Automatica 123 (2021) 109359

Contents lists available at ScienceDirect

Automatica

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

Optimal supervisory control with mean payoff objectives and under partial observation ☆

Yiding Ji [a], Xiang Yin [b,*], Stéphane Lafortune [a]

[a] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA
[b] Department of Automation, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Article history:
Received 14 June 2019
Received in revised form 19 May 2020
Accepted 13 October 2020
Available online 20 November 2020

Keywords:
Discrete event systems
Supervisory control
Partial observation
Optimal control
Algorithmic game theory

ABSTRACT

We investigate optimal mean payoff supervisory control problems on partially observed discrete event systems modeled as weighted finite-state automata. The event weights capture variations of a given resource (i.e., energy) expended or replenished during the operation of the system and the mean payoff is then defined as the average of the accumulative event weights. Two supervisory control problems are considered in this work. For the first, the system is equipped with a fixed amount of initial energy to support its operation and the supervised system should always have a nonnegative energy level. For the second, the limit mean payoff of any event sequence should never drop below zero in the supervised system. We further optimize the worst case limit mean payoff of infinite event sequences under both scenarios. The two problems are solved sequentially. In order to capture information on both the state estimate and the energy level of the system, we define energy information states which incorporate sufficient information for the decision making of the supervisor. Then we propose the First Cycle Energy Inclusive Controller (FCEIC) and further transfer the supervisory control problems into two-player games with properly defined objectives on the FCEIC. Finally, we perform a min–max search on the game graphs to synthesize the optimal supervisors for both scenarios.

1. Introduction

Supervisory control has been thoroughly studied under the framework of discrete event systems (DES). The supervisor restricts the behavior of the plant (system) by enabling and disabling events, so that the given specification is achieved. Supervisory control has been thoroughly discussed under various DES models from different perspectives (Cassandras & Lafortune, 2008; Wonham & Cai, 2019).

In the context of DES, due to the limited sensing capabilities and measurement noises, the plant is usually partially observed, which gives rise to supervisory control under partial observation. Many works fall into this category, see, e.g., Alves, Carvalho, and Basilio (2016), Alves, da Cunha, Carvalho, Moreira, and Basilio (2019), Cai, Zhang, and Wonham (2015), Giua, Seatzu, and Basile (2004), Gu, Wang, Li, and Wu (2018), Komenda and Masopust (2017), Lin, Masopust, Wonham, and Su (2019), Schmidt and Breindl (2014), Shu and Lin (2015, 2017), Takai and Ushio (2003) and Yin and Lafortune (2017). Recently, a novel approach was developed in Yin and Lafortune (2016a) to synthesize maximally permissive partial-observation supervisors without assumptions on the relationship between controllable and observable events. It was then extended to a uniform approach in Yin and Lafortune (2016b) for the enforcing a series of qualitative properties in DES.

In addition to logical properties, supervisory control has also been investigated under some quantitative performance measures. Optimal supervisory control is one problem of particular interest, where different frameworks have been developed. For example, Sengupta and Lafortune (1998) defined both event enablement and disablement costs, then found the controller with minimum total costs to reach the designated states. This framework was extended in Marchand, Boivineau, and Lafortune (2002) and Pruekprasert and Ushio (2016b) to consider partial observation of the system. Furthermore, Pruekprasert, Ushio, and Kanazawa (2016) solved an infinite horizon optimal supervisory control problem under the framework of mean payoff games with perfect information. A closely related problem of optimal stabilization by supervisory control was investigated in Han, Chen, and Su (2019) and Pruekprasert and Ushio (2016a, 2016b).

☆ Research supported in part by the US National Science Foundation, China under grant CNS-1738103, also by National Natural Science Foundation of China under grants 61803259, 61833012. The material in this paper was partially presented at the 57th IEEE Conference on Decision and Control, December 17–19, 2018, Miami Beach, Florida, USA. This paper was recommended for publication in revised form by Associate Editor Christoforos Hadjicostis under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: jiyiding@umich.edu (Y. Ji), yinxiang@sjtu.edu.cn (X. Yin), stephane@umich.edu (S. Lafortune).

https://doi.org/10.1016/j.automatica.2020.109359
0005-1098/

Along with the deterministic setting, optimal supervisory control in probabilistic DES was studied in Pantelic and Lawford (2012).

Motivation In many engineering applications, the system may generate or consume some resources over a relatively long time horizon and it is often essential to maintain a reasonable rate of resource generation/consumption. Consider the power management system for hybrid electric vehicles (Malikopoulos, 2014). A positive or negative torque is demanded from the powertrain depending on the driving mode, e.g., cruising or braking. The power from the electric machine is regulated by tuning the torque so that the torque complies with the driving mode. The electric machine generates power by consuming electrical energy from the battery in the motor mode, and it absorbs power from the driveline to charge the battery in the generator mode. When the vehicle is cruising, the engine should consistently provide sufficient power so that the vehicle moves smoothly. Here the supervisory control scheme may be applied to determine the power flow over a long time range when the vehicle is on the road. Note that the supervisor's observations may be compromised by measurement uncertainty or noise.

Contributions The above situation inspires us to investigate infinite horizon optimal supervisory control under partial observation, which has never been investigated in DES before to the best of our knowledge. We term the resource associated with the system as *energy*, which is a generic term. The system is modeled as a weighted automaton and the limit average weight characterizes the rate of energy generation/consumption, which is to be optimized. Specifically, we consider two cases where the supervisor entails an optimal limit mean payoff. The first is that the system is granted with certain amount of initial energy to support its operation and the energy level should never drop below 0. The second is that the limit mean payoff should always be above a given threshold. Correspondingly, we formulate two supervisory control problems and solve them in sequence.

In the first phase, *energy information states* are defined to incorporate information on state estimates and energy level of the system. Next we transfer the supervisory control problems into two-player games between the supervisor and the "environment" (system) on the First Cycle Energy Inclusive Controller (FCEIC). By construction, the supervisor's winning strategies in the FCEIC achieve a nonnegative energy level or a sufficiently large limit mean payoff. In the second phase, the optimal control strategies are synthesized by solving a minimax game on a substructure of the FCEIC. Those strategies in turn solve our proposed control problems after minor manipulation.

Related works Our solution methodology is inspired by the literature on infinite horizon optimal/stochastic control and algorithmic games in computer science. Here we briefly highlight our novelty compared with existing research from both fields.

Infinite horizon optimal/stochastic control under partial observation has long been a challenging problem (Bertsekas, 2012; Krishnamurthy, 2016). Optimal policy existence problem for infinite horizon partially observable Markov Decision Processes (MDPs) is generally undecidable, either with discounted or average reward objectives (Madani, Hanks, & Condon, 2003). To solve the infinite horizon optimal supervisory control problems, we make some necessary assumptions on the system (plant) and solve the problems under two-player quantitative games. It turns out that we may solve the game to synthesize supervisors by only focusing on the "first" simple cycles since the mean payoff game is a type of first-cycle games (Aminof & Rubin, 2017), so our game-theoretic technique is significantly different from the existing methods to solve optimal control and MDPs (Bertsekas, 2012; Puterman, 2005), such as value/policy iteration or simulation/approximation methods. To synthesize the optimal supervisor on the FCEIC, we perform a min–max search which is similar to the *minimax criterion* in optimal control theory (Başar & Bernhard, 2008). This is consistent with our problem formulation where we optimize the worst limit mean payoff. However, we cannot directly apply the minimax criterion as our problem is discussed under partial observation. Instead, we propose the Energy Inter Connected System to "retrieve" the unobservable strings in the FCEIC. So the minimum/maximum payoffs for both players are correctly evaluated before the optimal control strategy is determined by a min–max search.

Our supervisory control framework is also in contrast with algorithmic game theory for reactive synthesis (Apt & Grädel, 2011; Baier & Katoen, 2008). First, there is a *plant*, i.e., a system to be controlled, and a separate supervisor (controller) in our work. Additionally, the supervised system is *closed-loop* in the sense that the "input" to the supervisor is the set of strings generated by the system so far and the "output" of the supervisor is a control decision to inform the system what events are allowed to occur. Furthermore, the supervisor may allow multiple events to occur simultaneously, then the system decides what event to execute next. This mechanism is similar to the so-called *multi-strategy* in algorithmic games (Apt & Grädel, 2011), under which one player may choose more than one outgoing edges at its position. In general, the supervisor may only have limited control and observation capabilities, i.e., some events of the system can never be disabled and some events are not observed by the supervisor. Those limitations are usually not characterized in algorithmic games for reactive synthesis. The above mentioned differences impose additional difficulties on directly applying existing results of quantitative algorithmic games to solve the supervisory control problem in our work, thus special techniques are necessary.

Specifically, this work leverages some results from mean payoff games where the first player maximizes the limit average payoffs (weights) of traversed edges while the second player minimizes them. Well structured solutions were proposed for the *perfect information* mean payoff game (Zwick & Paterson, 1996), where both players know the complete history of the game up to their current positions. The more challenging case is mean payoff games with *imperfect information* where one player does not know the exact state or actions of its opponent. Such games are in general undecidable (Hunter, Pauly, Pérez, & Raskin, 2018). Briefly speaking, the undecidability is due to the presence of indefinite cycles with total payoffs of different signs. The game graph is unfolded to determine the winner of the game. However, the unfolding is never halted so no player is able to claim winning the game. Some decidable classes were presented in Hunter et al. (2018), which put some restrictions to eliminate the indefinite cycles. These results motivate our problem settings.

Our work is not the first to investigate problems in DES by leveraging results from algorithmic game theory, see, e.g., Ji, Yin, and Lafortune (2019a), Pruekprasert and Ushio (2017), Pruekprasert et al. (2016), Yin and Lafortune (2016a) and Yin and Lafortune (2016b). However, both Yin and Lafortune (2016a) and Yin and Lafortune (2016b) focused on supervisory control for qualitative properties and Pruekprasert et al. (2016) discussed optimal mean payoff supervisory control under full observation. In contrast to this work as well, Pruekprasert and Ushio (2017) studied supervisory control under fixed-initial-credit energy games and a more recent work (Ji, Yin, & Lafortune, 2019b) studied supervisory control under local mean payoff constraints, defined over a finite number of events. Finally, Ji et al. (2019a) discussed a different problem, namely opacity enforcement under energy constraints.

Organization The following sections are organized as follows. Section 2 describes the system model. In Section 3, we formulate two optimal mean payoff supervisory control problems under partial observation. Section 4 introduces energy information states and the First Cycle Energy Inclusive Controller (FCEIC)

as the game graph for each problem. In Section 5, we analyze some relevant properties of the FCEIC and partially solve the two proposed problems. Then in Section 6, we completely solve the two problems by finding the optimal solution from the partial solutions obtained in Section 5. Finally, Section 7 concludes the paper and raises potential directions for future work.

A preliminary version of this work appears in Ji, Yin, and Lafortune (2018) with partial results. The major improvements of this work are two-fold. First, we consider mean payoff supervisory control under constraints imposed by the system's energy capacity, i.e., Problem 1 in Section 3, which is not discussed in Ji et al. (2018). Second, we further investigate the optimal control of worst-case limit mean payoffs in Section 6, which is not treated in Ji et al. (2018) either.

2. System model

The system is modeled as a weighted finite-state automaton:

$$G = (X, E, f, x_0, \omega)$$

where X is the finite state space, E is the finite set of events, $f : X \times E \to X$ is the partial transition function, $x_0 \in X$ is the initial state, $\omega : E \to \mathbb{Z}$ is the weight function that assigns an integer to each event. We view the event's weight as its *energy payoff*. A positive number stands for energy gain while a negative number stands for energy cost. The transition function is extended to $X \times E^*$ in the standard manner and we still denote the extended function by f. The language generated by G is defined as $\mathcal{L}(G) = \{s \in E^* : f(x_0, s)!\}$ where ! means "is defined". We denote by $s \leq u$ if string s is a prefix of u, and $s < u$ if $s \leq u, s \neq u$. The function ω is additive and its domain can be extended to E^* by letting $\omega(\epsilon) = 0$, $\omega(se_o) = \omega(s) + \omega(e_o)$ for all $s \in E^*$ and $e \in E$. The *(accumulative) payoff* of $s in \mathcal{L}(G)$ is the sum of each event's weight in s, i.e. $\omega(s)$. G may also have $v_0 \in \mathbb{N}$ as its *initial energy*.

In this work, we assume that safety is satisfied *a priori* and we do not include marked states in G. Instead, we consider the (weak) *liveness* property: a system G is live if its generated language $\mathcal{L}(G)$ is live, i.e., $\forall s \in \mathcal{L}(G), \exists u \in E$, s.t. $su \in \mathcal{L}(G)$. That is, there is a transition defined at each state in G so every finite string may have an infinite suffix. This requirement is without loss of generality since it can be relaxed by adding observable self-loops at states where no active events are defined.

Given G, for $x_1, x_2 \in X$ and $e \in E$, we write $x_1 \xrightarrow{e} x_2$ if $f(x_1, e) = x_2$. A *run* in G is a sequence of alternating states and events: $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} x_n$ and it may be infinitely long. We denote the set of all runs in G by $Run(G)$, and specifically, the set of infinite runs by $Run_{inf}(G)$, so that $Run(G) \subset Run_{inf}(G)$. A run is called *initial* if its initial state is the initial state of G. Run r forms a *cycle* if $x_1 = x_n$, and r is called *simple* if $\forall i, j \in \{1, 2, \ldots, n-1\}$, $i \neq j \Rightarrow x_i \neq x_j$. If r is a cycle, the corresponding string $e_1 e_2 \cdots e_{n-1}$ forms a *loop*, which is also called *simple* if r is simple.

Given $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots \xrightarrow{e_n} x_{n+1}$, its (accumulative) payoff is $\sum_{i=1}^{n} \omega(e_i)$ and its mean payoff is $\frac{1}{n}\sum_{i=1}^{n} \omega(e_i)$. The system's *energy level* after r is written as $EL(r) = v_0 + \sum_{i=1}^{n} \omega(e_i)$. The energy level changes dynamically with event occurrences.

