A Game-Theoretical Approach for Optimal Supervisory Control of Discrete Event Systems for Cyclic Tasks

Peng Lv, Xiang Yin, Yiding Ji and Shaoyuan Li

***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 o ften*. F ollowing 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 propose a new 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. The constructed game is then solved in two stages: one focuses on the optimal execution within each single task cycle and the other focuses on the scheduling strategy among different tasks. Illustrative examples are provided to demonstrate the proposed algorithm.**

I. Introduction

Discrete event systems (DES) are widely used in the modeling and analysis of man-made engineering cyber-physical systems such that manufacturing systems, transportation systems and communication networks [1]. 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 s ome desired specification, s uch a s s afety, l iveness a nd non-blockingness, in the presence of uncontrollable events [2], [3].

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., [4]–[11]. Particularly, an optimal supervisory control framework was proposed in [5] by considering both the occurrence cost and the disable cost. The objective is to reach marked states with the smallest worst-case *accumulated total cost*. This framework has been extended subsequently to several different settings, including, e.g., multiple goals [12], partial observations [13] and probabilistic systems [14]. However, the framework of [5] 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., [15]–[17]. 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 [5], the authors in [18] 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.

In this paper, we formulate and solve a new class of optimal supervisory control problem for cyclic tasks. We also follow the basic setting in [5], 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 as the system needs to visit marked states infinitely often. To formulate the optimal control problem, we introduce a new 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. A game-theoretical approach is developed to solve the proposed optimal control problem. Specifically, by converting the supervisory control problem into a two-player graph game, the overall synthesis problem is decomposed into two parts. The first part focuses on the optimal micro-strategy for each single task cycle. The second part focuses on how to design cyclic macro-strategy over different tasks. By merging the micro and macro-strategies together, an optimal solution that minimizes the average cost per tasks in the worst-case is obtained.

Our solution methodology is motivated by the two-player graph games; see, e.g., [19]. In particular, our algorithm partially leverages the standard Büchi game and mean-payoff game algorithms [20]. The formulated optimal control problem is also related to the problems studied in [21], [22]. However, [21] studies the mean-payoff parity game, where the cost is still averaged per event not per task. The optimality measure in [22] is more similar to our setting. However, it considers a stochastic game (MDP) for the expected cost, while our work considers a non-stochastic

This work was supported by the National Natural Science Foundation of China (6217020111, 62061136004, 61803259) and the National Key Research and Development Program of China (2018AAA0101700).

Peng Lv, Xiang Yin and Shaoyuan Li are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. {lv-peng,yinxiang,syli}@sjtu.edu.cn. Yiding Ji is with Systems Hub, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, also with Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong, China. jiyiding@umich.edu.

supervisory control problem for the worst-case cost.

II. Preliminary on Optimal Supervisory Control

A. Supervisory Control Theory

Let Σ be a finite set of events. A string over Σ is a finite sequence of events; 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 $\overline{L} = \{s \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } sw \in L\}$.

We consider a DES modeled as a deterministic finite-state automata (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 [1]. 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 technical reason, we assume that $q_0 \in Q_m$.

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* $S : \mathcal{L}(G) \to \Gamma$ is a mapping that enables events dynamically based on the 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 $\mathcal{L}^\omega(S/G)$, which is defined analogously. 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 $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$.

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

B. Cost Functions

Following the standard framework of optimal supervisory control, we consider the following two types of costs:

- occurrence cost: $c_e : \Sigma \to \mathbb{N}^+$
- 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.

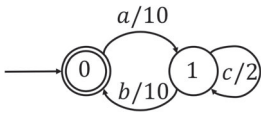

Fig. 1. System G in Example 1.

Given a supervisor S and for any string $s = \sigma_1 \dots \sigma_n \in \mathcal{L}(S/G)$, the *total cost* of s is defined by

$$\mathsf{Cost}_S(s) = \sum_{i=1,\dots,n} c_e(\sigma_i) + \sum_{s' \in \overline{\{s\}}} \sum_{\sigma \in (\Delta_G(s') \cap \Sigma_c) \setminus S(s')} c_d(\sigma),$$

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. For an infinite string $s \in \mathcal{L}^\omega(S/G)$, it does not make sense to talk about its total cost as it goes to infinity. Instead, it is of interest to consider its *average cost per event* defined by

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

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

1) *Total Cost Control for Reachability* [5]: 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 it is meaningful to discuss the total cost here as 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* [15], [17]: 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. 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.

