Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-4 (2020) 390–396

Local Mean Payoff Supervisory Control under Partial Observation ⋆

Yiding Ji * **Xiang Yin** ** **Wei Xiao** *

* *Division of Systems Engineering, Boston University, Boston, MA, United States (e-mail: jiyiding@bu.edu, xiaowei@bu.edu).*
** *Department of Automation, Shanghai Jiao Tong University, Shanghai, China (e-mail: yinxiang@sjtu.edu.cn).*

Abstract: The problem under investigation in this work is local mean payoff supervisory control of partially observed discrete event systems. The system is modeled as a weighted finite state automaton and weight flows are generated with transitions. The local mean payoff over a finite number of events may serve as a measure of stability or robustness of the weight flows. The range of events to evaluate the local mean payoff is termed a window, which slides along transitions. The window is called fuzzy due to the presence of unobservable events. A supervisor is designed to ensure that the mean payoff within each fuzzy window always lies in certain interval. In addition, qualitative properties like safety and liveness are also required. Then the partial observation supervisory control problem is transformed to a two-player safety game on the properly defined windowed bipartite transition system. By analyzing the game, we propose a method to synthesize supervisors that provably solve the original supervisory control problem.

Keywords: Discrete event systems, supervisory control, partial observation, safety game

1. INTRODUCTION

Supervisory control in the context of discrete event systems (DES) is a classic topic. The plant under control is usually modeled as a finite discrete structure and a specification is given as the desired behavior of the plant. The supervisor restricts the behavior of the plant by disabling some events so that the specification is achieved Cassandras and Lafortune [2008], Wonham and Cai [2019].

When the system dynamics is not perfectly monitored, the problem of supervisory control under partial observation naturally arises. This topic has been thoroughly discussed in the literature, see, e.g., Alves et al. [2019], Shu and Lin [2015], Yin and Lafortune [2016a,b], Komenda and Masopust [2017], Meira-Góes et al. [2017], Li and Takai [2019], Ricker et al. [2017], Mohajerani et al. [2017], Ma et al. [2018], Rashidinejad et al. [2018], Yin and Lafortune [2017], Yin [2017], Lin et al. [2019], Wu et al. [2019], Wang and Pajic [2019] for some recent results.

In many engineering applications, the system may generate or consume certain resources with its operation. It is thus essential to maintain a *stable* rate of resource generation/consumption. Under the framework of DES, we model the system as a weighted automaton where the mean weight/payoff of events over a finite number of transitions reflect the rate of resource change. Suppose two weight sequences are generated by the system: $6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0 \cdots$ (one 6 every 6 transitions) and $1, 1, \cdots$ (always 1). Asymptotically, they have the same limit mean payoff 1. However, if the mean payoff is evaluated every three transitions, then it is either 0 or 2 for the first sequence while it is always 1 for the second. We say that the second sequence is more *stable* since the weight fluctuates less. It is also more *robust* against stealth packet drop attacks as the local mean payoff changes less when certain values are randomly excluded from calculation.

Intuitively, we imagine there is a "window" of certain length, where the "local" mean payoff is calculated. The window is sliding with new event occurrences and called *fuzzy* due to unobservable events. Our goal is to guarantee that mean payoffs within fuzzy windows of a *fixed* number of observable events are always no less than a given threshold. Additionally, the system should keep operating while avoid certain bad states. Motivated by this, we discuss local mean payoff supervisory control under partial observation for the first time in DES and formulate the *supervisory control problem under desirable fuzzy windows*.

To tackle the challenge of partial observation, *windowed information states* are introduced to incorporate sufficient information on state estimate and local mean payoff. With the supervisor's decisions and event occurrences, windowed information states are updated and the *information flow* is formed. Then we introduce the *windowed bipartite transition system (WBTS)* and transform the supervisory control problem to a two-player *safety game*. After that, an algorithm is developed to build the "largest" WBTS which contains candidate supervisors to solve our proposed supervisory control problem. Finally, supervisors are synthesized from the WBTS and they are shown to be sound.

Our work leverages some results from algorithmic game theory in computer science Apt and Grädel [2011], espe-

⋆ The second author is supported by the National Natural Science Foundation of China under grants (61803259 and 61833012) and by Shanghai Jiao Tong University Scientific and Technological Innovation Funds.