For infinite runs, we also define $V_{mp} : Run_{inf}(G) \to \mathbb{R}$ as the *limit mean payoff* of an infinite run. Given $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots$,

$$V_{mp}(r) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \omega(e_i) \tag{1}$$

Since G is with finite state space and the weight of each event is bounded, the limit of the infimum of the sequence $\{\frac{1}{n}\sum_{i=1}^{n} \omega(e_i)\}$ always exists. Notice that the value of $V_{mp}(r)$ does not depend on any sequence that appears finitely often when r is infinite. Also for $i \leq j$, if $x_i \xrightarrow{e_i} x_{i+1} \xrightarrow{e_{i+1}} \cdots \xrightarrow{e_j} x_{j+1}$ is the only cycle that appears infinitely often in the run r, then we have:

$$V_{mp}(r) = \frac{1}{j-i+1} \sum_{l=i}^{j} \omega(e_l) \tag{2}$$

The event set E is partitioned as $E = E_c \cup E_{uc}$, where E_c is the set of controllable events and E_{uc} is the set of uncontrollable events. G is partially observed and E is also partitioned as $E = E_o \cup E_{uo}$, where E_o and E_{uo} are the sets of observable and unobservable events, respectively. The natural projection $P : E^* \to E_o^*$ is recursively defined as: $\forall t' \in E^*, e \in E, P(\epsilon) = \epsilon, P(t) = P(t'e) = P(t')P(e)$ where $P(e) = e$ if $e \in E_o$ and $P(e) = \epsilon$ if $e \in E_{uo} \cup \{\epsilon\}$.

The system G is controlled by a *supervisor* $S : P[\mathcal{L}(G)] \to 2^X$ that dynamically enables/disables events (Cassandras & Lafortune, 2008). Let $\mathbb{S}$ be the set of supervisors. We also use S/G to represent the controlled system under S. Accordingly, we denote by $\mathcal{L}(S/G)$ the language generated in S/G and $Run(S/G)$ the set of runs in S/G, respectively. A control decision $\gamma \in 2^E$ is called *admissible* if $E_{uc} \subseteq \gamma$, i.e., uncontrollable events are never disabled. We let $\Gamma = \{\gamma \in 2^E : E_{uc} \subseteq \gamma\}$ be the set of admissible control decisions and only consider Γ in the remainder of the work.

The supervisor only has partial observation of the system. Given G and a set of states $q \subseteq X$, the *unobservable reach*, denoted by $UR(q)$, is defined as: $UR(q) = \{x' \in X : \exists x \in q, s \in E_{uo}^*,$ s.t. $f(x, s) = x'\}$. Specifically, the unobservable reach under a set of events $\gamma \subseteq E$, denoted by $UR_\gamma(q)$, is defined as: $UR_\gamma(q) = \{x' \in X : \exists x \in q, s \in (E_{uo} \cap \gamma)^*,$ s.t. $f(x, s) = x'\}$. The *observable reach* under event $e_o \in E_o$, denoted by $Next_{e_o}(q)$, is defined as: $Next_{e_o}(q) = \{x' \in X : \exists x \in q$ s.t. $f(x, e_o) = x'\}$.

The *observer* of G is defined as: $Obs(G) = (X_{obs}, E_o, \delta, x_{obs,0})$ where $X_{obs} \subseteq 2^X$ is the state space; $x_{obs,0} = UR(\{x_0\})$ is the initial state and δ is the transition function where $\forall x_{obs} \in X_{obs}$, $\forall e_o \in E_o$: $\delta(x_{obs}, e_o) = UR(Next_{e_o}(x_{obs}))$. The event weight function is omitted here in the definition. An observer state is also termed a *(current) state estimate* of the system.

3. Problem formulations

In this section, we formulate the optimal mean payoff supervisory control problems with and without the constraint of nonnegative energy level, respectively. Before stating them, we first assume that there are no unobservable loops in $\mathcal{L}(G)$, and this assumption holds throughout the remainder of this work.

Assumption 1 (*No Unobservable Loops*)**.** Given an automaton G, $\forall x \in X, \forall s \in E^* \setminus \{\epsilon\}, [f(x, s) = x] \Rightarrow [P(s) \neq \epsilon]$.

Problem 1 (*Optimal Mean Payoff Supervisory Control Under Partial Observation-nonnegative Energy Level Case*)**.** Given system G with initial energy $v_0 \in \mathbb{N}$, design a supervisor $S^* \in \mathbb{S}$ such that: (i) $\mathcal{L}(S^*/G)$ is live; (ii) $\forall r \in Run(S^*/G)$: $EL(r) \geq 0$; (iii) $\inf_{r \in Run_{inf}(S^*/G)} V_{mp}(r) = \sup_{S \in \mathbb{S}} \inf_{r \in Run_{inf}(S/G)} V_{mp}(r)$.

In other words, the supervised system satisfies the following conditions: (i) it is live; (ii) its energy level for any run is nonnegative; (iii) its worst case limit mean payoff is maximized.

As a variant, we require the supervisor to enforce nonnegative limit mean payoffs. To study the new problem, we make Assumption 2 on the system. Given an observer state $x_{obs} \in X_{obs}$, we let $Loop(x_{obs}) = \{l \in E_o^* \setminus \{\epsilon\} : \delta(x_{obs}, l) = x_{obs}$ and $\forall l' < l$ s.t. $l' \neq \epsilon, \delta(x_{obs}, l') \neq x_{obs}\}$ be the set of non-ϵ simple loops starting from x_{obs}. Given string $l \in Loop(x_{obs})$, we let $SimLp(x_{obs}, l) = \{t \in E^* \setminus \{\epsilon\} : \exists x \in x_{obs}$ s.t. $f(x, t) = x, P(t) = l$ and $\forall t' < t, f(x, t') \neq x\}$ be the set of non-ϵ simple loops with the same projection l and starting from some state in x_{obs}.

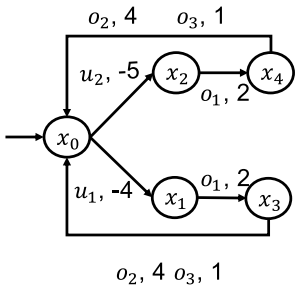


Fig. 1. An automaton with unambiguous cycle payoffs.

Assumption 2 (*Unambiguous Cycle Payoffs*)**.** Given G and its observer $Obs(G)$, $\forall x_{obs} \in X_{obs}$, $\forall l \in Loop(x_{obs})$, and $\forall s, s' \in SimLp(x_{obs}, l)$, we have either $\omega(s) < 0 \Rightarrow \omega(s') < 0$ or $\omega(s) \geq 0 \Rightarrow \omega(s') \geq 0$.

In other words, for two simple loops with the same projection, their payoffs should have the same sign. This assumption is inspired by the decidable classes of mean payoff games with partial observation in Hunter et al. (2018). Later on in Section 4, we will see how this assumption guarantees a finite game structure for solving Problem 2. We say that a system is *with unambiguous cycle payoffs* if it satisfies Assumption 2. Checking Assumption 2 may be reduced to comparing the accumulative weights of every pair of simple cycles in G. By graph theory, the number of simple cycles in a graph may be exponential with respect to the number of states, thus it may take exponential time to verify the assumption.

Example 1. Let the system G in Fig. 1 be with $E_{uo} = \{u_1, u_2\}$ and $E_o = \{o_1, o_2, o_3\}$. The weight of each event is shown in the figure. There are 4 simple cycles: $x_0 \xrightarrow{u_1} x_1 \xrightarrow{o_1} x_3 \xrightarrow{o_2} x_0$ with payoff 2, $x_0 \xrightarrow{u_2} x_2 \xrightarrow{o_1} x_4 \xrightarrow{o_2} x_0$ with payoff 1, $x_0 \xrightarrow{u_1} x_1 \xrightarrow{o_1} x_3 \xrightarrow{o_3} x_0$ with payoff -1 and $x_0 \xrightarrow{u_2} x_2 \xrightarrow{o_1} x_4 \xrightarrow{o_3} x_0$ with payoff -2. So G is with unambiguous cycle payoffs.

Problem 2 (*Optimal Mean Payoff Supervisory Control Under Partial Observation-nonnegative Mean Payoff Case*)**.** Given system G with unambiguous cycle payoffs and mean payoff threshold $v \in \mathbb{N}$, design a supervisor $S^* \in \mathbb{S}$ such that: (i) $\mathcal{L}(S^*/G)$ is live; (ii) $\forall r \in Run_{inf}(S^*/G)$: $V_{mp}(r) \geq v$; (iii) $\inf_{r \in Run_{inf}(S^*/G)} V_{mp}(r) = \sup_{S \in \mathbb{S}} \inf_{r \in Run_{inf}(S/G)} V_{mp}(r)$.

Compared with Problem 1, we still require that the supervised system be live and the worst case limit mean payoff be optimized. The difference is that we omit the requirement of nonnegative energy level. Instead, the limit mean payoff (rate of energy gain) of any infinite run is required to be above a given threshold v (not necessarily 0) in (ii) . However, given $v \neq 0$, we may subtract v from the weight of each event and equivalently evaluate whether the limit mean payoff is above 0. Hence, we simply let $v = 0$ in the following discussion without loss of generality.

Specifically, we call the first two conditions in Problem 1 (respectively Problem 2) as its *mean payoff decision problem* which does not consider optimization. In both Problems 1 and 2, the optimal supervisor should maximize the worst case limit mean payoff. We may imagine that the supervisor is "playing a game" against an antagonistic opponent, where the supervisor is to maximize its mean payoff while its opponent is to prevent the supervisor. Note that the two sides may have asymmetric information since the supervisor only has partial observation of the system. Thus, it is essential to construct proper estimates for current states and the energy level of the system so that the supervisor makes correct decisions. In the following discussion, we solve Problems 1 and 2 sequentially: we first find solutions to their corresponding mean payoff decision problems, then completely solve them by resolving the optimization issues.

4. First cycle energy inclusive controller

As a first step of solving Problems 1 and 2, we define *energy information states* and *First Cycle Energy Inclusive Controller* (FCEIC) to transform both problems to two-player games between the supervisor and the environment. The FCEIC is the game structure, which records the update of current state estimates and the energy level of the system under control. It is inspired by the Bipartite Transition System and All Enforcement Structure in Yin and Lafortune (2016a, 2016b), which include supervisors enforcing several logical properties in DES. We build two FCEICs (one for each problem): they are similar to each other except that we impose nonnegative energy level on the FCEIC for Problem 1.

4.1. Energy information states

In order to track state estimates and string payoffs, we define *energy information states* which provide a compact way of encoding information. Here we let $|\cdot|$ be the cardinality of a set.

Before giving the definition, we first present some necessary order relations for vectors in $\mathbb{Z}^n$. Given two vectors $v_1 = [v_1(1), v_1(2), \ldots, v_1(n)], v_2 = [v_2(1), v_2(2), \ldots, v_2(n)] \in \mathbb{Z}^n$, we denote by $v_1 \leq v_2$ (respectively $v_1 \geq v_2$) if $\forall 1 \leq i \leq n$, $v_1(i) \leq v_2(i)$ (respectively $v_1(i) \geq v_2(i)$). We also denote by $v_1 < v_2$ if $\forall 1 \leq i \leq n$, $v_1(i) \leq v_2(i)$ and $\exists 1 \leq j \leq n$, $v_1(j) < v_2(j)$ (respectively $\forall 1 \leq i \leq n$, $v_1(i) \geq v_2(i)$ and $\exists 1 \leq j \leq n$, $v_1(j) > v_2(j)$), i.e., at least one element in v_1 is strictly smaller (larger) than the element at the same position in v_2.

Definition 1 (*Energy Information States*)**.** Given system G, an energy information state is a tuple $q^e = (q, v) \in 2^X \times (\cup_{k=1}^{|X|} \mathbb{Z}^k)$. Let $Est(q^e)$ and $Lev(q^e)$ denote the state estimate and energy level components of q^e, respectively. So $q^e = (Est(q^e), Lev(q^e))$.

Denote by Q^E the set of energy information states. There are two components in an energy information state q_e: a current estimate of the system state and a vector representing the energy level of the system when reaching the states in the estimate. Each state in $Est(q^e)$ corresponds to a value in $Lev(q^e)$, whose dimension equals the number of states in the state estimate. Given $x \in Est(q^e)$, we also write $Lev(q^e, x)$ as the element in $Lev(q^e)$ that corresponds to x. When $Est(q^e) = \{x_1, x_2 \cdots x_k\}$, $Lev(q^e)$ is usually expressed in a vector form $[Lev(q^e, x_1), Lev(q^e, x_2), \ldots, Lev(q^e, x_k)]$. By convention in this work, elements in $Lev(q^e)$ are placed in an increasing order w.r.t. state names in $Est(q^e)$. Let $\vec{0}$ be the vector of all 0s with proper dimensions. We call q^e *desirable* if $Lev(q^e) \geq \vec{0}$, i.e., nonnegative energy level for every state in $Est(q^e)$.

We define an order $\preccurlyeq$ over Q^E: for $q_1^e, q_2^e \in Q^E$, $q_1^e \preccurlyeq q_2^e$ if $Est(q_1^e) = Est(q_2^e)$ and $Lev(q_1^e) \leq Lev(q_2^e)$. We also say that q_2^e *subsumes* q_1^e if $q_1^e \preccurlyeq q_2^e$, i.e., q_2^e shares the same state estimate with q_1^e and the energy level vector of q_2^e is no less than that of q_1^e in a point-wise sense. We define another order $\prec$ over Q^E: for $q_1^e, q_2^e \in Q^E$, $q_1^e \prec q_2^e$ if $Est(q_1^e) = Est(q_2^e)$, $Lev(q_1^e) < Lev(q_2^e)$. That is to say, q_1^e and q_2^e have the same state estimate and there exists $i \geq 1$ such that $Lev(q_1^e)(i) < Lev(q_2^e)(i)$. By Dickson's lemma (see, e.g., Levy, 2002), "$\leq$" on nonnegative integer space $\mathbb{N}^k$ is a *well-quasi ordering* for any $k \in \mathbb{N}^+$. We further argue that $\preccurlyeq$ on desirable energy information states is also a well-quasi ordering, i.e., for any infinite sequence of desirable energy information states $q_1^e, q_2^e \cdots$, there exist two indexes $i < j$, such that $q_i^e \preccurlyeq q_j^e$.

We call $q^{ae} \in Q^E \times \Gamma$ an *augmented energy information state*, which augments an energy information state with a control decision. Let $I_E(q^{ae})$, $\Gamma(q^{ae})$ denote the energy information state component and control decision component of q^{ae}, respectively, so $q^{ae} = (I_E(q^{ae}), \Gamma(q^{ae}))$. With a slight abuse of notation, we also use $Lev(q^{ae}, x)$ to stand for $Lev(I_E(q^{ae}), x)$ where $x \in Est(I_E(q^{ae}))$. An

augmented energy information state q^{ae} is also called desirable if $Lev(I_E(q^{ae})) \geq 0$. Then we discuss how (augmented) energy information states are updated when the supervisor makes decisions or enabled observable events occur.