III. Optimal Control for Cyclic Tasks

A. 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 per event is not a suitable performance measure for optimality.

Example 1: Let us consider system G shown in Figure 1, where all events are controllable and the double circle denotes the 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 0 infinitely often, while minimizing the average cost per event, a possible optimal solution is as follows: "*upon the kth occurrence of event a, the supervisor repeats control decision* $\{c\}$ *for* k*-times and then changes to control decision* $\{b\}$." Therefore, the cycle of event c at state 1 will eventually dominate the average cost per event, i.e., $\mathsf{Cost}_S^{Ave}(s)$ goes to 2. Furthermore, marked state 0 is still visited infinitely often.

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. ■

B. 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. Recall that we have assumed $q_0 \in Q_m$, which means that $I_m(s) \geq 1$. Let $s \in \mathcal{L}^\omega(S/G)$ be an infinite string. Then the *average cost per task* of s under S is

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

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

Problem 1: (OSCP-AT) Given a 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

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

IV. Game-based formulation of SCT

To solve the optimal supervisory control problem, we convert it as a two-player game over a weighted graph.

A. Two-Player Graph Games

A *game graph (or arena)* 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 $\langle \rho_i, \rho_{i+1} \rangle \in E, \forall i \geq 1$ and we say that the play starts in $\rho_1 \in V$. The set of all plays starting in v is denoted by $\mathrm{Plays}(\mathcal{A}, v)$. A *strategy* for Player $i \in \{0,1\}$ is a function

$$\theta_i : V^* V_i \to V$$

such that $\forall w \in V^*, v \in V_i : \theta_i(wv) = v' \Rightarrow \langle v, v' \rangle \in E$. We denote by Θ_i the set of all strategies for Player $i \in \{0,1\}$.

Given a strategy θ_i of Player $i \in \{0,1\}$, we say a play $\rho \in \mathrm{Plays}(\mathcal{A}, v)$ from $v \in V$ is *consistent* with strategy θ_i if $\forall n \geq 0 : \rho_n \in V_i \Rightarrow \rho_{n+1} = \theta_i(\rho_{[1,n]})$. We denote by $\mathrm{Play}(\mathcal{A}, v, \theta_i)$ as 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 $\rho(v, \boldsymbol{\theta}) = \cap_{i=0,1} \mathrm{Play}(\mathcal{A}, v, \theta_i)$.

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 $\mathrm{Play}(\mathcal{A}, v_0, \theta_i) \subseteq \mathsf{Win}$. In the paper, we focus on finding winning strategy 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: $\mathsf{Win}_S(V_m) = \{\rho \in V^\omega : \mathrm{Occ}(\rho) \cap V_m = \emptyset\}$;
- Reachability: $\mathsf{Win}_R(V_m) = \{\rho \in V^\omega : \mathrm{Occ}(\rho) \cap V_m \neq \emptyset\}$;
- Büchi: $\mathsf{Win}_B(V_m) = \{\rho \in V^\omega : \mathrm{Inf}(\rho) \cap V_m \neq \emptyset\}$.

where $\mathrm{Occ}(\rho)$ and $\mathrm{Inf}(\rho)$ 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, the reachability game and the Büchi game, respectively, in the literature. Effective algorithm has been proposed for solving each of the above games; see, e.g., [19].

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., $\mathsf{MP}(\rho) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i<n} w(\rho_i, \rho_{i+1})$. The value *secured* by strategy θ_0 of Player 0 at vertex $v \in V$ is $\mathsf{val}_{\theta_0}(v) = \sup_{\theta_1 \in \Theta_1} \mathsf{MP}(\rho(v, \theta_0, \theta_1))$, which is the worst-case payoff. The optimal value for Player 0 at vertex $v \in V$ in the game is $\mathsf{val}(v) = \inf_{\theta_0 \in \Theta_0} \sup_{\theta_1 \in \Theta_1} \mathsf{MP}(\rho(v, \theta_0, \theta_1))$. It was shown by [20] 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.

B. 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

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

where

- $V_0^G \subseteq (Q \times \Sigma) \cup \{(q_0, \epsilon)\} \cup \{v_{D,0}\}$ is the set of Player 0's vertices;
- $V_1^G \subseteq (Q \times \Gamma) \cup \{v_{D,1}\}$ is the set of Player 1's vertices;

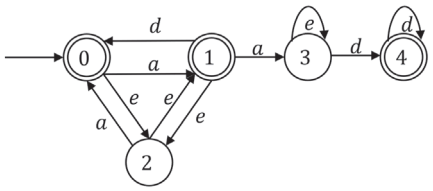

Fig. 2. DES G for Example 2.