10.1016/j.ifacol.2021.04.066

cially mean payoff games, see, e.g., Hunter et al. [2018]. Different from reactive synthesis, we discuss a supervisory control problem where there is a plant to be controlled, while the supervisor may choose to enable multiple events at one time. Several works study DES related problems under quantitative game frameworks, like Pruekprasert et al. [2016], Pruekprasert and Ushio [2017], Ji et al. [2018, 2019a,b]. However, our problem setting is significantly different: Pruekprasert et al. [2016] and Ji et al. [2018] focus on supervisory control with a limit (global) mean payoff objective; Ji et al. [2019b] deals with local mean payoff supervisory control under full observation; Ji et al. [2019a] and Pruekprasert and Ushio [2017] solve corresponding problems under the setting of energy games. Besides, our solution methodology is also incomparable with theirs.

The rest of the work is organized as follows. Section 2 introduces the system model. Section 3 formulates the key problem of this work. Section 4 presents the information-flow analysis and solves the proposed problem under the framework of safety game. Finally, Section 5 concludes the paper and mentions potential future research directions.

2. SYSTEM MODEL

We model a quantitative discrete event system 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 and $\omega : E \to \mathbb{Z}$ is the weight function and the weight may be interpreted as the amount of resource associated with the event. A positive number indicates increase of resource while a negative number indicates decrease. The domain of f can be extended to $X \times E^*$ in the standard manner described in Cassandras and Lafortune [2008] 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". The function ω is additive and its domain is extended to E^* by letting $\omega(\epsilon) = 0$, $\omega(se_o) = \omega(s)+\omega(e_o)$ for all $s \in E^*$ and $e \in E$. We denote by W the maximum absolute value of event weights in G, i.e., $W = \max_{e\in E} |\omega(e)|$.

Given $s = e_1e_2 \cdots e_n \in E^*$, for some $1 \leq j < m < n$, we call $e_j \cdots e_{m+1}$ a *substring* of s and denote it by $s(j, m)$. We also call $e_j \cdots e_n$ a *suffix*, and $e_1 \cdots e_j$ a *prefix* of s. Given $s, t \in E^*$, we write $s \preceq u$ if string s is a prefix of u.

With the occurrence of events, we may imagine some *weight flows* are generated. For string $s \in \mathcal{L}(G)$, its *(accumulative) weight/payoff* is $\omega(s)$ while its *mean weight/payoff* is $\frac{1}{n}\omega(s)$. The range of events (horizon) to evaluate the mean payoff may be viewed as a "window", which is sliding when new events occur. When the window's length approaches infinity, the limit mean payoff characterizes asymptotic performance of the weight flows. The "local" mean weight may serve as a measure of stability or robustness for the weight flows. It should remain relatively "smooth". Given $s = e_1e_2 \cdots e_n$ and some integer $v \in \mathbb{Z}$, it is desirable to have $\frac{1}{n}\sum_{i=1}^{n} \omega(e_i) \geq v$. In addition, we may subtract vector v from every $\omega(e_i)$ and equivalently evaluate whether $\frac{1}{n}\sum_{i=1}^{n}(\omega(e_i) - v) \geq 0$ holds. Thus, we assume $v = 0$ without loss of generality.

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. Furthermore, the system is *partially observed*. To this end, E is also partitioned as $E = E_o \cup E_{uo}$ where E_o is the set of observable events and E_{uo} is the set of unobservable events. Additionally, 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 domain of P can be extended to 2^{E^*} naturally.

Given G, for $x_1, x_2 \in X$ and $e \in E$, we write $x_1 \xrightarrow{e} x_2$ if $f(x_1, e) = x_2$, for simplicity. A *run* in G is a sequence of states and events: $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} x_n$ and we denote the set of runs in G by $Run(G)$. We call a run *initial* if its initial state is the initial state of the system. In addition, we also write $x_1 \xrightarrow{s} x_2$ for $s \in E^*$ if $f(x_1, s) = x_2$.

By the way, $r = x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} x_n$ forms a *cycle* if $x_1 = x_n$. If r is a cycle, the corresponding string $e_1e_2 \cdots e_{n-1}$ forms a *loop*. In this work, we assume that *no loops are purely composed of unobservable events* in G.