Definition 2 (γ-*successor*)**.** For $\gamma \in \Gamma$, $q^{ae} \in Q^E \times \Gamma$ is a γ-successor of $q^e \in Q^E$ if: (i) $Est(I_E(q^{ae})) = UR_\gamma(Est(q^e))$; (ii) $\forall x' \in Est(I_E(q^{ae}))$, $Lev(q^{ae}, x') = \min_\xi \{Lev(q^e, x) + \omega(\xi) : \exists x \in Est(q^e), \xi \in (E_{uo} \cap \gamma)^*$ s.t. $f(x, \xi) = x'\}$.

If q^{ae} is a γ-successor of q^e, then the state estimate of $I_E(a^{ae})$ is the unobservable reach of $Est(q^e)$ under γ and we append $I_E(a^{ae})$ with the control decision γ. We also track the *minimum* energy level under γ, which is achieved by some unobservable string ξ reaching a possible state in $Est(I_E(q^{ae}))$.

Definition 3 (e_o-*successor*)**.** For $e_o \in E_o$, $q^e \in Q^E$ is an e_o-successor of $q^{ae} \in Q^E \times \Gamma$ if: (i) $e_o \in \Gamma(q^{ae}) = \gamma$ and $Est(q^e) = Next_{e_o}(Est(I_E(q^{ae})))$; (ii) $\forall x \in Est(q^e)$, $Lev(q^e, x) = \min_{x'} \{Lev(q^{ae}, x') + \omega(e_o) : \exists x' \in Est(I_E(q^{ae}))$ s.t. $f(x', e_o) = x\}$.

If q^e is an e_o-successor of q^{ae}, then the state estimate component of q^e is the observable reach of $Est(I_E(q^{ae}))$ under e_o. Meanwhile, we track the *minimum* energy level when e_o occurs and a certain state in $Est(q^{ae})$ is reached. When there is a sequence of alternating control decisions and observable events, we introduce *control-observation sequence* to characterize the update of (augmented) energy information states.

Definition 4 (*Control-observation Sequence*)**.** A control-observation sequence is a sequence of alternating energy/augmented energy information states, observable events and control decisions:

$$\rho = q_1^e \xrightarrow{\gamma_1} q_1^{ae} \xrightarrow{e_1} q_2^e \xrightarrow{\gamma_2} q_2^{ae} \cdots \xrightarrow{\gamma_{n-1}} q_{n-1}^{ae} \xrightarrow{e_{n-1}} q_n^e \text{ or}$$

$$\rho' = q_1^e \xrightarrow{\gamma_1} q_1^{ae} \xrightarrow{e_1} q_2^e \xrightarrow{\gamma_2} q_2^{ae} \cdots \xrightarrow{\gamma_{n-1}} q_{n-1}^{ae} \xrightarrow{e_{n-1}} q_n^e \xrightarrow{\gamma_n} q_n^{ae}$$

where $\forall i \leq n$, $\gamma_i \in \Gamma$, $e_i \in E_o$, $q_i^e \in Q^E$, $q_i^{ae} \in Q^E \times \Gamma$, q_i^{ae} is a γ_i-successor of q_i^e and q_{i+1}^e is an e_i-successor of q_i^{ae}.

By convention, we also denote by $\rho_k = q_1^e \xrightarrow{\gamma_1} q_1^{ae} \xrightarrow{e_1} q_2^e \xrightarrow{\gamma_2} q_2^{ae} \cdots \xrightarrow{\gamma_{k-1}} q_{k-1}^{ae} \xrightarrow{e_{k-1}} q_k^e$ and $\rho_k' = q_1^e \xrightarrow{\gamma_1} q_1^{ae} \xrightarrow{e_1} q_2^e \xrightarrow{\gamma_2} q_2^{ae} \cdots \xrightarrow{\gamma_{k-1}} q_{k-1}^{ae} \xrightarrow{e_{k-1}} q_k^e \xrightarrow{\gamma_k} q_k^{ae}$, for $1 \leq k \leq n$. Strings are generated under the control decisions in such sequences.

Definition 5 (*Strings Generated By a Control-observation Sequence*)**.** Given a control-observation sequence ρ or ρ', the set of strings generated by ρ is defined recursively as: $\forall 1 \leq k \leq n$,

$$\begin{aligned}
Str(\rho_1) =& \{\epsilon\}\\
Str(\rho_1') =& \{\xi_1 \in E_{uo}^* : \exists x \in Est(q_1^e), x' \in Est(I_E(q_1^{ae})),\\
&\xi_1 \in (\gamma_1 \cap E_{uo})^* \text{ s.t. } f(x, \xi_1) = x'\}\\
Str(\rho_{k+1}) =& \{s_k' e_k : \exists x \in Est(q_1^e), x' \in Est(I_E(q_k^{ae})), x'' \in Est(q_{k+1}^e),\\
&s_k' \in Str(\rho_k'), \text{ s.t. } f(x, s_k') = x', f(x', e_k) = x''\}\\
Str(\rho_{k+1}') =& \{s_{k+1}\xi_{k+1} : \exists x \in Est(q_1^e), x' \in Est(q_{k+1}^e), x'' \in\\
&Est(I_E(q_{k+1}^{ae})), s_{k+1} \in Str(\rho_{k+1}), \xi_{k+1} \in (\gamma_{k+1} \cap E_{uo})^*,\\
&\text{s.t. } f(x, s_{k+1}) = x', f(x', \xi_{k+1}) = x''\}
\end{aligned}$$

The following proposition shows that given a control-observation sequence, the energy level vector of an energy or augmented energy information state always tracks the *minimum* payoff of strings reaching the states in the state estimate.

Proposition 1. *Given a control-observation sequence ρ as in Definition 4, we have that $\forall x \in Est(q_n^e)$:*

$$Lev(q_n^e, x) = \min_{s \in Str(\rho)} \{\omega(s) : \exists \tilde{x} \in Est(q_1^e), \text{ s.t. } f(\tilde{x}, s) = x\} \tag{3}$$

Given a control-observation sequence ρ' as in Definition 4, we have that $\forall x \in Est(I_E(q_n^{ae}))$:

$$Lev(q_n^{ae}, x') = \min_{s \in Str(\rho')} \{\omega(s) : \exists \tilde{x} \in Est(q_1^e), \text{ s.t. } f(\tilde{x}, s) = x'\} \tag{4}$$

Proof. See the Appendix. □

The proof of Proposition 1 is in a dynamic programming manner. Since we account for the minimum string payoff when creating a new e_o-successor or γ-successor, the minimum payoff is computed by taking the minimal energy value of all strings consistent with the observation. Note that those strings only differ in their unobservable substrings.

4.2. Construction of the FCEIC

Next we transfer Problems 1 and 2 to games between the supervisor and the environment. In general, the games are infinite since we require livness and evaluate limit mean payoffs. To efficiently solve the problems, we define a compact information structure called the First Cycle Energy Inclusive Controller (FCEIC) by considering the "first cycles" formed in the games.

The two variants of FCEICs are formally defined by construction, i.e., by adding feasible e_o-successors and γ-successors to the state space recursively in Algorithms 1 and 2, respectively. The FCEICs with respect to system G for both problems are constructed in a similar way and of the same generic form $(Q_Y^F, Q_Z^F, E, f_{yz}^F, f_{zy}^F, \Gamma, y_0^e, Q_l^F, v_0)$ where:

- $Q_Y^F \subseteq Q^E$ is the set of energy information states;
- $Q_Z^F \subseteq Q^E \times \Gamma$ is the set of augmented energy information states and for $z^e \in Q_Z^F$, $z^e = (I_E(z^e), \Gamma(z^e))$;
- $f_{yz}^F : Q_Y^F \times \Gamma \to Q_Z^F$ is the transition function from Q_Y^F states to Q_Z^F states, where for all $y^e \in Q_Y^F$, $\gamma \in \Gamma$ and $z^e \in Q_Z^F$, $[f_{yz}^F(y^e, \gamma) = z^e] \Leftrightarrow$ [z^e is a γ-successor of y^e];
- $f_{zy}^F : Q_Z^F \times E_o \to Q_Y^F$ is the transition function from Q_Z^F states to Q_Y^F states, where for all $z^e \in Q_Z^F$, $e_o \in E_o$ and $y^e \in Q_Y^F$, $[f_{zy}^F(z^e, e_o) = y^e] \Leftrightarrow$ [y^e is an e_o-successor of z^e];
- Γ is the set of admissible control decisions;
- $y_0^e \in Q_Y^F$ is the initial energy information state where $Est(y_0^e) = \{x_0\}$ and $Lev(y_0^e) = v_0$;
- $Q_l^F \subset Q_Y^F$ is the set of leaf states where no transitions are defined and we partition $Q_l^F = Q_{lg}^F \cup Q_{lb}^F$;
- $v_0 \in \mathbb{N}$ is the initial energy of the system.

For simplicity, Q_Y^F states are also named Y-states and Q_Z^F states are named Z-states in the remainder of the work. A Z-state z^e is *deadlock free* if $\forall x \in Est(I_E(z^e))$, $\exists e \in \Gamma(z^e)$, s.t. $f(x, e)!$, i.e., at least one event is enabled at every state in the state estimate of z^e. Otherwise, z^e is called a *deadlocking* state. Since there are no unobservable loops in G by Assumption 1, a deadlock free Z-state always has transitions defined out of it.

The FCEIC in general describes a game between the supervisor and the environment. A Y-state is an energy information state where the supervisor issues control decisions. If the supervisor issues an admissible control decision γ, f_{yz}^F transition is defined out of a Y-state, which follows the definition of γ-successor. A Z-state is an augmented energy information state, where the environment selects observable events to occur from the events enabled by the supervisor. When a particular observable event e_o is selected to occur by the environment, f_{zy}^F transition is defined out of a Z-state, which follows the definition of e_o-successor. Then it is again the supervisor's turn to make the next control decision. This is consistent with the mechanism of supervisory control

Algorithm 1 Construction of the FCEIC for Problem 1

Input: G, v_0
Output: $FCEIC = (Q_Y^F, Q_Z^F, E, f_{yz}^F, f_{zy}^F, \Gamma, y_0^e, Q_I^F, v_0)$
1: $Q_Y^F = \{y_0^e\}$, $Q_Z^F = \emptyset$, $Q_I^F = \emptyset$, $Q_{Ib}^F = \emptyset$;
2: $FirstCycle_1(y_0^e, FCEIC)$;
3: Return $FCEIC$;
4: **procedure** $FirstCycle_1(y^e, \text{FCEIC})$
5: **for** $\gamma \in \Gamma$ **do**
6: Let z^e be the γ-successor of y^e;
7: **if** z^e is deadlock free and energy safe **then**
8: Add transition $y^e \xrightarrow{\gamma} z^e$ to f_{yz}^F;
9: **if** $z^e \notin Q_Z^F$ **then**
10: $Q_Z^F = Q_Z^F \cup \{z^e\}$;
11: **for** $e_o \in \gamma \cap E_o$ **do**
12: Let $\tilde{y}^e$ be an e_o-successor of z^e;
13: Add transition $z^e \xrightarrow{e_o} \tilde{y}^e$ to f_{zy}^F;
14: **if** $\tilde{y}^e \notin Q_Y^A$ **then**
15: $Q_Y^F = Q_Y^F \cup \{\tilde{y}^e\}$;
16: **if** $\tilde{y}^e$ is energy safe **then**
17: **if** there exists a run: $y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e} \tilde{y}^e$ and $\exists j < n$, s.t. $y_j^e \preccurlyeq \tilde{y}^e$ **then**
18: Stop searching from $\tilde{y}^e$, define $Sub(\tilde{y}^e) = y_j^e$, let $Q_I^F = Q_I^F \cup \{\tilde{y}^e\}$, $Q_{Ig}^F = Q_{Ig}^F \cup \{\tilde{y}^e\}$;
19: **else** $FirstCycle_1(\tilde{y}^e, FCPEC)$;
20: **else** Stop searching from $\tilde{y}^e$, let $Q_I^F = Q_I^F \cup \{\tilde{y}^e\}$ and $Q_{Ib}^F = Q_{Ib}^F \cup \{\tilde{y}^e\}$;

Algorithm 2 Construction of the FCEIC for Problem 2

Input: G, v_0
Output: $FCEIC = (Q_Y^F, Q_Z^F, E, f_{yz}^F, f_{zy}^F, \Gamma, y_0^e, Q_I^F, v_0)$
1: $Q_Y^F = \{y_0^e\}$, $Q_Z^F = \emptyset$, $Q_I^F = \emptyset$, $Q_{Ib}^F = \emptyset$;
2: $FirstCycle_2(y_0^e, FCEIC)$;
3: Return $FCEIC$;
4: **procedure** $FirstCycle_2(y^e, \text{FCEIC})$
5: **for** $\gamma \in \Gamma$ **do**
6: Let z^e be a γ-successor of y^e;
7: **if** z^e is deadlock free and energy safe **then**
8: Add transition $y^e \xrightarrow{\gamma} z^e$ to f_{yz}^F;
9: **if** $z^e \notin Q_Z^F$ **then**
10: $Q_Z^F = Q_Z^F \cup \{z^e\}$;
11: **for** $e_o \in \gamma \cap E_o$ **do**
12: Let $\tilde{y}^e$ be an e_o-successor of z^e;
13: Add transition $z^e \xrightarrow{e_o} \tilde{y}^e$ to f_{zy}^F;
14: **if** $\tilde{y}^e \notin Q_Y^A$ **then**
15: $Q_Y^F = Q_Y^F \cup \{\tilde{y}^e\}$;
16: **if** there exists a run: $y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e} \tilde{y}^e$ and $\exists j < n$, s.t. $y_j^e \preccurlyeq \tilde{y}^e$ **then**
17: Stop searching from $\tilde{y}^e$, define $Sub(\tilde{y}^e) = y_j^e$, let $Q_I^F = Q_I^F \cup \{\tilde{y}^e\}$, $Q_{Ig}^F = Q_{Ig}^F \cup \{\tilde{y}^e\}$;
18: **if** There exists a run: $y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \xrightarrow{\gamma_1} z_1^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e} \tilde{y}^e$ and $\exists j < n$, s.t. $\tilde{y}^e \prec y_j^e$ **then**
19: Stop searching from $\tilde{y}^e$, let $Q_I^F = Q_I^F \cup \{\tilde{y}^e\}$ and $Q_{Ib}^F = Q_{Ib}^F \cup \{\tilde{y}^e\}$;
20: **else**$FirstCycle_2(\tilde{y}^e, FCEIC)$;

under partial observation where the supervisor's decisions get updated after the occurrence of observable events. In this manner, the two players take turns to play and a game is formed.