- $E \subseteq (V_0^G \times V_1^G) \cup (V_1^G \times V_0^G)$ is the set of edges defined by:
  - for any $(q,\sigma) \in V_0^G$ and $\gamma \in \Gamma$, we have
$$\langle (q,\sigma),(q,\gamma)\rangle \in E$$
  - for any $(q,\gamma) \in V_1^G$, if $\Delta_G(q) \cap \gamma \neq \emptyset$, then for any $\sigma \in \Delta_G(q) \cap \gamma$, we have
$$\langle (q,\gamma),(\delta(q,\sigma),\sigma)\rangle \in E$$
    otherwise, when $\Delta_G(q) \cap \gamma = \emptyset$, we have
$$\langle (q,\gamma), v_{D,0}\rangle \in E$$
  - for $v_{D,0} \in V_0^G$ and $v_{D,1} \in V_1^G$, we have
$$\langle v_{D,0}, v_{D,1}\rangle, \langle v_{D,1}, v_{D,0}\rangle \in E$$
- $v_0^G = (q_0,\epsilon)$ is the initial vertex.

Intuitively, game graph $\mathcal{A}^G$ distinguishes explicitly between the supervisor's decision stage V_0^G from which a control pattern is chosen and the environment's decision stage V_1^G from which an event occurs. Furthermore, each state in V_0^G is of form (q,σ), where q represents its current state in the plant and $\sigma \in \Sigma$ represents the latest event leading to this state. Similarly, each state in V_1^G is of form (q,γ), where q is still the current state, while $\gamma \in \Gamma$ represents the current control pattern applied. Note that (q,γ) can only move to $(\delta(q,\sigma),\sigma)$ for σ that is enabled and feasible, i.e., $\sigma \in \Delta_G(q) \cap \gamma$. For the case that $\Delta_G(q) \cap \gamma = \emptyset$, we introduce two new vertices $v_{D,0} \in V_0^G$ and $v_{D,1} \in V_1^G$, where "D" represents "deadlock". Therefore, the graph has at least an 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

$$V_m = \{(q,\sigma) \in V_0 : q \in Q_m\}$$

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: for any $(q,\sigma) \in V_0$ and $(q',\gamma) \in V_1$,

- $w((q',\gamma),(q,\sigma)) = c_e(\sigma)$; and
- $w(((q,\sigma),(q,\gamma)) = \sum_{\sigma \in \Delta_G(q) \setminus \gamma} c_d(\sigma)$

and $w((q,\gamma),v_{D,0}) = w(v_{D,0},v_{D,1}) = w(v_{D,1},v_{D,0}) = \infty$.

Example 2: Consider system G shown in Figure 2, where $\Sigma_c = \{a\}$ and $Q_m = \{0,1,4\}$. 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 Figure 3, where we use circles and squares to denote Player 0's vertices and Player 1's vertices respectively. For the sake of simplicity, at each state, we only consider control patterns in which all disabled events are feasible. For example, in state 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. ■

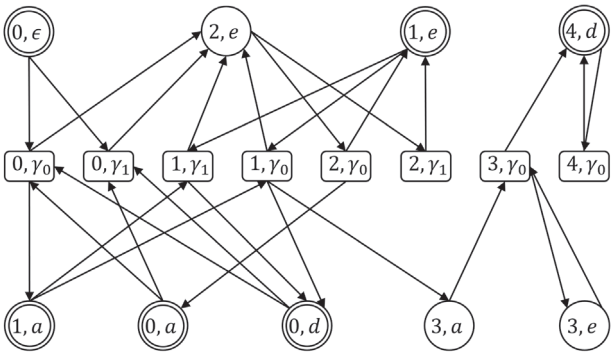

Fig. 3. The SC game graph $\mathcal{A}$, where $\gamma_0 = \{a,d,e\}$ and $\gamma_1 = \{d,e\}$.

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 V_m in ρ. Then the *mean payoff per task* of a play $\rho \in V^\omega$ is

$$\overline{\mathsf{MP}}(\rho) = \limsup_{n\to\infty} \frac{\mathsf{Cost}(\rho_{[1,n]})}{N_m(\rho_{[1,n]})},$$

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

$$\overline{\mathsf{val}}_{\theta_0}(v) = \sup_{\theta_1 \in \Theta_1} \overline{\mathsf{MP}}(\rho(v,\theta_0,\theta_1))$$

We consider a game in which Player 0 not only aims to minimize the mean payoff per task, but also needs to fulfill the Büchi winning condition w.r.t. V_m so that target vertices are visited infinitely often. 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 V_m and weight function $w : V \to \mathbb{N} \cup \{\infty\}$, find an optimal strategy $\theta_0^* \in \Theta_0$ for Player 0 such that

(1) $\mathrm{Play}(\mathcal{A},v_0,\theta_0^*) \subseteq \mathsf{Win}_B(V_m)$;

(2) for any strategy $\theta_0' \in \Theta_0$ satisfying (1), we have $\overline{\mathsf{val}}_{\theta_0^*}(v_0) \leq \overline{\mathsf{val}}_{\theta_0'}(v_0)$.