We consider *safety* in terms of the state space of G and let $X_{us} \subset X$ be the set of unsafe states to be avoided. We also consider the (weak) *liveness* property while do not involve marked states in this work. 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. $su \in \mathcal{L}(G)$. Thus, a transition is always defined out of a state in G. This requirement is without loss of generality as it can be relaxed by adding observable self-loops at terminal states where no active events are defined.

The system is controlled by a *supervisor* which dynamically enables and disables events while only has partial observation of the system. Formally, a supervisor is a function $S : P(\mathcal{L}(G)) \to \Gamma$ and we denote by $\mathbb{S}$ the set of supervisors. A control decision $\gamma \in 2^E$ is the set of events enabled by the supervisor. It is further called *admissible* if $E_{uc} \subseteq \gamma$, i.e., no uncontrollable event is disabled. We denote by Γ the set of admissible control decisions and only consider them in this work. We 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 supervisor is called *safe* and *live* if its supervised system is safe and live.

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 control decision $\gamma \subseteq E$, 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 a tuple: $Obs(G) = (X_{obs}, E_o, \delta, x_{obs,0})$ 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 weight function is omitted here. An observer state is also termed a *(current) state estimate* of the system.

3. PROBLEM FORMULATION

In this section, we introduce some concepts and formulate the *supervisory control problem under desirable fuzzy windows*. The goal is to design a supervisor to achieve both

the qualitative objectives, i.e., safety and liveness, as well as the quantitative objective of local mean payoff.

The supervisor only monitors observable events, thus evaluates the local mean payoff based on the length of observed events. Intuitively, the window becomes "fuzzy" due to the presence of unobservable events. Here we let $|\cdot|$ be the length of a string and have the following definition.

Definition 1. (Desirable Fuzzy Window). Given system G and maximum fuzzy window size $N \in \mathbb{N}^+$, a string $s = \xi_1 e_1^o \xi_2 e_2^o \cdots \xi_N e_N^o \in \mathcal{L}(G)$ where $\forall i \leq N$ s.t. $e_i^o \in E_o$ and $\xi_i \in E_{uo}^*$ forms a desirable fuzzy window if there exists $1 \leq \ell \leq N$ such that for $t = \xi_1 e_1^o \xi_2 e_2^o \cdots \xi_\ell e_\ell^o \preceq s$, $\frac{\omega(t)}{|t|} \geq 0$.

A fuzzy window is desirable if the local mean payoff is above 0 within at most N observable events. Since the system is partially observed, we only count the number of observable events to evaluate a fuzzy window. That is, we view the number of observable events as the "length" of a fuzzy window whose maximum length is fixed. The string in Definition 1 ends with an observable event by our convention. On the other hand, string $s = \xi_1 e_1^o \xi_2 e_2^o \cdots \xi_N e_N^o$ forms an *undesirable* fuzzy window if $\frac{\omega(s)}{|s|} < 0$.

When the fuzzy window is sliding with new event occurrences, we subsequently check the local mean payoff within each window. Some strings in the open-loop system may not form desirable fuzzy windows, while some others may lead to unsafe states. In those cases, supervisory control is employed and we formulate the supervisory control problem under desirable fuzzy windows as follows.

Problem 1. Given system G with the set of unsafe states X_{us}, maximum fuzzy window size $N \in \mathbb{N}^+$, design a supervisor $S \in \mathbb{S}$ such that: (1) S/G is both safe and live; (2) for all $s \in \mathcal{L}(S/G)$, $i \geq 1$ and $M \geq N$, if $s(i, i+M)$ ends with an observable event and $|P(s(i, i+M))| = N$, then $s(i, i+M)$ forms a desirable fuzzy window.

From the problem statement, safety and liveness are required first; then for every string in the supervised system, whenever a substring ends with an observable event and contains N observable events, it should form a desirable fuzzy window. The major challenge of Problem 1 is the supervisor's partial observation. To this end, we need to properly estimate the current state and track the local mean payoff within the fuzzy windows for the supervisor's decision making. These issues will be resolved in the next section and we end this section with an example.