The procedure $FirstCycle_i$ where $i \in \{1, 2\}$ in either algorithm builds the state space of the FCEIC by a depth-first search like process. We first discuss $FirstCycle_1$ in Algorithm 1. In this process, we only add deadlock free Z-states to the structure and ensure that there are events enabled at every state in the state estimate of any Z-state. In lines 16, 17 and 18 of Algorithm 1, if the newly added energy safe state $\tilde{y}^e$ subsumes a non-leaf state y_j^e on the run starting from the initial state, then the two energy information states share the same state estimate but the new state $\tilde{y}^e$ has a higher or equal energy level vector compared with y_j^e. We also know that some simple cycles with nonnegative payoffs are formed in the system for the first time. Then we terminate searching and add the new state as a leaf state of the FCEIC. That is why we call this structure first cycle energy inclusive controller. In the following sections, we will explain in more detail why it is sufficient to consider simple cycles to solve Problem 1. On the other hand, if a new Z-state or Y-state is not energy safe, we stop searching since the system's energy level drops below 0 at some state, thus the second requirement in Problem 1 is violated.

Similarly for $FirstCycle_2$ of Algorithm 2, in lines 16 and 18, if the newly added state $\tilde{y}^e$ subsumes or is subsumed by an existing state on the run from initial state y_0^e to $\tilde{y}^e$, we know that the two energy information states share the same state estimate and $\tilde{y}^e$ has a higher, lower or equal energy level vector compared with that state. We also know that some simple cycles with nonnegative or negative payoffs are formed in the system for the first time. Then we terminate searching and add the new state $\tilde{y}^e$ as a leaf state of the FCEIC. Since Problem 2 does not require nonnegative energy level of the system, the states created by $FirstCycle_2$ are not necessarily energy safe.

Next, we partition leaf Y-states as: $Q_I^F = Q_{Ig}^F \cup Q_{Ib}^F$ where Q_{Ig}^F represents *good leaf states* and Q_{Ib}^F represents *bad leaf states*. In the FCEIC for Problem 1, a good leaf state is energy safe and subsumes a non-leaf state, while a bad leaf state is energy unsafe. If a good leaf state is reached, there are simple cycles with nonnegative payoffs in the system whose energy level would be nonnegative forever if those cycles are traversed indefinitely. However, if a bad leaf state is reached, there exists some string so that the energy level of the system drops below 0. Similarly, in the FCEIC for Problem 2, a good leaf state subsumes a non-leaf state while a bad leaf state is subsumed by a non-leaf state. If a good leaf state is reached, we know there exist simple cycles with nonnegative payoffs in the system; if bad leaf state is reached, there exist simple cycles with negative payoffs. In both algorithms, we define $Sub(y^e)$ to store the preceding state subsumed by good leaf state y^e. Actually, the supervisors in both Problems 1 and 2 should reach good leaf states and avoid bad ones, which is explained in more detail later on. Finally, if no state subsumes another, we call *FirstCycle* recursively in both algorithms until no more new states are added to the structure. We may also show that Algorithm 1 and Algorithm 2 return a finite and acyclic structure.

Theorem 1. *Algorithm 1 returns a finite structure.*

Proof. By contradiction. Assume that the FCEIC is infinite. Since E, $\Gamma \subseteq 2^E$ and E_o are finite, the number of transitions at each state in the structure is finite. By König's lemma (Levy, 2002) and Algorithm 1, there exists an infinite run $y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \xrightarrow{\gamma_1} z_1^e \cdots$ in the FCEIC and it is never the case that $\exists y_i^e, y_j^e, i < j$, s.t. $y_i^e \preccurlyeq y_j^e$. However, this contradicts with $\preccurlyeq$ being a well-quasi ordering on energy safe energy information states. □

Theorem 2. *Algorithm 2 returns a finite structure.*

Table 1
Energy and augmented energy information states in Fig. 3.

State name	State components
y_0^e	$\{\{x_0\}, 3\}$
z_0^e	$\{\{x_0, x_1, x_2\}, [3, 1, 0], \gamma_0\}$
y_1^e	$\{\{x_3, x_4\}, [2, 1]\}$
z_1^e	$\{\{x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}, [2, 1, 5, 2, 7, 2, 6, 5], \gamma_1\}$
y_2^e	$\{\{x_{12}\}, 4\}$
z_2^e	$\{\{x_{12}\}, 4, \gamma_0\}$
y_{2-2}^e	$\{\{x_{12}\}, 6\}$
z_8^e	$\{\{x_9, x_{12}, x_{14}\}, [0, 4, 3], \gamma_2'\}$
y_{2-3}^e	$\{\{x_{12}\}, -2\}$
y_{2-4}^e	$\{\{x_{12}\}, 6\}$
y_3^e	$\{\{x_{13}\}, 2\}$
z_3^e	$\{\{x_{13}\}, 2, \gamma_0\}$
y_{3-2}^e	$\{\{x_{13}\}, 4\}$
z_9^e	$\{\{x_{10}, x_{13}, x_{15}\}, [1, 2, 1], \gamma_3\}$
y_{3-3}^e	$\{\{x_{13}\}, -2\}$
y_{3-4}^e	$\{\{x_{13}\}, 4\}$
z_4^e	$\{\{x_3, x_4, x_5, x_7\}, [2, 1, 5, 7], \gamma_4\}$
y_{1-2}^e	$\{\{x_3, x_4\}, [3, 2]\}$
z_5^e	$\{\{x_3, x_4, x_6, x_8\}, [2, 1, 2, 2], \gamma_5\}$
y_{1-3}^e	$\{\{x_3, x_4\}, [3, 2]\}$
z_6^e	$\{\{x_3, x_4\}, [2, 1], \gamma_0\}$
y_{1-4}^e	$y_{1-4}^e = \{\{x_3, x_4\}, [3, 2]\}$
y_{1-5}^e	$y_{1-5}^e = \{\{x_3, x_4\}, [3, 2]\}$

Proof. By contradiction. Assume that the FCEIC is infinite. Since E, $\Gamma \subseteq 2^E$ and E_o are finite, the number of transitions defined at each state in the structure is finite. Then by König's lemma (see, e.g., Levy, 2002), there exists an infinite run $y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \xrightarrow{\gamma_1} z_1^e \cdots$ in the FCEIC such that it is neither the case that $\exists y_i^e, y_j^e$, $i < j$, s.t. $y_i^e \preccurlyeq y_j^e$ nor the case that $y_j^e \prec y_i^e$. That means there exist y_i^e, y_j^e $(i < j)$ and integers $k \neq l$ s.t. $Est(y_i^e) = Est(y_j^e)$, $Lev(y_i^e)(k) \leq Lev(y_j^e)(k)$ and $Lev(y_i^e)(l) > Lev(y_j^e)(l)$ for elements in $Lev(y_i^e)$ and $Lev(y_j^e)$. Hence there exist two simple cycles in G: $x_1 \xrightarrow{e_1} x_2 \cdots \xrightarrow{e_n} x_1$ and $x_1' \xrightarrow{e_1'} x_2' \cdots \xrightarrow{e_n'} x_1'$ s.t. $x_1, x_1' \in Est(y_i^e)$, $P(e_1 \cdots e_n) = P(e_1' \cdots e_n')$, $\omega(e_1 \cdots e_n) \geq 0$ and $\omega(e_1' \cdots e_n') < 0$. However, this contradicts with Assumption 2 that G is with unambiguous cycle payoffs. □

The size of the state space of the FCEIC is bounded by non-primitive recursive Ackermann functions (see, e.g., Rackoff, 1978) following a similar argument as Hunter et al. (2018) and Pérez (2017). Mean payoff games with incomplete information were also solved by evaluating first simple cycles after the game graphs are unfolded in Hunter et al. (2018).

Example 2. In this example, we construct a first cycle energy inclusive controller following Algorithm 1. Let the system G in Fig. 2 be with $E_o = \{o_1, o_2, o_3, o_4\}$, $E_{uo} = \{a_1, a_2, a_3, a_4, b_1, b_2, c_1, c_2, c_3, c_4.c_5\}$, $E_c = \{c_1, c_2, c_3, c_4, c_5\}$, $E_{uc} = \{a_1, a_2, a_3, a_4, b_1, b_2, o_1, o_2, o_3, o_4\}$. The weight of each event is shown in the figure and the system has initial energy $v_0 = 3$. Then all admissible control decisions are: $\gamma_0 = E_{uc}$, $\gamma_1 = \{c_1, c_2\} \cup E_{uc}$, $\gamma_2 = \{c_3\} \cup E_{uc}$, $\gamma_2' = \{c_3, c_5\} \cup E_{uc}$, $\gamma_3 = \{c_4\} \cup E_{uc}$, $\gamma_4 = \{c_1\} \cup E_{uc}$, $\gamma_5 = \{c_2\} \cup E_{uc}$. For simplicity, we only include feasible controllable events at the corresponding states in the admissible control decisions.

Then we follow Algorithm 1 to build the FCEIC in Fig. 3. For simplicity of the graph, we do not put the energy level vectors in the figure but show them in Table 1. The elements in each energy level vector are placed in the same order as the order of states in the corresponding state estimate.

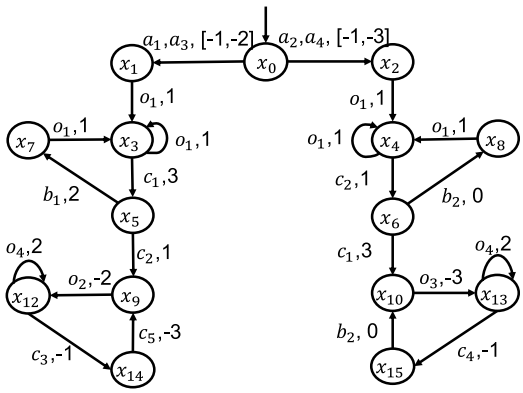


Fig. 2. The automaton G in Example 2.

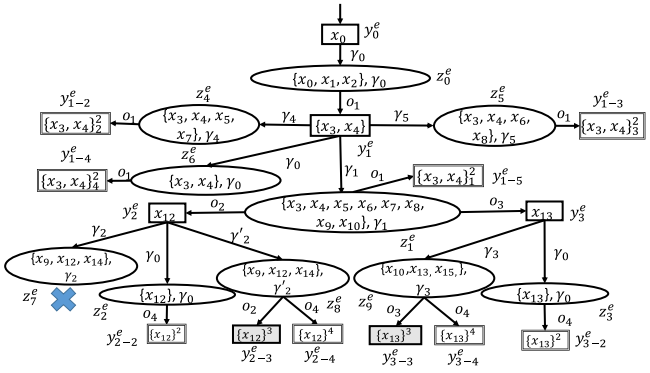


Fig. 3. The first cycle energy controller in Example 2 (without z_7^e).

In the FCEIC, the game is initiated from y_0^e where the only feasible control decision is γ_0. If the supervisor plays γ_0, a Z-state z_0^e is reached where the environment selects observable event o_1 to occur. Then the supervisor takes the turn to play at y_1^e and the rest of the structure is interpreted in a similar way. Notice that at y_2^e, if the supervisor issues control decision γ_2 (enables c_3 and disables c_5), then a deadlocking Z-state z_7^e is reached, where no event can occur at x_{14} after c_5 is disabled. Here z_7^e is not included in the FCEIC by Algorithm 1 and we mark it by a blue cross in Fig. 3. Meanwhile, we calculate the energy level vector of each state. For example, $Est(y_0^e) = \{x_0\}$, $Lev(y_0^e) = v_0 = 3$; since z_0^e is the γ_0-successor of y_0^e, we have that $Est(I_E(z_0^e)) = UR_{\gamma_0}(Est(y_0^e)) = \{x_0, x_1, x_2\}$, $Lev(z_0^e, x_1) = \min\{\omega(a_1), \omega(a_3)\} = 1$, $Lev(z_0^e, x_2) = \min\{\omega(a_2), \omega(a_4)\} = 0$ and $z_0^e = \{\{x_0, x_1, x_2\}, [3, 1, 0], \gamma_0\}$; since y_1^e is the o_1-successor of z_1^e, we have that $Est(y_1^e) = Next_{o_1}(\{x_0, x_1, x_2\}) = \{x_3, x_4\}$, $Lev(y_1^e, x_3) = Lev(z_0^e, x_1) + \omega(o_1) = 2$, $Lev(y_1^e, x_4) = Lev(z_0^e, x_2) + \omega(o_1) = 1$ and $y_1^e = \{\{x_3, x_4\}, [2, 1]\}$.

From the table, we find that $y_1^e \preccurlyeq y_{1-2}^e$, $y_1^e \preccurlyeq y_{1-3}^e$, $y_1^e \preccurlyeq y_{1-4}^e$, $y_1^e \preccurlyeq y_{1-5}^e$, $y_2^e \preccurlyeq y_{2-2}^e$, $y_2^e \preccurlyeq y_{2-4}^e$, $y_3^e \preccurlyeq y_{3-2}^e$ and $y_3^e \preccurlyeq y_{3-4}^e$ by evaluating their energy level vectors. We also find two unsafe states y_{2-3}^e and y_{3-3}^e since $Lev(y_{2-3}^e) = -2$ and $Lev(y_{3-3}^e) = -2$. After checking all states in Fig. 3, we stop adding new states from the leaf states of the FCEIC. Then we have good leaf states $Q_{lg}^F = \{y_{1-2}^e, y_{1-3}^e, y_{1-4}^e, y_{1-5}^e, y_{2-2}^e, y_{2-4}^e, y_{3-2}^e, y_{3-4}^e\}$ and bad leaf states $Q_{lb}^F = \{y_{2-3}^e, y_{3-3}^e\}$. For example, when y_{1-2}^e is reached, we locate three simple cycles with nonnegative payoffs in G: $x_3 \xrightarrow{c_1} x_5 \xrightarrow{b_1} x_7 \xrightarrow{o_1} x_3$ with payoff 6, $x_3 \xrightarrow{o_1} x_3$ with payoff 1 and $x_4 \xrightarrow{o_1} x_4$ with payoff 1. The bad leaf states actually come from the two simple cycles with negative payoffs in G: $x_9 \xrightarrow{o_2} x_{12} \xrightarrow{c_3} x_{14} \xrightarrow{c_5} x_9$ with

payoff -6 and $x_{10} \xrightarrow{o_3} x_{13} \xrightarrow{c_4} x_{15} \xrightarrow{b_2} x_{10}$ with payoff -4. Those two cycles should be avoided if we want to solve Problem 1.