Note that, we do not require $\overline{\mathsf{val}}_{\theta_0}(v_0) < \infty$ additionally here, because according to the definition of the weight function, if $\mathrm{Play}(\mathcal{A},v_0,\theta_0) \subseteq \mathsf{Win}_B(V_m)$, then $\overline{\mathsf{val}}_{\theta_0}(v_0) < \infty$; otherwise V_m will not be visited infinitely often.

C. 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 a strategy θ_0, it induces a supervisor, denoted by S_{θ_0}, inductively by: for any play $\rho =$

$(q_0,\epsilon)(q_0,\gamma_0)(q_1,\sigma_1)(q_1,\gamma_1)\dots(q_n,\sigma_n)(q_n,\gamma_n) \in \text{Play}(\mathcal{A},(q_0,\epsilon),\theta_0)$, we have $S_{\theta_0}(\sigma_1\dots\sigma_n)=\gamma_n$.

- Given a 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 $\rho=(q_0,\epsilon)(q_0,\gamma_0)(q_1,\sigma_1)(q_1,\gamma_1)\dots(q_n,\sigma_n) \in \text{Play}(\mathcal{A},(q_0,\epsilon),\theta_S)$.

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 $\mathsf{Cost}^{AveT}_{S^*}(s)<\infty$. This is because, if the system is 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 Figure 1, the system itself is already live and non-blocking, but its cost per task is infinite due to the cycle at state 1. Therefore, it does not satisfy the Büchi winning condition.

The following theorem showsthat, 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.

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

V. Synthesis Procedure

In the section, we present our main synthesis procedure for solving Problem 2 (and hence solves Problem 1). Our approach consists of two parts. Firstly, we consider the decision process within each task cycle and obtain micro-strategies. Then we abstract the decision process among different tasks in order to obtain a macro-strategy that determines which micro-strategy to play at each stage.

A. Single Task Strategy Graph

Recall that a *path* in a directed graph $\mathcal{G}=(V,E)$ is a sequence of vertices $v_1v_2\dots v_n$ such that $\langle v_i,v_{i+1}\rangle \in E, \forall i \geq 1$. Then a graph is said to be *acyclic* if does not exist a path $v_1v_2\cdots v_nv_1$ that starts from and ends up with the same vertex. For each vertex $v\in V$, we define $\text{Succ}_{\mathcal{G}}(v)=\{v'\in V:\langle v,v'\rangle\in E\}$ as the set of successor vertices of v.

Note that due to the presence of uncontrollable events in G or the presence of adversary player in $\mathcal{A}$, at each marked state, we cannot determine the next marked state to visit for sure. Instead, we can only have a *micro-strategy* rather than a single path for each task cycle and need to consider all accessible marked states. Such an issue is captured by the structure of *single task strategy graph* defined as follows.

Definition 1: Let $\mathcal{A}=(V=V_0\cup V_1,E,v_0)$ be the SC game graph constructed from G and $r\in V_m$ be a target vertex in $\mathcal{A}$. Then a *single task strategy graph* (STSG) rooted at vertex r is an acyclic graph

$$\mathcal{T}=(V_{\mathcal{T}},E_{\mathcal{T}},\hat{r}),$$

where

- the root $\hat{r}\notin V$ is a new copy vertex of $r\in V_m$;
- $V_{\mathcal{T}}\subseteq V\cup\{\hat{r}\}$ is the set of vertices and we define $V_{\mathcal{T},0}=(V_0\cap V_{\mathcal{T}})\cup\{\hat{r}\}, V_{\mathcal{T},1}=V_1\cap V_{\mathcal{T}}$ and $V_{\mathcal{T},m}=V_m\cap V_{\mathcal{T}}$;
- $E_{\mathcal{T}}\subseteq V_{\mathcal{T}}\times(V_{\mathcal{T}}\setminus\{\hat{r}\})$ is the set of edges satisfying the following constraints:
  - $\forall v\in V_{\mathcal{T},0}\setminus V_m: |\text{Succ}_{\mathcal{T}}(v)|=1$;
  - $\forall v\in V_{\mathcal{T},1}: |\text{Succ}_{\mathcal{T}}(v)|=|\text{Succ}_{\mathcal{A}}(v)|$;
  - $\forall v\in V_{\mathcal{T},m}: |\text{Succ}_{\mathcal{T}}(v)|=0$;
  - $\forall\langle v,v'\rangle\in E_{\mathcal{T}}: \langle P(v),v'\rangle\in E$, where $P:V_{\mathcal{T}}\to V$ simply removes "hat" for each vertex, i.e., it maps $\hat{r}$ to r and does nothing for other vertices.