Example 1. Consider automaton G in Figure 1, with $E_c = \{c_1, c_2\}$, $E_{uc} = \{u_1, u_2, u_3, o_1, o_2, o_3\}$, $E_{uo} = \{c_1, c_2, u_1, u_2, u_3\}$ and $E_o = \{o_1, o_2, o_3\}$. The unsafe state is x_6 and the event weight is shown on each transition. Here we let the maximum fuzzy window size be $N = 3$. Clearly, string $s = c_1 o_1 c_2 o_2 u_2 o_3$ does not form a desirable fuzzy window since $\omega(c_1 o_1) < 0$, $\omega(c_1 o_1 c_2 o_2) < 0$ and $\omega(s) < 0$. Later on, a supervisor will be employed to solve Problem 1 on G and this example will be used through the work.

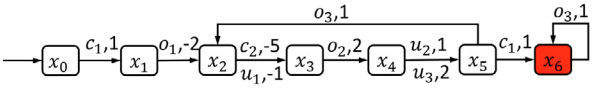

Fig. 1. The system G

4. SOLUTION TO SUPERVISORY CONTROL PROBLEM UNDER FUZZY WINDOWS

Due to the potential challenges of directly solving Problem 1, we transfer it to a two-player game, where the supervisor plays against the antagonistic environment. For this purpose, *windowed information states* are defined which incorporate sufficient information on both the state estimate and the payoffs within windows. Then we introduce a novel information structure called *windowed bipartite transition system* as the game graph and discuss its relevant properties. After some analysis, it turns out that we are tackling a safety game and Problem 1 is solved by locating the supervisor's winning strategies in the game.

4.1 Information-Flow Analysis

Before introducing windowed information states, we give a generic definition of *windowed belief functions*.

Definition 2. (Windowed Belief Functions). Given G and maximum window size N, a N-dimensional windowed belief function is defined as $h : X \to (\mathbb{Z} \cup \{\perp\})^{N+1}$, where the symbol $\perp$ means "value unspecified".

Let $h^{(i)}(x)$ stand for the i-th element in $h(x)$ for some $0 \leq i \leq N$. We just leave a generic definition here and the formula of specifying values is discussed in detail later.

Definition 3. (Windowed Information States). Consider G and maximum fuzzy window size N, a windowed information state is $q^w = ((x_1, \cdots, x_m), [h(x_1), \cdots, h(x_m)]) \in 2^X \times (\mathbb{Z} \cup \{\perp\})^{(N+1)\times m}$ for some $m \in \mathbb{N}^+$. Denote by $\mathcal{E}(q^w)$ and $\mathcal{M}(q^w)$ the state estimate and the matrix in q^w, respectively, thus we write $q^w = (\mathcal{E}(q^w), \mathcal{M}(q^w))$.

We denote by Q^W the set of windowed information states. By definition, $\mathcal{E}(q^w)$ is the state estimate of G and may contain m states where m is a positive integer. The matrix $\mathcal{M}(q^w)$ contains m vectors (windowed belief functions), tracking the accumulative payoff within the last N observable events. Specifically, we denote by $\mathcal{M}(q^w, x)$ the vector in $\mathcal{M}(q^w)$ that is associated with $x \in \mathcal{E}(q^w)$, i.e., $\mathcal{M}(q^w, x) = h(x)$. In other words, we say a windowed information state *induces* a series of windowed belief functions.

We call $q^{aw} \in Q^W \times \Gamma$ an *augmented windowed information state*, i.e., a windowed information state augmented with a control action. Let $I_W(q^{aw})$ and $\Gamma(q^{aw})$ denote the windowed information state and the control decision in q^{aw}, respectively. q^{aw} is *safe* if $\mathcal{E}(I_W(q^{aw})) \cap X_{us} = \emptyset$, i.e., no unsafe state of G is included in the estimate of q^{aw}.

For $q^w = (\mathcal{E}(q^w), \mathcal{M}(q^w)) \in Q^W$ and $x \in \mathcal{E}(q^w)$, we use notations $h_{q^w}(x)$ and $h_{q^w}^{(i)}(x)$ to represent the windowed belief function associated with x and the i-th element in $h_{q^w}(x)$, respectively. Similarly for $q^{aw} = (I_W(q^{as}), \Gamma(q^{as})) \in Q^W \times \Gamma$ and $x \in \mathcal{E}(I_W(q^{as}))$, we write the windowed belief function associated with x and the i-th element of the function as $h_{q^{aw}}(x)$ and $h_{q^{aw}}^{(i)}(x)$, respectively. Windowed information states are updated with control decisions and event occurrences, thus the *information flow* is formed. Then we have the following two concepts.