Example 3. The system G is the same as the one in Example 2 and we construct the FCEIC following Algorithm 2. It happens that the FCEIC is the same as the one in Fig. 3. Specifically, $y^e_{2-3} \prec y^e_2$ and $y^e_{3-3} \prec y^e_3$, so y^e_{2-3} and y^e_{3-3} are also bad leaf states in this example. They are due to the two simple cycles with negative payoffs mentioned at the end of Example 2. Again, those two cycles should be avoided if we want to solve Problem 2.

5. Mean payoff decision problems

In this section, we show that there exist solutions for the mean payoff decision problems mentioned at the end of Section 3 if and only if the supervisor has strategies to win the game on the FCEIC. Therefore the first two requirements of Problems 1 and 2 are satisfied. The last requirement (optimization) in both problems will be discussed and addressed in the next section. The following analyses of this work apply to both FCEICs returned by Algorithm 1 and Algorithm 2, so we will not distinguish them but just use the term "FCEIC" when there is no confusion.

By the construction process of Algorithms 1 and 2, we stop expanding the game graph when the first cycles with positive payoffs are formed or the energy level drops below 0. Therefore, the runs in the FCEIC (defined by either Algorithm 1 or 2) are finite control-observation sequences. We denote by $Run(F)$ the set of runs in the FCEIC. Given $r_f \in Run(F)$, we write $y^e \in r_f$ (respectively $z^e \in r_f$) if y^e (respectively z^e) is a Y-state (respectively Z-state) in r_f. We also let $Last_Y(r_f)$ and $Last_Z(r_f)$ be the last Y-state and Z-state of r_f, respectively. Specifically, we denote by $Run_y(F)$ (respectively $Run_z(F)$) the set of runs whose last states are Y-states (respectively Z-states).

Then we discuss *strategies* for both players in the FCEIC, which indicate the choices for players when it is their turn to play. Define the *supervisor's strategy (control strategy)* as a function $\pi_s : Run_y(F) \to \Gamma$ and the *environment's strategy* as $\pi_e : Run_z(F) \to E_o$. Both players select a transition according to their strategies when it is their turn to play. Since the supervisor only has partial observation of the system and makes decisions from state estimates, we call its strategy *observation based*. Denote the set of all supervisor's strategies by Π_s and the set of all environment's strategies by Π_e. If the supervisor plays π_s and the environment plays π_e from the initial state y^e_0, then a unique initial run, denoted by $r_f(\pi_s, \pi_e)$, is generated. We also let $Run(y^e, \pi_s) = \{y^e \xrightarrow{\gamma_1} z^e_1 \xrightarrow{e_1} y^e_2 \cdots \xrightarrow{\gamma_{n-1}} z^e_{n-1} \xrightarrow{e_{n-1}} y^e_n : \forall i < n, \gamma_i = \pi_s(y^e \xrightarrow{\gamma_1} z^e_1 \xrightarrow{e_1} y^e_2 \cdots \xrightarrow{\gamma_{i-1}} z^e_{i-1} \xrightarrow{e_{i-1}} y^e_i)\}$ be the set of runs starting from y^e and *consistent* with control strategy π_s, i.e., the control decisions in the run are specified by π_s.

In the FCEIC, we say the supervisor *wins* the game if only good leaf states are reached, otherwise, the environment wins the game if at least one bad leaf state is reached. So the game on the FCEIC is a *safety game* under full observation after introducing the energy information states. Either the supervisor or the environment has a *winning strategy* from any state in the FCEIC, since safety games are *determined* (Apt & Grädel, 2011).

A strategy $\pi_i \in \Pi_i$ for player $i \in \{s, e\}$ in the FCEIC is *information state based* if the decisions only depend on the current energy or augmented energy information state. In other words, $\pi_i \in \Pi_i$ is information state based if $\pi_i(r_f) = \pi_i(r'_f)$ for all $r_f, r'_f \in Run(F)$ such that $Last(r_f) = Last(r'_f)$. Therefore, information state based strategies for the supervisor and the environment can be represented by $\pi_s : Q^F_Y \to \Gamma$ and $\pi_e : Q^F_Z \to E_o$, respectively. Such a strategy is also called *positional* in the literature, see, e.g., Apt and Grädel (2011), as it only depends on the current position of the player. Since positional strategies are sufficient to win a finite safety game (Apt & Grädel, 2011), we assume that both players play positional strategies in the remainder of this section.

Following the transitions in the FCEIC, we can specify control decisions from Y-states and the control decisions are updated after observable events occur from Z-states. Thus, the control strategies in the FCEIC work in the same way as standard supervisors. In what follows, we will use the words "supervisor" and "supervisor's strategy (control strategy)" interchangeably.

We define *the supervisor's winning region* Win_s as the set of states from which the supervisor has a strategy to only reach good leaf states *for sure* regardless of the environment's strategies. To solve Problem 1 or Problem 2, the supervisor should only reach good leaf states. Actually, the procedures to obtain Win_s for both problems are the same after the FCEIC is built. Hence we present one unified algorithm, i.e., Algorithm 3, to compute Win_s.

Algorithm 3 Compute the winning region of the FCEIC

Input: *FCEIC* returned by Algorithm 1 or Algorithm 2
Output: Win_s for Problem 1 or Problem 2

1: **while** $\exists y^e \in Q^F_Y \setminus Q^F_{lg}$, s.t. y^e has no successor **do**
2: Remove y^e and all $z^e \in Q^F_Z$, such that $f^F_{zy}(z^e, e_o) = y^e$ for some $e_o \in E_o$;
3: Take the accessible part of the structure;
4: Denote the structure by $FCEIC_w$ and return its states;

We briefly discuss the process in Algorithm 3, which calculates the winning region in a *fixed point calculation* manner. All bad leaf states are removed first as well as their preceding Z-states. Then we further prune away Y-states that have no successor states and their preceding Z-states in an iterative manner. Notice that when we remove a Y-state, we also need to remove all its preceding Z-states, otherwise the already enabled observable events are blocked from happening. However, when a Z-state is deleted, we will only remove its preceding Y-state if the Y-state has no successors. The reason is that the supervisor is still able to avoid the removed Z-state when it has other successors. The algorithm stops when no more states can be removed. In this way, we make sure that only good leaf states are reached under certain control strategies and we have the winning region. That is, any control strategy in the $FCEIC_w$ is a winning control strategy in the FCEIC, and all winning control strategies are in the $FCEIC_w$. It is possible that Algorithm 3 returns an empty set thus the environment always wins the game regardless of the supervisor's strategies.

Intuitively, Algorithm 3 is similar to calculating the *supremal controllable sublanguage* in supervisory control under full observation (Cassandras & Lafortune, 2008). The bad leaf states are viewed as undesirable marked states while the good leaf states are viewed as desirable ones; transitions for f^F_{yz} are viewed as controllable while transitions for f^F_{zy} are viewed as uncontrollable.

Next we discuss how a supervisor solving the mean payoff decision problem is obtained. In the FCEIC, the supervisor either aims to achieve a nonnegative energy level (corresponding to Algorithm 1 and Problem 1) or a nonnegative limit mean payoff (corresponding to Algorithm 2 and Problem 2). If only good leaf states are reached under a wining control strategy in the FCEIC, then only simple cycles with a nonnegative payoff are formed in the supervised system. Since the energy level vector in an energy information state returns the minimum string payoff by Proposition 1, the payoffs of strings with the same observation and reaching the same state are all nonnegative if the minimum string payoff is nonnegative.

Therefore, we let the supervisor make the same decision whenever the state estimate of a good leaf state is reached. Intuitively speaking, the supervisor "ignores" the actual energy level of the system and just views the game starting from a good leaf state y^e as the same game that starts from the state subsumed

by y^e. In Algorithms 1 and 2, we define $Sub(y^e)$ as the state that subsumes y^e. We may imagine that y^e is "merged" with $Sub(y^e)$ by letting all transitions going to y^e lead to $Sub(y^e)$ instead. In this manner, the game on the FCIEIC is extended to be infinite and we call the resulting game as an *extended game*. This is essentially the process of *strategy transfer* discussed in Aminof and Rubin (2017) and Hunter et al. (2018), which transfers the strategies on the induced finite game to the original infinite game. A similar procedure is presented in Section 5 of Pruekprasert et al. (2016). In this way, the supervisor perpetually completes cycles with nonnegative payoffs since every simple cycle has a nonnegative payoff. Thus, the limit mean payoff and energy level also become nonnegative for infinite runs. The goal of the supervisor is either to achieve a nonnegative energy level (extended game for Problem 1) or a nonnegative limit mean payoff (extended game for Problem 2). Both objectives may be evaluated by focusing on the first cycles formed by the supervisor and we stop expanding the game graph when the first cycles are formed.

Overall, we claim that any control strategy in the FCEIC$_w$ solves the mean payoff decision problem of Problem 1 or Problem 2. Conversely, we also claim that if the mean payoff decision problem has solutions, then there exist winning control strategies in the FCEIC returned by either Algorithm 1 or 2. Formally speaking, the following two theorems hold.

Theorem 3. *There exists a supervisor solving the mean payoff decision problem of Problem 1 if and only if the supervisor has a winning strategy in the FCEIC defined by Algorithm 1.*

Proof. The "only if" part. Proof by contrapositive, i.e., if there does not exist a winning control strategy in the FCEIC, then there does not exist a supervisor solving the mean payoff decision problem. If no winning control strategy exists, Win_s is empty by Algorithm 3. So $\forall \pi_s \in \Pi_s$, $\exists \pi_e \in \Pi_e$, s.t. $Last_Y(r_f(\pi_s, \pi_e)) \in Q_l^F \Rightarrow Last_Y(r_f(\pi_s, \pi_e)) \in Q_{lb}^F$, i.e, no matter what control decisions are made, there always exist runs ending in bad leaf states. So for π_s, there always exists a run r consistent with π_s in the supervised system such that $V(r) < 0$, i.e., the supervised system's energy level becomes negative under π_s for some string. That is to say, no supervisor solves the mean payoff decision problem.

The "if" part. Suppose that π_s is a winning control strategy in the FCEIC. We follow Algorithm 3 and obtain Win_s and FCEIC$_w$, so π_s is also in the FCEIC$_w$. In the following discussion, we imagine that all transitions leading to a leaf state y^e in the FCEIC$_w$ lead to $Sub(y^e)$ so that the game on the FCEIC$_w$ becomes infinite-duration. That is, $\forall r_f = y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_i^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e_{n-1}} y_n^e \in Run(y_0^e, \pi_s)$ where y_0^e is the initial state of the FCEIC, if $y_n^e \in Q_{lg}^F$, then we extend the domain of π_s by letting $\pi_s(r_f) = \pi_s(y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_i^e \cdots \xrightarrow{e_{m-1}} y_m^e)$ for some $m < n$ and $y_m^e \preceq y_n^e$. Whenever $Est(y_n^e)$ is reached again, the control strategy (supervisor) makes the same decision as if $Est(y_n^e)$ is reached for the first time. By perpetually making the same decision whenever a state estimate is reached, the supervisor guarantees that the energy level after any string in the supervised system never becomes negative. The reasons are that all states in the FCEIC$_w$ are energy safe and the energy level does not decrease when we form the extended games.

Finally, since there are no deadlocking Z-states and every Y-state has successors in the FCEIC$_w$, π_s is live if we follow a similar argument as in Section 5 of Yin and Lafortune (2016b). Thus, π_s solves the mean payoff decision problem of Problem 1. □

Theorem 4. *There exists a supervisor solving the mean payoff decision problem of Problem 2 if and only if the supervisor has a winning strategy in the FCEIC defined by Algorithm 2.*

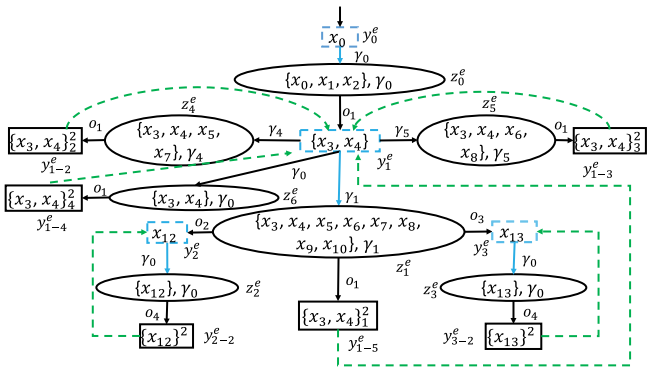


Fig. 4. The FCEIC$_w$ with dashed green lines connecting good leaf states with their subsumed states; Win_s is the set of all states. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Proof. The proof is similar to that of Theorem 3 and we just sketch it here. We show the "only if" part by contrapositive as well. If no winning control strategy exists, then Win_s is empty by Algorithm 3, i.e., no matter what decisions made by the supervisor, there always exist runs ending in bad leaf states. The supervisor only form cycles with negative payoffs so that the limit mean payoff for any run is negative and no supervisor solves the mean payoff decision problem of Problem 2.

The "if" part. If there exists a winning strategy for the supervisor in the FCEIC, then the supervisor achieves nonnegative limit mean payoff since are cycles in the FCEIC$_w$ are with nonnegative payoffs. The supervisor is also live, so it solves the mean payoff decision problem of Problem 2. □

Therefore, we have shown the soundness and completeness of Algorithms 1 and 2. Overall, we transform the mean payoff decision problem for Problem 1 (Problem 2) into a safety game under perfect information and solve it by finding winning control strategies. We end this section with an example.

Example 4. We revisit Example 2 (Example 3) to find the winning regions of the FCEIC following Algorithm 3. Since the good (bad) leaf states in both examples coincide, the winning regions for both examples remain the same. The FCEIC$_w$ is shown in Fig. 4, where green dashed lines connect each good leaf state with the state subsumed by it, indicating that the supervisor always makes the same decision from the two connected states. So the game is extended to be infinite-duration. In building the FCEIC$_w$, shaded states y_{2-3}^e and y_{3-3}^e in Fig. 3 are bad leaf states, thus are pruned by Algorithm 3. Meanwhile, good leaf states y_{2-4}^e and y_{3-4}^e are also removed as they become no longer accessible from the initial state y_0^e after their preceding Z-states z_8^e and z_9^e are removed. That means that the supervisor should not choose γ_2' at y_2^e or γ_3 at y_3^e, otherwise, the environment may choose o_2 at z_8^e or o_3 at z_9^e to reach some bad leaf states and wins the game.

