. (2021). Planning on discrete event systems using parallelism maximization. *Control Engineering Practice*, *112*, Article 104813.

Bloem, Roderick, Chatterjee, Krishnendu, Greimel, Karin, Henzinger, Thomas A, Hofferek, Georg, Jobstmann, Barbara, et al. (2014). Synthesizing robust systems. *Acta Informatica*, *51*, 193–220.

Brim, Lubos, Chaloupka, Jakub, Doyen, Laurent, Gentilini, Raffaella, & Raskin, Jean-François (2011). Faster algorithms for mean-payoff games. *Formal Methods in System Design*, *38*(2), 97–118.

Cassandras, Christos G., & Lafortune, Stephane (2009). *Introduction to discrete event systems*. Springer Science & Business Media.

Chatterjee, Krishnendu, Henzinger, Thomas A, & Jurdzinski, Marcin (2005). Mean-payoff parity games. In *20th annual IEEE symposium on logic in computer science* (pp. 178–187).

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.

Ehrenfeucht, Andrzej, & Mycielski, Jan (1979). Positional strategies for mean payoff games. *International Journal of Game Theory*, *8*, 109–113.

Fu, Jinbo, Ray, Asok, & Lagoa, Constantino M. (2004). Unconstrained optimal control of regular languages. *Automatica*, *40*(4), 639–646.

Gimbert, Hugo, & Zielonka, Wiesław (2005). Games where you can play optimally without any memory. In *International conference on concurrency theory* (pp. 428–442). Springer.

Gradel, Erich, & Thomas, Wolfgang (2002). *Automata, kogics, and infinite games: A guide to current research*. Springer Science & Business Media.

Grigorov, Lenko, & Rudie, Karen (2006). Near-optimal online control of dynamic discrete-event systems. *Discrete Event Dynamic Systems*, *16*(4), 419–449.

Hill, Richard C., & Lafortune, Stéphane (2016). Planning under abstraction within a supervisory control context. In *55th IEEE conference on decision and control* (pp. 4770–4777).

Ji, Yiding, Yin, Xiang, & Lafortune, Stéphane (2021a). Local mean payoff supervisory control for discrete event systems. *IEEE Transactions on Automatic Control*, *67*(5), 2282–2297.

Ji, Yiding, Yin, Xiang, & Lafortune, Stéphane (2021b). Optimal supervisory control with mean payoff objectives and under partial observation. *Automatica*, *123*, Article 109359.

Kumar, Ratnesh, & Garg, Vijay K. (1995). Optimal supervisory control of discrete event dynamical systems. *SIAM Journal on Control and Optimization*, *33*(2), 419–439.

Lacerda, Bruno, Parker, David, & Hawes, Nick (2014). Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 1511–1516).

Li, Jinglun, & Takai, Shigemasa (2022). Synthesis of maximally permissive supervisors for similarity control of partially observed nondeterministic discrete event systems. *Automatica*, *135*, Article 109978.

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, & Cai, Kai (2022). On resilient supervisory control against indefinite actuator attacks in discrete-event systems. *IEEE Control Systems Letters*.

Ma, Ziyue, & Zhang, Jiafeng (2020). Determining optimal control sequences for reconfiguration in Petri nets using cost trees. In *59th IEEE conference on decision and control* (pp. 4485–4491).

Marchand, Hervé, Boivineau, Olivier, & Lafortune, Stéphane (2000). On the synthesis of optimal schedulers in discrete event control problems with multiple goals. *SIAM Journal on Control and Optimization*, *39*(2), 512–532.

Marchand, Hervé, Boivineau, Olivier, & Lafortune, Stéphane (2002). On optimal control of a class of partially observed discrete event systems. *Automatica*, *38*(11), 1935–1943.

Pantelic, Vera, & Lawford, Mark (2011). Optimal supervisory control of probabilistic discrete event systems. *IEEE Transactions on Automatic Control*, *57*(5), 1110–1124.

Pena, Patrícia N, Vilela, Juliana N, Alves, Michel RC, & Rafael, Gustavo C (2021). Abstraction of the supervisory control solution to deal with planning problems in manufacturing systems. *IEEE Transactions on Automatic Control*, *67*(1), 344–350.

Pruekprasert, Sasinee, & Ushio, Toshimitsu (2016). Optimal stabilizing controller for the region of weak attraction under the influence of disturbances. *IEICE Transactions on Information and Systems*, *99*(6), 1428–1435.