We note that a STSG starts from $\hat{r}$ and terminates at vertices $V_{\mathcal{T},m}$. Therefore, $V_{\mathcal{T},m}$ are also referred to as the *termination vertices* of $\mathcal{T}$. We denote by $\mathbb{T}$ the set of all STSGs and denote by $\mathbb{T}(r,S)\subseteq\mathbb{T}$ as the set of all STSGs that are rooted at r and terminate at $S\subseteq V_m$. We also define $\mathbb{T}(r)=\cup_{S\subseteq V_m}\mathbb{T}(r,S)$ as the set of STSGs rooted at r.

Intuitively, a STSG $\mathcal{T}\in\mathbb{T}(r,S)$ represents a micro-strategy for the decision process from a target state r until the next target state is reached. Therefore, each vertex of Player 0 has a unique successor state representing the choice of the strategy, i.e., $|\text{Succ}_{\mathcal{T}}(v)|=1$. For each vertex of Player 1, we need to consider all possible choices of Player 1 that are feasible in the system, i.e., $|\text{Succ}_{\mathcal{T}}(v)|=|\text{Succ}_{\mathcal{A}}(v)|$. Furthermore, since we are considering the decision process for a single task cycle, the graph terminates whenever a target state in S is reached. Note that, although the micro-strategy is played from $r\in V_m$, the graph is actually rooted at its copy $\hat{r}$ because it may go back to r to complete the task cycle. Therefore, we use a copy $\hat{r}$ to distinguish the possible state $r\in S$ from which the next task cycle starts.

In words, $\mathcal{T}\in\mathbb{T}(r,S)$ represents a micro-strategy guaranteeing that the next accessible target vertices are in S, and all vertices in S are possibly to be visited under this strategy. With this understanding, we denote by $\theta_{\mathcal{T}}$ the micro-strategy of Player 0 that is effective from vertex r until a new target vertex $v\in S$ is reached. However, it cannot determine which vertex in S will be reached; this depends on the strategy of Player 1 and we need to handle all possibilities.

Example 3: Again, let us consider the SC game graph $\mathcal{A}$ shown in Figure 3. Then Figure 4 shows three STSGs for $\mathcal{A}$. For example, we have $\mathcal{T}_1\in\mathbb{T}((0,a),\{(1,e),(0,a),(1,a)\})$ as we cannot determine which target vertex to visit but we can guarantee that by playing micro-strategy $\theta_{\mathcal{T}_1}$, the next task cycle must start from either $(1,e),(0,a)$ or $(1,a)$. Also, we have $\mathcal{T}_3\in\mathbb{T}((0,a),\{(4,d)\})$ because the choice of Player 1 is unique. ■

Remark 1: In the above definition, we require that a STSG $\mathcal{T}$ is *acyclic*, which is motivated by the following observations. Firstly, if $\mathcal{T}$ forms a cycle before reaching a target state, then this means that, by playing this micro-strategy from the root, Player 1 will have a strategy such

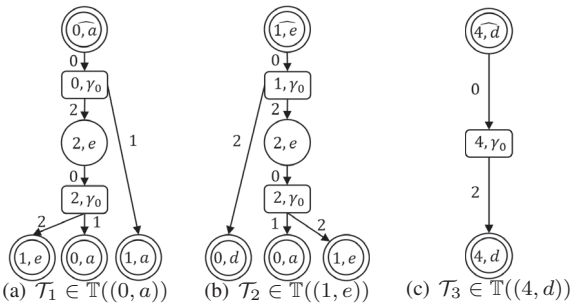

Fig. 4. Examples of STSGs.

that to loop forever in the cycle. Therefore, Player 0 cannot guarantee to reach a target vertex and the micro-strategy is invalid. Another reason for requiring $\mathcal{T}$ to be acyclic is that, positional strategy is known to be sufficient for optimality for the purpose of reachability. Therefore, looping within a cycle without target state for a finite number of times does not gain anything for completing the task. Therefore, the acyclicity condition essentially also restricts each micro-strategy to be a positional strategy, which is without loss of generality.

Since $\mathcal{T} \in \mathbb{T}(r, S)$ is acyclic, there are only a finite number of paths from the root vertex $\hat{r}$ to each termination vertex $s \in S$. We define $\text{Path}_{\mathcal{T}}(\hat{r}, s)$ as the set of all paths from $\hat{r}$ to $s \in S$ in $\mathcal{T} \in \mathbb{T}(r, S)$. Note that, although Player 0 cannot determine which termination vertex it will reach at the end of this task cycle, the costs incurred when reaching different termination vertices are different, which depends on the strategies of Player 1. Therefore, for any $\mathcal{T} \in \mathbb{T}(r, S)$, we define a new partial weight function $w_{\mathcal{T}} : S \to \mathbb{N}^+$ that assigns each termination vertex $s \in S$ the worst weight of the path from $\hat{r}$ to s, i.e.,

$$w_{\mathcal{T}}(s) = \max_{\rho \in \text{Path}_{\mathcal{T}}(\hat{r},s)} \sum_{i<|\rho|} w(P(\rho_i), \rho_{i+1}).$$

For example, for $\mathcal{T}_1$ shown in Figure 4(a), we have $\omega_{\mathcal{T}_1}((1,e)) = 4$, $\omega_{\mathcal{T}_2}((0,a)) = 3$ and $\omega_{\mathcal{T}_3}((4,d)) = 2$.