Then we present a winning control strategy indicated by blue lines in Fig. 4. As is seen, the supervisor S issues γ_0 at y_0^e, γ_1 at y_1^e, γ_0 at y_2^e and γ_0 at y_3^e. If S makes those decisions infinitely often, then only cycles with nonnegative payoffs are formed in the supervised system. Finally we show the supervised system under this strategy in Fig. 5. Compared with the original system in Fig. 2, the cycles with a negative payoff have been broken. Then it is easy to verify that the supervised system is live and all infinite runs have a positive limit mean payoff. Thus, S solves the mean payoff decision problem of Problem 1 (Problem 2).

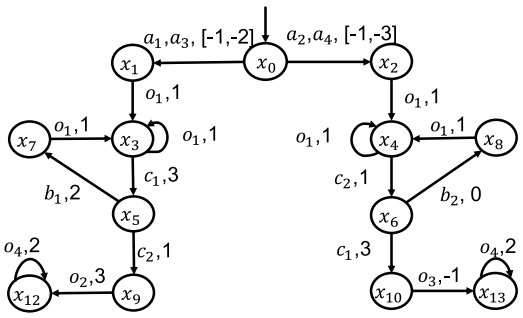


Fig. 5. A supervisor solving the mean payoff decision problem.

6. Mean payoff optimization problems

Based on results of the preceding section, we continue to find the control strategy that optimizes the worst-case limit mean payoff to completely solve Problem 1 and Problem 2 in this section. Our method is inspired by the technique of solving min–max games (Osborne & Rubinstein, 1994), however, additional analysis is necessary here due to the partial observation. As there is no difference between the procedures of synthesizing the optimal control strategies for both problems, we present a uniform approach in the following discussion.

In the FCEIC$_w$, we denote by $Run(F_w)$ the set of runs and $Run_{leaf}(F_w)$ the set of runs ending in a good leaf state, respectively. Given a run $r_f = y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e_{n-1}} y_n^e \in Run(F_w)$ with $y_j^e \preccurlyeq y_n^e$ for some $j < n$ where y_n^e is a leaf state, we know that simple loops with nonnegative payoffs are generated from each state in state estimate $Est(y_j^e)$.

In order to determine the mean payoffs of strings generated by runs in the FCEIC$_w$, we need to know exactly what observable and unobservable events are in the string. However, we only know the observable events from transitions in the FCEIC$_w$ since the unobservable transitions are hidden within each state. For the purpose of explicitly revealing the inner connections between states by unobservable strings *inside* each Y-state or Z-state in the FCEIC$_w$, we introduce a new transition system called the *Energy Inter Connected System* (EICS), which is inspired by the Inter Connected System proposed in Yin and Lafortune (2016a).

Definition 6 (*Energy Inter Connected System (EICS)*)**.** Given the FCEIC$_w$ with respect to G, its corresponding Energy Inter Connected System (EICS) is defined as a tuple: $EICS = (Q^{EICS}, E^{EICS}, f^{EICS}, q_0^{EICS}, Q_l^{EICS})$ where

- $Q^{EICS} \subseteq (Q_Y^F \times X) \cup (Q_Z^F \times X)$ is the state space such that:
  - $(y^e, x) \in Q^{EICS}$ if $y^e \in Q_Y^F$ and $x \in Est(y^e)$;
  - $(z^e, x) \in Q^{EICS}$ if $z^e \in Q_Z^F$ and $x \in Est(I_E((z^e)))$;
- $E^{EICS} = E \cup \Gamma$ is the set of events in the EICS;
- $f^{EICS} : Q^{EICS} \times E^{EICS} \to Q^{EICS}$ is the partial transition function defined as: $\forall \gamma \in \Gamma, \forall e \in E$:
  - $f^{EICS}((y^e, x_1), \gamma) = (z^e, x_2)$ if $x_1 = x_2$ in G and $f_{yz}^F(y^e, \gamma) = z^e$ in the FCEIC$_w$;
  - $f^{EICS}((z^e, x_1), e) = (z^e, x_2)$ if $f(x_1, e) = x_2$ in G and $e \in \Gamma(z^e) \cap E_{uo}$;
  - $f^{EICS}((z^e, x_1), e) = (y^e, x_2)$ if $f(x_1, e) = x_2$ in G, $e \in \Gamma(z^e) \cap E_o$ and $f_{zy}^F(z^e, e) = y^e$ in the FCEIC$_w$;
- $q_0^{EICS} = \{y_0^e, x_0\}$ is the initial state;
- $Q_l^{EICS} = \{(y^e, x) \in Q^{EICS} : y^e \in Q_{lg}^F$ in the FCEIC$_w\}$ is the set of leaf states where no transitions are defined.

Intuitively, the EICS is similar to the structure after parallel composition between the FCEIC$_w$ and G. It explicitly shows both observable and unobservable reaches between and within states of the FCEIC$_w$. A state in the EICS contains a state from the FCEIC$_w$ and a state from G. There are three types of f^{EICS} transitions defined in the EICS. The first type indicates the supervisor's decisions from certain states of the system, so the first component of an EICS state changes from a Y-state to its succeeding Z-state in the FCEIC$_w$ and the second component stays the same. The second type indicates the unobservable reaches within Z-states in the FCEIC$_w$, so the first state component of (z^e, x_1) stays the same and the second component becomes $x_2 = f(x_1, e)$ under $e \in \Gamma(z^e) \cap E_{uo}$. The third type indicates observable reaches between Y-states and Z-states in the FCEIC$_w$, so the first component gets updated from a Z-state to its succeeding Y-state in the FCEIC$_w$ and the second component also gets updated by the enabled observable event. With the EICS built, we are able to explicitly see how strings are generated under control decisions in the FCEIC$_w$.

The leaf states of the EICS contain leaf states of the FCEIC$_w$, which also indicate simple cycles in the FCEIC$_w$. For a leaf state $(y^e, x) \in Q_l^{EICS}$, we are able to track simple loops starting from $x \in Est(y^e)$ by following transitions between $(\tilde{y}^e, x)$ and (y^e, x), where $\tilde{y}^e \preceq y^e$. We define $Lp_{sim}(y^e, x) = \{t \in E^* : \exists r_f = y_0^e \xrightarrow{\gamma_0} z_0^e \xrightarrow{e_0} y_1^e \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e_{n-1}} y^e \in Run(F_w)$, s.t. $\exists j < n, y_j^e \preceq y^e, t \in Str(y_j^e \xrightarrow{\gamma_j} z_j^e \xrightarrow{e_j} \cdots \xrightarrow{\gamma_{n-1}} z_{n-1}^e \xrightarrow{e_{n-1}} y^e), f(x, t) = x\}$ as the set of simple loops starting from x. For a simple loop $t \in Lp_{sim}(y^e, x)$, we define its mean payoff as $V_{sl}(t) = \frac{\omega(t)}{|t|}$.

Furthermore, we define $V_{leaf} : Run_{leaf}(F_w) \to \mathbb{R}$ to characterize the (limit) mean payoff of runs ending in a leaf state of the FCEIC$_w$. If a run r_f ends in a leaf state y^e, we have $V_{leaf}(r_f) = \min_{x \in Est(y^e)} \min_{t \in Lp_{sim}(y^e, x)} V_{sl}(t)$, i.e., the minimum possible mean payoff of all simple loops formed from states in $Est(y^e)$. We take the minimum mean payoff among simple loops to characterize the (limit) mean payoff of the run, since only the cyclic part of a run contributes to the limit mean payoff and the supervisor maximizes the worst-case limit mean payoff. With a slight abuse of notation, we let $V_{leaf}(Last(r_f))$ stand for $V_{leaf}(r_f)$.

Given a pair of strategies $\pi_s \in \Pi_s$ and $\pi_e \in \Pi_e$ in the FCEIC$_w$, we let $r_f(\pi_s, \pi_e)$ be the unique initial run generated under (π_s, π_e) and its last state $Last(r_f(\pi_s, \pi_e)) \in Q_{lg}^F$. Then we define the optimal control strategy which maximizes the worst mean payoffs of runs.

Definition 7 (*Optimal Control Strategy in the FCEIC$_w$*)**.** A winning control strategy π_s^* in the FCEIC$_w$ is optimal if

$$\min_{\pi_e \in \Pi_e} V_{leaf}(r_f(\pi_s^*, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(\pi_s, \pi_e)) \tag{5}$$

Since the FCEIC$_w$ is acyclic and the number of positional strategies for both players are finite, the optimal control strategy always exists. Here we are ready to synthesize a (positional) optimal control strategy from the FCEIC$_w$ and present Algorithm 4. From Definition 7, an optimal control strategy maximizes its mean payoff against the antagonistic environment's strategies, which minimize the supervisor's mean payoff. So the supervisor and the environment play a *min–max game* (Osborne & Rubinstein, 1994) on the FCEIC$_w$, where the supervisor is the maximizer and the environment is the minimizer. In Algorithm 4, we leverage *backward induction* (Osborne & Rubinstein, 1994) to determine an optimal control strategy on the FCEIC$_w$.

First we compute the string mean payoffs from the leaf states of the EICS. Furthermore, it is possible to calculate $V_{leaf}(r_f(\pi_s, \pi_e))$ from the FCEIC$_w$, with the EICS defined. Specifically, the EICS is used to determine the mean payoffs of simple loops from the leaf states of the FCEIC$_w$ in line 5 of Algorithm 4. For a leaf state $(y^e, x) \in Q_l^{EICS}$, we can always find another state $(\tilde{y}^e, x) \in Q^{EICS}$

such that $\tilde{y}^e \preceq y^e$ in the FCIEC$_w$. Then we track f^{EICS} transitions to find both observable and unobservable events between $(\tilde{y}^e, x)$ and $(y^e, x) \in Q_l^{EICS}$. Afterwards, we determine $Lp_{sim}(y^e, x)$ and calculate $V_{sl}(t)$ for each $t \in Lp_{sim}(y^e, x)$. There may be multiple simple loops formed from $x \in Est(y^e)$, with different mean payoffs. Then we calculate $V_{leaf}(y^e)$, the minimum mean payoff of all possible simple loops formed from all states in $Est(y^e)$. $V_{leaf}(y^e)$ is also the minimum possible mean payoff that the supervisor may achieve when state estimate $Est(y^e)$ is reached.

Then we run Procedure *Optimal* to assign a value $V_F(q^e)$ to each state q^e in the FCEIC$_w$. In this procedure, we first determine the values to leaf states in line 6. Next we propagate backwards to determine the values of predecessor states until the root state is assigned a value. Specifically, if the current state is a Z-state, we assign the minimum value of its successor states to it in line 18, since the environment always minimizes the mean payoff of the supervisor. If the current state is a Y-state (not a leaf state), we assign the maximum value of its successor states to it in line 21, since the supervisor always maximizes its mean payoff. This min–max procedure is consistent with Definition 7 where the optimal supervisor maximizes the worst-case payoff it may achieve. The procedure goes on until a value is assigned to the initial state y_0^e of the FCEIC$_w$. Since the FCEIC$_w$ is finite, Algorithm 4 terminates after all states are assigned their values.

When Procedure *Optimal* is implemented, we can assign orders to states in the FCEIC$_w$ so that a state is evaluated after all its successors are evaluated. This is essentially the process of backward induction in solving min–max games (Osborne & Rubinstein, 1994). After obtaining V_F values, we specify the optimal control decisions at Y-states of the FCEIC$_w$, which constitute the optimal control strategy. It is possible that there are multiple optimal control decisions at the current Y-state when some of the successor states have the same V_F value. Then we randomly choose a control decision. Similar min–max search processes were presented in Pruekprasert et al. (2016) and Wu and Lafortune (2016) to synthesize optimal strategies of mean payoff games, for the specific problems discussed in those works.

Algorithm 4 Synthesize an optimal control strategy

Input: the FCEIC$_w$ and the EICS
Output: An optimal strategy π_s for Problem 1 or 2

1: **for** leaf state y^e in FCEIC$_w$ **do**
2: **for** leaf state (y^e, x) in EICS **do**
3: Get $Lp_{sim}(y^e, x)$ following transitions in EICS;
4: **for** $t \in Lp_{sim}(y^e, x)$ **do**
5: Calculate $V_{sl}(t)$;
6: Calculate $V_{leaf}(y^e) = \min_{x \in Est(y^e)} \min_{t \in Lp_{sim}(y^e, x)} V_{sl}(t)$;
7: **for** $q^e \in Q_Y^F \cup Q_Z^F$ **do**
8: $V_F(q^e) = Optimal(q^e)$;
9: **for** $y^e \in Q_Y^F \setminus Q_l^F$ **do**
10: Find one $\gamma \in \Gamma$, s.t. $\exists z^e \in Q_Z^F$, $f_{yz}^F(y^e, \gamma) = z^e$ and $V_F(y^e) = V_F(z^e)$;
11: Return $\pi_s^*(y^e) = \gamma$;
12: **procedure** *Optimal*(q^e)
13: **for** $q^e \in Q_l^F$ **do**
14: $V_F(q^e) = V_{leaf}(q_e)$;
15: Return $V_F(q^e)$;
16: **for** $q_e \in (Q_Y^F \cup Q_Z^F) \setminus Q_l^F$ **do**
17: **if** $q^e \in Q_Z^F$ **then**
18: $V_F(q^e) = \min_{\tilde{q}^e \in Q_Y^F} \{Optimal(\tilde{q}^e) : \exists e_o \in E_o,$ s.t. $f_{zy}^F(q_e, e_o) = \tilde{q}^e\}$;
19: Return $V_F(q^e)$;
20: **if** $q^e \in Q_Y^F$ **then**
21: $V_F(q^e) = \max_{\tilde{q}^e \in Q_Z^F} \{Optimal(\tilde{q}^e) : \exists \gamma \in \Gamma,$ s.t. $f_{yz}^F(q^e, \gamma) = \tilde{q}^e\}$;
22: Return $V_F(q^e)$;

After obtaining an optimal positional control strategy in the FCEIC$_w$, we again let the supervisor make the same decision from the current Y-state as from the state subsumed by it. In this way, the game is extended to be infinite and we obtain a supervisor that perpetually issues control decisions to generate a live system. Intuitively, the supervisor always traverses the simple cycle with the highest mean payoff since alternating between cycles with different mean payoffs does not result in a higher mean payoff. Hence, a positional strategy is sufficient to solve Problem 1 (Problem 2), summarized in the following theorem.