Pruekprasert, Sasinee, Ushio, Toshimitsu, & Kanazawa, Takafumi (2015). Quantitative supervisory control game for discrete event systems. *IEEE Transactions on Automatic Control*, *61*(10), 2987–3000.

Sakakibara, Ami, Urabe, Natsuki, & Ushio, Toshimitsu (2021). Finite-memory supervisory control of discrete event systems for LTL [f] specifications. *IEEE Transactions on Automatic Control*.

Sakakibara, Ami, & Ushio, Toshimitsu (2020). On-line permissive supervisory control of discrete event systems for scLTL specifications. *IEEE Control Systems Letters*, *4*(3), 530–535.

Schmidt, Klaus Werner (2015). Optimal supervisory control of discrete event systems: cyclicity and interleaving of tasks. *SIAM Journal on Control and Optimization*, *53*(3), 1425–1439.

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

Sengupta, Raja, & Lafortune, Stéphane (1998). An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, *36*(2), 488–541.

Su, Rong, Van Schuppen, Jan H., & Rooda, Jacobus E. (2011). The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Transactions on Automatic Control*, *57*(1), 105–118.

Takai, Shigemasa (2021). Synthesis of maximally permissive supervisors for nondeterministic discrete event systems with nondeterministic specifications. *IEEE Transactions on Automatic Control*, *66*(7), 3197–3204.

van der Sanden, Bram (2018). *Performance analysis and optimization of supervisory controllers* (Ph.D. thesis), Eindhoven University of Technology.

Ware, Simon, & Su, Rong (2016). Time optimal synthesis based upon sequential abstraction and its application to cluster tools. *IEEE Transactions on Automation Science and Engineering*, *14*(2), 772–784.

Wonham, W. Murray, & Cai, Kai (2019). *Supervisory control of discrete-event systems*. Springer.

Yin, Xiang, & Lafortune, Stéphane (2016a). Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, *61*(5), 1239–1254.

Yin, Xiang, & Lafortune, Stéphane (2016b). A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, *61*(8), 2140–2154.

Zwick, Uri, & Paterson, Mike (1996). The complexity of mean payoff games on graphs. *Theoretical Computer Science*, *158*(1–2), 343–359.

Peng Lv was born in Heilongjiang, China, in 1998. He received the B.Eng degree in automation from Harbin Engineering University in 2020. He is currently a Doctoral student at Shanghai Jiao Tong University. His research interests include formal methods, temporal logic task planning and game theory in Discrete-Event Systems.

Zhangcong Xu was an undergraduate student at the Department of Automation, Shanghai Jiao Tong University.

Yiding Ji is an assistant professor of Robotics and Autonomous Systems Thrust, Hong Kong University of Science and Technology (Guangzhou), China, also affiliated with Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), China. He received bachelor's degree of Electrical Engineering and its Automation from Tianjin University, China in 2014, then master's degree and doctor's degree of Electrical and Computer Engineering both from the University of Michigan, United States, in 2016 and 2019, respectively. From 2019 to 2021, he was a postdoc researcher at Boston University in United States, then a research scientist at Siemens Corporation in United States. He joined HKUST in late 2021. His research interests lie in the general field of systems science, especially automatic control and cyber physical systems. He is a member of Institute of Electrical and Electronics Engineers (IEEE) and IEEE Control Systems Society Technical Community on Discrete Event Systems. He has served as reviewers for multiple top academic journals and conferences.

Shaoyuan Li was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree from Nankai University, Tianjin, in 1997. Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a chair Professor. His current research interests include model predictive control, dynamic system optimization, and cyber–physical systems. He is the vice-president of the Chinese Association of Automation.

Xiang Yin was born in Anhui, China, in 1991. He received the B.Eng degree from Zhejiang University in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering. Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, where he is an Associate Professor. His research interests include formal methods, discrete-event systems and cyber–physical systems. Dr. Yin is serving as the chair of the *IEEE CSS Technical Committee on Discrete Event Systems*, Associate Editors for the *Journal of Discrete Event Dynamic Systems: Theory & Applications*, the *IEEE Control Systems Letters*, the *IEEE Transactions on Intelligent Vehicles*, and a member of the *IEEE CSS Conference Editorial Board*. Dr. Yin received the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016.