B. Macro SC Game Graph

In the previous subsection, we have shown how to represent a micro-strategy within each task cycle as a STSG. However, the question is: suppose that we have completed the previous task cycle and is at a target vertex, what is the micro-strategy we need to play for the next cycle? The question is highly non-trivial because: (i) the resulting target state under a strategy is non-deterministic; and (ii) the optimality of the target states, from which the next cycle start, again depends on all possible micro-strategies that can be played from them. In order to resolve the above issues, our approach is to abstract all possible connections between each target vertex under different STSGs as a *macro-game*. Based on the macro-game, we synthesize a *macro-strategy* that determines which micro-strategy to play from each target vertex. This idea is formalized as follows.

Definition 2: Let $\mathcal{A}$ be the SC game graph and $\mathbb{T}$ be the set of all STSGs. Then the *macro SC game graph* (MSCGG) is a game arena

$$\mathcal{R} = (V_{\mathcal{R}} = V_{\mathcal{R},0} \cup V_{\mathcal{R},1}, E_{\mathcal{R}}, v_{\mathcal{R},0})$$

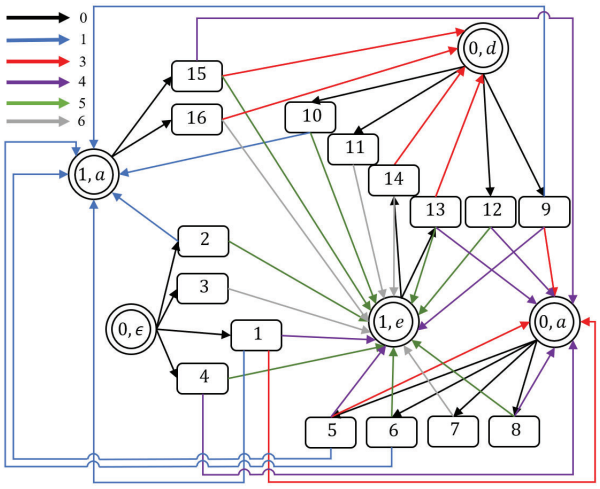

Fig. 5. The MSCGG $\mathcal{R}$.

where

- $V_{\mathcal{R},0} = V_m$ and $V_{\mathcal{R},1} = \mathbb{T}$ are vertices;
- $E \subseteq (V_{\mathcal{R},0} \times V_{\mathcal{R},1}) \cup (V_{\mathcal{R},1} \times V_{\mathcal{R},0})$ is the set of edges defined by:
  - $\forall r \in V_m, \mathcal{T} \in \mathbb{T}(r) : \langle r, \mathcal{T} \rangle \in E_{\mathcal{R}}$
  - $\forall \mathcal{T} \in V_{\mathcal{R},1}, s \in V_{\mathcal{T},m} : \langle \mathcal{T}, s \rangle \in E_{\mathcal{R}}$
- $v_{\mathcal{R},0} = v_0 \in V_m$.

Furthermore, we associate $\mathcal{R}$ a weight function $w_{\mathcal{R}} : E_{\mathcal{R}} \to \mathbb{N}^+$ by $w_{\mathcal{R}}(\langle r, \mathcal{T} \rangle) = 0$ and $w_{\mathcal{R}}(\langle \mathcal{T}, s \rangle) = w_{\mathcal{T}}(r, s)$.

Essentially, $\mathcal{R}$ abstracts the cyclic process between different target vertices; hence, Player 0's vertices are V_m. At each $r \in V_m$, Player 0 chooses to play a micro-strategy represented by a STSG $\mathcal{T}$ rooted at r and moves to an intermediate vertex $\mathcal{T}$. Once Player 0 agrees with the micro-strategy induced by $\mathcal{T}$, it can only make sure that the next target state reached will be in $V_{\mathcal{T},m}$ but cannot choose which one; this is the role of Player 1. Therefore, each $\mathcal{T}$ is a Player 1's vertex and it will lead to all possible target states $V_{\mathcal{T},m}$. The cost incurs when a specific target state $s \in V_{\mathcal{T},m}$ is reached. The definition of $w_{\mathcal{T}}(r, s)$ captures the worst-case cost from r to s under micro-strategy induced by $\mathcal{T}$. Therefore, we assign edge $\langle \mathcal{T}, s \rangle$ the same cost as $w_{\mathcal{T}}(r, s)$.