Theorem 5. *If π_s^* is returned by Algorithm 4, then we can extend π_s^* to a supervisor S^* that solves Problem 1 (Problem 2).*

Proof. By Algorithm 4, for every leaf state $y^e \in Q_{lg}^F$, $V_{leaf}(y^e) = \min_{x \in Est(y^e)} \min_{t \in Lp_{sim}(y^e, x)} V_{sl}(t)$. Let string $t^*(y^e)$ be such that $V_{sl}(t^*(y^e)) = \min_{x \in Est(y^e)} \min_{t \in Lp_{sim}(y^e, x)} V_{sl}(t) = V_{leaf}(y^e)$. Suppose that a Z-state z^e can reach k leaf states $y_1^e, y_2^e, \ldots, y_k^e \in Q_{lg}^F$, i.e., $\forall i \le k$, $\exists e_i \in E_o$, s.t. $f_{zy}^F(z^e, e_i) = y_i^e$. Thus we know:

$$V_F(z^e) = \min\{V_F(y_1^e), \ldots, V_F(y_k^e)\} = \min\{V_{sl}(t(y_1^e)), \ldots, V_{sl}(t(y_k^e))\}$$

Let string $t^*(z^e)$ be such that $V_{sl}(t^*(z^e)) = \min\{V_{sl}(t(y_1^e)), \ldots, V_{sl}(t(y_k^e))\}$ so it has the minimum loop mean payoff. The environment still locates the string whose simple loop has the minimum mean payoff, by evaluating $V_{leaf}(y^e)$. From the EICS, we explicitly know which cyclic string has the minimum loop mean payoff.

Suppose that one predecessor state of z^e is $\tilde{y}^e$ and $\tilde{y}^e$ has successor states $z_1^e, \ldots, z_m^e$ (z^e is one of them). Then the supervisor maximizes its V_F value among the successor states of $\tilde{y}^e$, i.e., we let $V_F(\tilde{y}^e) = \max_{z_i^e} V_F(z_i^e)$ where $i \le m$. Since $V_F(z_i^e)$ is the minimum mean payoff of some simple loop, $V_F(\tilde{y}^e)$ still maximizes the minimum mean payoffs of simple loops obtained from some leaf states in the FCEIC$_w$. Thus, the supervisor loses no information when making decisions by evaluating $V_F(z^e)$. By Algorithm 4, the supervisor chooses the control decision that maximizes $V_F(z_i^e)$. Then we repeat the same argument backwards to the root state. In this way, we show that by evaluating the V_F values for Y-states or Z-states, the supervisor correctly performs maximization among V_F values from its successors and the environment correctly performs minimization.

Finally $V_F(y_0^e) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(\pi_s, \pi_e))$ holds. Then we extend π_s to a supervisor S^* by the same argument as in the proof of Theorem 3, i.e., imagine that each leaf state in the FCEIC$_w$ is "merged" with the state subsumed by it and let the supervisor make the same decision whenever the same state estimate is reached. By checking the transitions in the EICS, we find a run in S^*/G leading to $V_F(y_0^e) = \inf_{r \in Run_{inf}(S^*/G)} V_{mp}(r) = \sup_{S \in \mathbb{S}} \inf_{r \in Run_{inf}(S/G)} V_{mp}(r)$. So S^* solves Problem 1 (Problem 2). □

We analyze the complexity for Algorithm 4, which essentially performs a minimax search. Results in Du and Pardalos (2013) show that the time complexity of the minimax search is $O(b^n)$ and the space complexity is $O(bn)$, where b is the maximum number of choices at each point in the search tree and n is the depth of the tree. For Algorithm 4, we have $b = \max\{2^{|E_c|}, |E_o|\}$ and $n = 2 \cdot 2^{|X|} + 1$ in the worst case. Here $2^{|E_c|}$ is the maximum number of control decisions at a state and there are at most $2 \cdot 2^{|X|} + 1$ states between any two states in the FCEIC$_w$. Thus we obtain the complexity bounds for Algorithm 4.

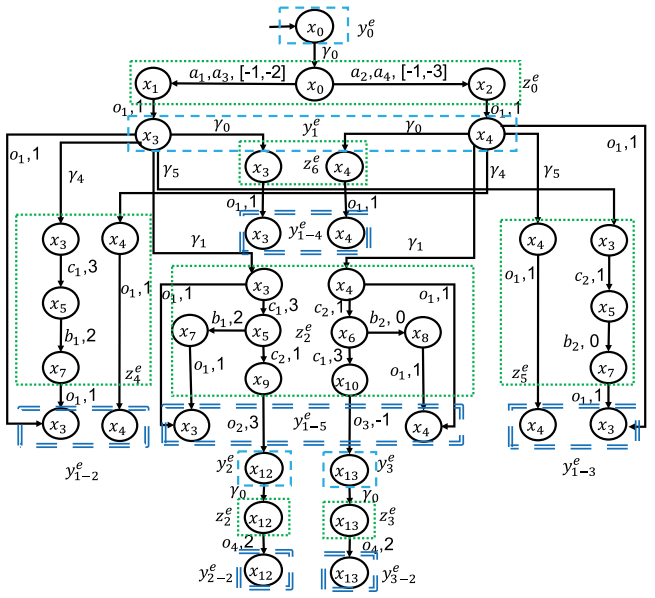


Fig. 6. The energy inter-connected system w.r.t. the FCEIC$_w$ in Example 4. The blue and green dashed rectangles correspond to the Y-states and Z-states in the FCEIC$_w$, respectively. The leaf states are marked in double blue lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

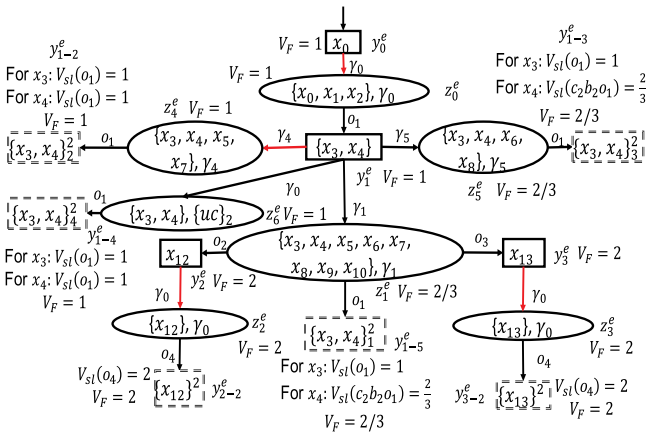


Fig. 7. Optimal decisions of the supervisor at each Y-state are indicated in red; the V_F values for each state of the FCEIC$_w$ are also shown in the figure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We further discuss the structure of the optimal strategy from Algorithm 4. Given strategies $(\pi_s, \pi_e) \in \Pi_s \times \Pi_e$ and an initial run $r'_f \in Run(F_w)$, let $r_f(r'_f; \pi_s, \pi_e)$ be the run that has "prefix" r'_f, continues under (π_s, π_e) and ends in a leaf state of the FCEIC$_w$. Formally, $r_f(r'_f; \pi_s, \pi_e) = r'_f \xrightarrow{\gamma_1} z^e_1 \xrightarrow{e_1} y^e_2 \xrightarrow{\gamma_2} \cdots \xrightarrow{e_n} y^e_n$ where $y^e_n \in Q^F_{lg}$, $\gamma_1 = \pi_s(r'_f)$, $e_1 = \pi_e(r'_f \xrightarrow{\gamma_1} z^e_1)$ and $\gamma_i = \pi_s(r'_f \xrightarrow{\gamma_1} z^e_1 \xrightarrow{e_1} y^e_2 \xrightarrow{\gamma_2} \cdots \xrightarrow{e_i} y^e_i)$, $e_i = \pi_e(r'_f \xrightarrow{\gamma_1} z^e_1 \xrightarrow{e_1} y^e_2 \xrightarrow{\gamma_2} \cdots \xrightarrow{\gamma_i} z^e_i)$ for all $2 \le i \le n$. We write $r_f(r'_f; \pi_s, \pi_e)$ as $r_f(Last(r'_f); \pi_s, \pi_e)$ since both players' decisions only depend on their current positions. Now we are ready to show that the optimal control strategy enjoys a structural property resembling *subgame perfect equilibrium* in game theory (Osborne & Rubinstein, 1994) and *Bellman's optimality principle* in dynamic programming (Bertsekas, 2012).

Proposition 2. *Let π^*_s be a control strategy returned by Algorithm 4, then for any initial run $r'_f \in Run(F_w)$, we have:*

$$\min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi^*_s, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi_s, \pi_e)) \tag{6}$$

Proof. See the Appendix. □

This proposition illustrates the structure of the optimal control strategy obtained from Algorithm 4. If the supervisor follows the strategy indicated by Algorithm 4 from its current position, then its onward decisions still constitute an optimal strategy in the remaining game, which can be viewed as a "subgame" (Osborne & Rubinstein, 1994). In other words, the supervisor has no incentive to deviate from its optimal strategy given that the environment does its best to minimize the supervisor's mean payoff. As seen from the proof, this result is due to the backward induction process of maximization and minimization in Algorithm 4.

Example 5. We revisit Example 4 and find an optimal control strategy to solve Problems 1 and 2 completely. First we obtain the EICS w.r.t. the FCEIC$_w$ in Fig. 6. For simplicity of the graph, we still preserve the state names from G and use dashed rectangles to indicate the Y-states or Z-states of the FCEIC$_w$. For example, the top green dashed rectangle corresponds to three states in the EICS, i.e. (z^e_0, x_0), (z^e_0, x_1) and (z^e_0, x_2) where $Est(I_E(z^e_0)) = \{x_0, x_1, x_2\}$. Specifically, blue and green dashed rectangles correspond to the Y-states and Z-states of the FCEIC$_w$ respectively. As is seen, the EICS is a tree-like structure whose leaf states (y^e_{1-2}, x_3), (y^e_{1-2}, x_4), (y^e_{1-3}, x_3), (y^e_{1-3}, x_4), (y^e_{1-4}, x_3), (y^e_{1-4}, x_4), (y^e_{1-5}, x_3), (y^e_{1-5}, x_4), (y^e_{2-2}, x_{12}) and (y^e_{3-2}, x_{13}) are marked in double dark blue lines.

With the EICS built, we proceed to find the optimal control strategy by Algorithm 4. We start by calculating the values of V_{leaf} for each leaf state of the FCEIC$_w$. For example, in the EICS, there are two simple cycles between Y-states y^e_1 and y^e_{1-2}, i.e., $x_3 \xrightarrow{o_1} x_3$ and $x_3 \xrightarrow{c_1} x_5 \xrightarrow{b_1} x_7 \xrightarrow{o_1} x_3$. Then we obtain $V_{sl}(o_1) = 1$ (for x_3), $V_{sl}(c_1 b_2 o_1) = 2$ and $V_{sl}(o_1) = 1$ (both for x_4). Therefore, $V_F(y^e_{1-2}) = \min\{1, 2\} = 1$. Similarly, we obtain the V_T values for other leaf states in the FCEIC$_w$, which are shown in Fig. 7. Next, we apply backward induction from the leaf states until the root state, then determine an optimal control strategy. In this process, we always choose to minimize at Z-states and maximize at Y-states. By Algorithm 4, we know $V_F(z^e_1) = \min\{2, \frac{2}{3}\} = \frac{2}{3}$ and $V_F(z^e_4) = V_F(z^e_6) = 1$. Thus, we have the supervisor's decisions at each Y-state, which are indicated by solid red lines in Fig. 7. An optimal supervisor enables c_1 upon observing o_1, as shown in Fig. 8. The worst limit mean payoff is 1 in the supervised system. Actually, it is also optimal for the supervisor to disable both c_1 and c_2 at y^e_1, which yields the same worst case limit mean payoff.

Notice that choosing γ_4 or γ_6 at y^e_1 is optimal in the sense that the environment also follows its "optimal strategy" to minimize the supervisor's limit mean payoff. If the supervisor deviates from γ_4 or γ_0 and chooses γ_1 at y^e_1, then the environment may choose o_1 at z^e_1, which leads to leaf state y^e_{1-5} and a potentially lower limit mean payoff $\frac{2}{3}$. Interestingly, if the environment also deviates from choosing o_1 from z^e_1 by choosing o_2 or o_3, then the supervisor should choose γ_0 at y^e_2 and y^e_3, which yields a better limit mean payoff for the supervisor compared with the case of choosing γ_4 at y^e_1. Those two decisions are optimal in the "subgame" given that y^e_2 or y^e_3 is reached and viewed as starting points of the subgame. This is consistent with Proposition 2.

7. Conclusion

This work studied infinite horizon optimal supervisory control under partial observation for the first time in discrete event

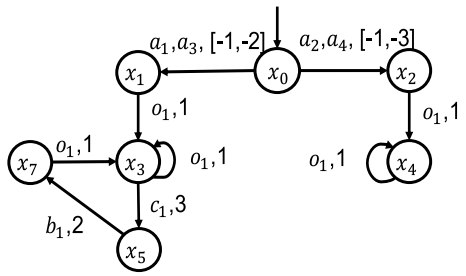


Fig. 8. An optimal supervisor solving Problems 1 and 2.

systems. We considered two optimal control scenarios, then formulated two supervisory control problems correspondingly. To this end, we defined energy information states and the First Cycle Energy Inclusive Controller (FCEIC) for each problem. Based on the FCEIC, each problem was transformed into a finite game with perfect information and proper objectives. As an intermediate solution step, we solved the mean payoff decision problems via safety games. Finally we solved a min–max game to find the optimal control strategy among partial solutions. For future work, it would be of interest to explore infinite horizon optimal supervisory control with other quantitative performance objectives and under partial observation. In addition, it would also be worthwhile to investigate the application of the theoretical framework developed herein on specific engineering platforms such as the power management system of electric hybrid vehicles.

Appendix. Proofs of propositions

Proof of Proposition 1:

Proof. Proof by induction. Consider the observable string $t = e_1 \cdots e_{n-1}$ $(n \geq 1)$. We also use the notations ρ_k and ρ'_k from Definition 5 in the following discussion.