Example 4: We still consider our running example whose SC game graph $\mathcal{A}$ is shown in Figure 3. Then its macro SC game graph is in Figure 5, where double circles and squares represent Player 0's vertices and Player 1's vertices, respectively. For the sake of simplicity, STSGs are not shown explicitly and are denoted by numbers. For example, states 5 and 13 in the figure correspond to STSGs $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 4, respectively. Also for the sake of simplicity, different costs for edges from $\mathcal{T}$ to $r \in V_{\mathcal{T},m}$ are depicted by different colors, whose values are given at the upper left corner of the graph. ■

C. Synthesis Algorithm

The above discussion suggests an approach for solving the mean-payoff Büchi game per task as formulated in Problem 2. In particular, since the worst-case total cost under each micro-strategy from a root target vertex to another specific target vertex has already been abstracted as a single

cost on an edge in $\mathcal{R}$, we have actually transferred the per cycle cost criterion to the per edge criterion in the standard mean-payoff game. Therefore, it suffices to solve a standard mean payoff game over the macro-graph $\mathcal{R}$, which gives us a macro-strategy determining which $\mathcal{T} \in \mathbb{T}(r)$ to play at each $r \in V_m$. We denote by θ^*_{MP} as the optimal strategy for Player 0 in mean payoff game on $(\mathcal{R}, w_{\mathcal{R}})$, which can be obtained by using existing algorithms, e.g., [20], [23]. Furthermore due to the structural property of mean payoff game, such an optimal strategy is positional in the sense it only depends on the current vertex of Player 0 in $\mathcal{R}$. Then for each $r \in V_m$, we denote by $\theta^*_{MP}(r) \in \mathbb{T}(r)$ the STSG θ^*_{MP} chooses to go. As we discussed early, $\theta^*_{MP}(r) = \mathcal{T}$ again induces a micro-strategy $\theta_{\mathcal{T}}$. It suggests a way to induce a strategy, based on θ^*_{MP}, for the original game $\mathcal{A}$ as follows:

- Initially, the system is at state $v_0 \in V_m$, and Player 0 plays micro-strategy $\theta^*_{MP}(v_0)$, until it reaches a new target vertex $r \in V_m$;
- Then upon reaching target vertex $r \in V_{\mathcal{T},m}$, Player 0 changes its strategy to micro-strategy $\theta^*_{MP}(r)$, until it reaches a new target vertex;
- Repeat the above process indefinitely.

We denote by θ^*_0 the strategy on $\mathcal{A}$ which is induced by θ^*_{MP} based on the above. Essentially, θ^*_0 consists of a sequence of micro-strategies $\mathcal{T}_1, \mathcal{T}_2, \dots$ determined by macro-strategy θ^*_{MP}. According to Section IV-C, strategy θ^*_0 can again induce a supervisor $S_{\theta^*_0}$ that controls the original plant G.

Theorem 2: Let strategy θ^*_{MP} be the solution to the mean payoff game on $\mathcal{R}$, then its induced strategy θ^*_0 solves Problem 2, which further implies that S^* solves Problem 1.

We illustrate the entire procedure by the follow example.

Example 5: Let us still consider our running example, where the macro SC game graph $\mathcal{R}$ has been shown in Figure 5. Then, by solving the standard mean payoff game on $\mathcal{R}$, we can get the optimal stationary strategy θ^*_{MP} for Player 0 in $\mathcal{R}$, which works as follows: $\theta^*_{MP}(0, \epsilon) = 1, \theta^*_{MP}(0, a) = 5, \theta^*_{MP}(0, d) = 9, \theta^*_{MP}(1, a)15$ and $\theta^*_{MP}(1, e) = 13$. Then, we can induce θ^*_0 for Player 0 in $\mathcal{A}$ from θ^*_{MP}, which works as follows: $\theta^*_0(0, \epsilon) = (0, \gamma_0), \theta^*_0(0, a) = (0, \gamma_0), \theta^*_0(0, d) = (0, \gamma_0), \theta^*_0(1, a) = (1, \gamma_1), \theta^*_0(1, e) = (1, \gamma_1)$ and $\theta^*_0(2, e) = (2, \gamma_0)$. Finally, we obtain supervisor S^* for G based on θ^*_0, which works as follows:

- $S(s) = \gamma_0 = \{a, d, e\}$, when $\delta(q_0, s) \neq 1$;
- $S(s) = \gamma_1 = \{d, e\}$, when $\delta(q_0, s) = 1$. ■

VI. Conclusion

In this paper, we formulated and solved a new type of optimal supervisory control problem for discrete-event systems. Our setting captures the scenario where tasks, modeled by marked states, needs to be completely indefinitely in a cyclic manner. Our main contributions are summarized as follows. Firstly, we introduced a new optimality measure called the average cost per task, which is much more suitable when cyclic tasks are involved in infinite behaviors. Secondly, we provided a game-theoretical approach for solving the formulated problem. Specifically, our approach is based on abstracting the strategy for each task cycle as a micro-game and the overall strategy consists of both the macro-strategy and the micro-strategy. Our results further extend the theory of optimal supervisory control of DES.

References

[1] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[2] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially-observed discrete-event systems," *IEEE rans. Automatic Control*, vol. 61, no. 5, pp. 1239–1254, 2016.

[3] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Trans. Automatic Control*, vol. 61, no. 8, pp. 2140–2154, 2016.

[4] R. Kumar and V. K. Garg, "Optimal supervisory control of discrete event dynamical systems," *SIAM Journal on Control and Optimization*, vol. 33, no. 2, pp. 419–439, 1995.

[5] R. Sengupta and S. Lafortune, "An optimal control theory for discrete event systems," *SIAM Journal on control and Optimization*, vol. 36, no. 2, pp. 488–541, 1998.

[6] J. Fu, A. Ray, and C. M. Lagoa, "Unconstrained optimal control of regular languages," *Automatica*, vol. 40, no. 4, pp. 639–646, 2004.

[7] R. Su, J. H. Van Schuppen, and J. E. Rooda, "The synthesis of time optimal supervisors by using heaps-of-pieces," *IEEE Trans. Automatic Control*, vol. 57, no. 1, pp. 105–118, 2011.

[8] R. C. Hill and S. Lafortune, "Planning under abstraction within a supervisory control context," in *55th IEEE Conference on Decision and Control (CDC)*, pp. 4770–4777, 2016.

[9] S. Ware and R. Su, "Time optimal synthesis based upon sequential abstraction and its application to cluster tools," *IEEE Trans. Automation Science and Engineering*, vol. 14, no. 2, pp. 772–784, 2016.

[10] A. Sakakibara and T. Ushio, "On-line permissive supervisory control of discrete event systems for scltl specifications," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 530–535, 2020.

[11] Z. Ma and J. Zhang, "Determining optimal control sequences for reconfiguration in petri nets using cost trees," in *59th IEEE Conference on Decision and Control (CDC)*, pp. 4485–4491, 2020.

[12] H. Marchand, O. Boivineau, and S. Lafortune, "On the synthesis of optimal schedulers in discrete event control problems with multiple goals," *SIAM Journal on Control and Optimization*, vol. 39, no. 2, pp. 512–532, 2000.

[13] H. Marchand, O. Boivineau, and S. Lafortune, "On optimal control of a class of partially observed discrete event systems," *Automatica*, vol. 38, no. 11, pp. 1935–1943, 2002.

[14] V. Pantelic and M. Lawford, "Optimal supervisory control of probabilistic discrete event systems," *IEEE Trans. Automatic Control*, vol. 57, no. 5, pp. 1110–1124, 2011.

[15] S. Pruekprasert and T. Ushio, "Optimal stabilizing controller for the region of weak attraction under the influence of disturbances," *IEICE Trans. Information and Systems*, vol. 99, no. 6, pp. 1428–1435, 2016.

[16] Y. Ji, X. Yin, and S. Lafortune, "Local mean payoff supervisory control for discrete event systems," *IEEE Tran. Automatic Control*, 2021.

[17] Y. Ji, X. Yin, and S. Lafortune, "Optimal supervisory control with mean payoff objectives and under partial observation," *Automatica*, vol. 123, p. 109359, 2021.

[18] K. W. Schmidt, "Optimal supervisory control of discrete event systems: cyclicity and interleaving of tasks," *SIAM Journal on Control and Optimization*, vol. 53, no. 3, pp. 1425–1439, 2015.

[19] E. Gradel and W. Thomas, *Automata, Kogics, and Infinite Games: A Guide to Current Research*. Springer Science & Business Media, 2002.

[20] U. Zwick and M. Paterson, "The complexity of mean payoff games on graphs," *Theoretical Computer Science*, vol. 158, no. 1-2, pp. 343–359, 1996.

[21] K. Chatterjee, T. A. Henzinger, and M. Jurdzinski, "Mean-payoff parity games," in *20th Annual IEEE Symposium on Logic in Computer Science*, pp. 178–187, 2005.

[22] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Trans. Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[23] L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin, "Faster algorithms for mean-payoff games," *Formal Methods in System Design*, vol. 38, no. 2, pp. 97–118, 2011.