Induction Basis: $n = 1$ and consider q_1^e or $q_1^e \xrightarrow{\gamma_1} q_1^{ae}$. The result obviously holds for single state q_1^e and also holds for $q_1^e \xrightarrow{\gamma_1} q_1^{ae}$ by Definition 5 and the definition of γ-successor.

Inductive Hypothesis: we assume the lemma holds when $n = k$, i.e., for control-observation sequences ρ_k and ρ'_k.

Induction Step: when $n = k+1$, consider ρ_{k+1} and ρ'_{k+1}. First, q_{k+1}^e is an e_k-successor or q_k^{ae}. Let $Est(I_E(q_k^{ae})) = q'_k$ and $Est(q_{k+1}^e) = q_{k+1}$, then $\forall x \in q_{k+1}$,

$$Lev(q_{k+1}^e, x) = \min_{x'}\{Lev(q_k^{ae}, x') + \omega(e_k) : \exists x' \in q'_k,\ \text{s.t.}\ f(x', e_k) = x\}$$

By the inductive hypothesis and Definition 5, we have:

$$\begin{aligned} Lev(q_{k+1}^e, x) &= \min_{x'} \min_{s'_k}\{\omega(s'_k) + \omega(e_k) : \exists \tilde{x} \in Est(q_1^e), s'_k \in Str(\rho'_k) \\ &\quad \text{s.t.}\ f(\tilde{x}, s'_k) = x'\} \\ &= \min_{s_{k+1}}\{\omega(s_{k+1}) : \exists \tilde{x} \in Est(y_1^e), s_{k+1} \in Str(\rho_{k+1})\ \text{s.t.} \\ &\quad s_{k+1} = s'_k e_k, f(\tilde{x}, s_{k+1}) = x\} \end{aligned}$$

Then q_{k+1}^{ae} is a γ_{k+1}-successor of q_{k+1}^e. Let $Est(q_{k+1}^e) = q_{k+1}$ and $Est(I_E(q_{k+1}^{ae})) = q'_{k+1}$, so $\forall x' \in q'_{k+1}$,

$$\begin{aligned} Lev(q_{k+1}^{ae}, x') &= \min_{\xi_{k+1}}\{Lev(q_{k+1}^e, x) + \omega(\xi_{k+1}) : \exists x \in q_{k+1}, \\ &\quad \xi_{k+1} \in (E_{uo} \cap \gamma_{k+1})^*\ \text{s.t.}\ f(x, \xi_{k+1}) = x'\} \end{aligned}$$

From what we just proved,

$$\begin{aligned} Lev(q_{k+1}^e, x') &= \min_{\xi_{k+1}}\{Lev(q_{k+1}^e, x) + \omega(\xi_{k+1}) : \exists x \in q_{k+1}, \\ &\quad \xi_{k+1} \in (E_{uo} \cap \gamma_{k+1})^*\ \text{s.t.}\ f(x, \xi_{k+1}) = x'\} \\ &= \min_{s'_{k+1}}\{\omega(s'_{k+1}) : \exists \tilde{x} \in Est(q_1^e), s'_{k+1} \in Str(\rho'_{k+1})\ \text{s.t.} \\ &\quad s'_{k+1} = s_{k+1}\xi_{k+1}, f(\tilde{x}, s'_{k+1}) = x'\} \end{aligned}$$

Thus the result holds at $k+1$, completing the proof. □

Proof of Proposition 2:

Proof. By definition, the FCEIC$_w$ is an acyclic structure and the depth of its runs is thus bounded. In the FCEIC$_w$, there exists a positive integer m such that every leaf state can be reached within m steps from the initial state. Then we prove this proposition by induction on the number of steps for an initial run to reach leaf states of the FCPEC$_w$. In other words, we show that $V_F(Last(r'_f)) = \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi_s^*, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi_s, \pi_e))$ for any initial run r'_f.

Induction Basis: Consider the case when the last state of r'_f is a leaf states in the FCPEC$_w$. Then this proposition becomes Theorem 5, thus, it naturally holds.

Inductive Hypothesis: Suppose that the result holds for any r'_f that reaches leaf states within at most k steps, where $k \leq m-2$ for some integer $m > 2$. In addition, the function *Optimal* in the algorithm assigns $V_F(Last(r'_f)) = \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi_s^*, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(r'_f; \pi_s, \pi_e))$ to the last state of r'_f.

Induction Step: Consider r'_f that reaches leaf states within at most $k+2$ steps. Suppose that $Last(r'_f) = Last_Y(r'_f) = y'^e$. We know that there exists $z^e = f_{yz}^F(y'^e, \gamma)$ for some $\gamma \in \Gamma$ and specifically, $\tilde{z}^e = f_{yz}^F(y'^e, \gamma^*)$ for $\gamma^* = \pi_s^*(y'^e, \gamma^*)$. Thus, succeeding Z-state $z^e = f_{yz}^F(y'^e, \gamma)$ of y'^e reaches a leaf state within at most $k+1$ steps. By Algorithm 4, $V_F(y'^e) = V_F(\tilde{z}^e) = \max_{z^e} V_F(z^e)$. Also some f_{zy}^F transitions are defined from z^e and lead to succeeding Y-state y^e which reaches the leaf states within at most k steps. By the inductive hypothesis, $\min_{\pi_e \in \Pi_e} V_{leaf}(r_f(y^e; \pi_s^*, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(y^e; \pi_s, \pi_e))$ for any r'_f with $Last(r'_f) = y^e$. Again from Algorithm 4, we know:

$$\begin{aligned} V_F(z^e) &= \min_{y^e} V_F(y^e) = \min_{y^e} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(y^e; \pi_s^*, \pi_e)) \\ &= \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(z^e; \pi_s^*, \pi_e)) = \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(z^e; \pi_s, \pi_e)) \end{aligned}$$

thus the result holds for runs whose last states reach the leaf states of the FCEIC$_w$ within $k+1$ steps. Furthermore, we have:

$$\begin{aligned} V_F(y'^e) &= \max_{z^e} V_F(z^e) = \max_{z^e} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(z^e; \pi^*, \pi_e)) \\ &= \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(y'^e; \pi^*, \pi_e)) \\ &= \max_{\pi_s \in \Pi_s} \min_{\pi_e \in \Pi_e} V_{leaf}(r_f(y'^e; \pi_s, \pi_e)) \end{aligned}$$

Therefore the result holds for $k+2$, completing the proof. □

References

Alves, M. V. S., Carvalho, L. K., & Basilio, J. C. (2016). New algorithms for verification of relative observability and computation of supremal relatively observable sublanguage. *IEEE Transactions on Automatic Control, 62*(11), 5902–5908.

Alves, M. V. S., da Cunha, A. E. C., Carvalho, L. K., Moreira, M. V., & Basilio, J. C. (2019). Robust supervisory control of discrete event systems against intermittent loss of observations. *International Journal of Control*, 1–13.

Aminof, B., & Rubin, S. (2017). First-cycle games. *Information and Computation, 254*, 195–216.

Apt, K. R., & Grädel, E. (2011). *Lectures in game theory for computer scientists*. Cambridge University Press.

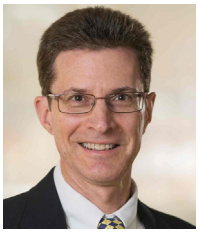
Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. MIT press.
Başar, T., & Bernhard, P. (2008). *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer.
Bertsekas, D. P. (2012). *Dynamic programming and optimal control*. Athena Scientific.
Cai, K., Zhang, R., & Wonham, W. M. (2015). Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Transactions on Automatic Control*, *60*(3), 659–670.
Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems* (2nd ed.). Springer.
Du, D., & Pardalos, P. M. (2013). *Minimax and applications*. Springer.
Giua, A., Seatzu, C., & Basile, F. (2004). Observer-based state-feedback control of timed Petri nets with deadlock recovery. *IEEE Transactions on Automatic Control*, *49*(1), 17–29.
Gu, C., Wang, X., Li, Z., & Wu, N. (2018). Supervisory control of state-tree structures with partial observation. *Information Sciences*, *465*, 523–544.
Han, X., Chen, Z., & Su, R. (2019). Synthesis of minimally restrictive optimal stability-enforcing supervisors for nondeterministic discrete event systems. *Systems & Control Letters*, *123*, 33–39.
Hunter, P., Pauly, A., Pérez, G. A., & Raskin, J.-F. (2018). Mean-payoff games with partial observation. *Theoretical Computer Science*, *735*, 82–110.
Ji, Y., Yin, X., & Lafortune, S. (2018). Mean payoff supervisory control under partial observation. In *Proceedings of the 57th IEEE conference on decision and control* (pp. 3981–3987).
Ji, Y., Yin, X., & Lafortune, S. (2019a). Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, *108*, Article 108476.
Ji, Y., Yin, X., & Lafortune, S. (2019b). Supervisory control under local mean payoff constraints. In *58th IEEE conference on decision and control* (pp. 1043–1049).
Komenda, J., & Masopust, T. (2017). Computation of controllable and coobservable sublanguages in decentralized supervisory control via communication. *Discrete Event Dynamic Systems: Theory and Applications*, *27*(4), 585–608.
Krishnamurthy, V. (2016). *Partially observed Markov decision processes: From filtering to controlled sensing*. Cambridge University Press.
Levy, A. (2002). *Basic set theory, Vol. 13*. Courier Corporation.
Lin, L., Masopust, T., Wonham, W. M., & Su, R. (2019). Automatic generation of optimal reductions of distributions. *IEEE Transactions on Automatic Control*, *64*(3), 896–911.
Madani, O., Hanks, S., & Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, *147*(1–2), 5–34.
Malikopoulos, A. A. (2014). Supervisory power management control algorithms for hybrid electric vehicles: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *15*(5), 1869–1885.
Marchand, H., Boivineau, O., & Lafortune, S. (2002). On optimal control of a class of partially observed discrete event systems. *Automatica*, *38*(11), 1935–1943.
Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Massachusetts Institute of Technology press.
Pantelic, V., & Lawford, M. (2012). Optimal supervisory control of probabilistic discrete event systems. *IEEE Transactions on Automatic Control*, *57*(5), 1110–1124.
Pérez, G. A. (2017). The fixed initial credit problem for partial-observation energy games is ack-complete. *Information Processing Letters*, *118*, 91–99.
Pruekprasert, S., & Ushio, T. (2016a). 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, S., & Ushio, T. (2016b). Optimal stabilizing supervisor of quantitative discrete event systems under partial observation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *99*(2), 475–482.
Pruekprasert, S., & Ushio, T. (2017). Supervisory control of partially observed quantitative discrete event systems for fixed-initial-credit energy problem. *IEICE Transactions on Information and Systems*, *100*(6), 1166–1171.
Pruekprasert, S., Ushio, T., & Kanazawa, T. (2016). Quantitative supervisory control game for discrete event systems. *IEEE Transactions on Automatic Control*, *61*(10), 2987–3000.
Puterman, M. L. (2005). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
Rackoff, C. (1978). The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, *6*(2), 223–231.
Schmidt, K. W., & Breindl, C. (2014). A framework for state attraction of discrete event systems under partial observation. *Information Sciences*, *281*, 265–280.
Sengupta, R., & Lafortune, S. (1998). An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, *36*(2), 488–541.
Shu, S., & Lin, F. (2015). Supervisor synthesis for networked discrete event systems with communication delays. *IEEE Transactions on Automatic Control*, *60*(8), 2183–2188.
Shu, S., & Lin, F. (2017). Predictive networked control of discrete event systems. *IEEE Transactions on Automatic Control*, *62*(9), 4698–4705.
Takai, S., & Ushio, T. (2003). Effective computation of an L_m (g)-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, *49*(3), 191–200.
Wonham, W. M., & Cai, K. (2019). *Supervisory control of discrete-event systems*. Springer.
Wu, Y.-C., & Lafortune, S. (2016). Synthesis of optimal insertion functions for opacity enforcement. *IEEE Transactions on Automatic Control*, *61*(3), 571–584.
Yin, X., & Lafortune, S. (2016a). Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, *61*(5), 1239–1254.
Yin, X., & Lafortune, S. (2016b). A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, *61*(8), 2140–2154.
Yin, X., & Lafortune, S. (2017). Synthesis of maximally-permissive supervisors for the range control problem. *IEEE Transactions on Automatic Control*, *62*(8), 3914–3929.
Zwick, U., & Paterson, M. (1996). The complexity of mean payoff games on graphs. *Theoretical Computer Science*, *158*(1–2), 343–359.

Yiding Ji received the Bachelor of Engineering degree of Electrical Engineering and Automation from Tianjin University, China, in 2014, the Master of Science degree and the Ph.D degree of Electrical and Computer Engineering from the University of Michigan, United States, in 2016 and 2019, respectively. From 2019 to 2020, he worked as a postdoc researcher at Division of Systems Engineering, Boston University, United States.

His research interests include discrete event systems, formal methods, control systems, game theory and cyber security. He is now a member of IEEE and the IEEE Control Systems Society Technical Committee on Discrete Event Systems.

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, control of discrete event systems, model based fault diagnosis, security and their applications to cyber and cyber–physical systems. Dr. Yin received the Outstanding Reviewer Awards from Automatica, the IEEE Transactions on Automatic Control and the Journal of Discrete Event Dynamic Systems. Dr. Yin also received the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016. He is the co-chair of the IEEE Control Systems Society Technical Committee on Discrete Event Systems.



Stéphane Lafortune received the B.Eng degree from Ecole Polytechnique de Montréal in 1980, the M.Eng degree from McGill University in 1982, and the Ph.D degree from the University of California at Berkeley in 1986, all in electrical engineering. Since September 1986, he has been with the University of Michigan, Ann Arbor, where he is a Professor of Electrical Engineering and Computer Science. Lafortune is a Fellow of the IEEE (1999) and of IFAC (2017). He received the Presidential Young Investigator Award from the National Science Foundation in 1990 and the George S. Axelby Outstanding Paper Award from the Control Systems Society of the IEEE in 1994 (for a paper co-authored with S.-L. Chung and F. Lin) and in 2001 (for a paper co-authored with G. Barrett). Lafortune's research interests are in discrete event systems and include multiple problem domains: modeling, diagnosis, control, optimization, and applications to computer and software systems. He co-authored, with C. Cassandras, the textbook *Introduction to Discrete Event Systems - Second Edition* (Springer, 2008). Lafortune is Editor-in-Chief of the Journal of Discrete Event Dynamic Systems: Theory and Applications.