Definition 4. (γ-successor). For $\gamma \in \Gamma$, $q^{aw} = (q'^w, \gamma) \in Q^W \times \Gamma$ is a γ-successor of $q^w \in Q^W$ if (i) $\mathcal{E}(q'^w) = UR_\gamma(\mathcal{E}(q^w))$; (ii) $\forall x' \in \mathcal{E}(q'^w)$, $\forall 0 \leq i \leq N$, we have:

$$
\begin{aligned}
h^{(i)}_{q^{aw}}(x') &= \min_{x,\xi}\{\omega(\xi) : \exists \xi \in (E_{uo}\cap\gamma)^*, x \in \mathcal{E}(q^w),\ \text{s.t.}\\
&\qquad f(x,\xi) = x'\} \text{ if } i = 0\\
h^{(i)}_{q^{aw}}(x') &= \min_{x,\xi}\{h^{(i)}_{q^w}(x) + \omega(\xi) : \exists \xi \in (E_{uo}\cap\gamma)^*, x \in \mathcal{E}(q^w),\\
&\qquad \text{s.t. } f(x,\xi) = x'\} \text{ if } i \geq 1, h^{(i)}_{q^w}(x) \neq 0, h^{(i)}_{q^w}(x) \neq \perp\\
h^{(i)}_{q^{aw}}(x') &= 0 \text{ if } i \geq 1, \forall x \in \mathcal{E}(q^w), \forall \xi \in (E_{uo}\cap\gamma)^*,\\
&\qquad [f(x,\xi) = x'] \Rightarrow [h^{(i)}_{q^{aw}}(x) = 0]\\
h^{(i)}_{q^{aw}}(x') &= \perp \text{ if } i \geq 1, \forall x \in \mathcal{E}(q^w), \forall \xi \in (E_{uo}\cap\gamma)^*,\\
&\qquad [f(x,\xi) = x'] \Rightarrow [h^{(i)}_{q^w}(x) = \perp] \qquad (1)
\end{aligned}
$$

The γ-successor characterizes how the information flow proceeds with control decisions. First the state estimate is updated to the unobservable reach under γ. Then for every state in the estimate, we update the accumulative weight within the fuzzy window, under the enabled unobserable events in γ. Specifically, there are four cases involved in Definition 4, depending on the index i and the values of the windowed belief functions of the predecessor state.

In general, we use the windowed belief functions as a "counter" to track the minimum accumulative weight incurred by the unobservable reach under γ. Since we only evaluate the mean payoff within fuzzy windows after an observable event occurs, the index i remains the same on both sides of the equations in Definition 4. The index indicates the number of observable events that have occurred and $i = 0$ means that the first observable event has not be counted. The third equation implies that the accumulative payoff has turned nonnegative before or upon the last observable event. Here we briefly explain the role of symbol $\perp$. If only $1 \leq m < N$ observable events have occurred currently, we simply set $h^{(j)}(x) = \perp$ for $j \in \{m+1, \cdots, N\}$.

Definition 5. (e_o-successor). For $e_o \in E_o$, $q^w \in Q^W$ is an e_o-successor of $q^{aw} = (q'^w, \gamma)$ if (i) $e_o \in \gamma$, $\mathcal{E}(q^w) = Next_{e_o}(\mathcal{E}(q'^w))$; (ii) $\forall x \in \mathcal{E}(q^w)$, $\forall 0 \leq i \leq N$, we have:

$$
\begin{aligned}
&h^{(i)}_{q^w}(x) = 0 \text{ if } i = 0\\
&h^{(i)}_{q^w}(x) = \min\{0, \min_{\xi}\{\omega(\xi) + \omega(e_o) : \exists x' \in \mathcal{E}(q'^w),\\
&\xi \in (E_{uo}\cap\gamma)^*,\ \text{s.t. } f(x',\xi e_o) = x\}\} \text{ if } i = 1\\
&h^{(i)}_{q^w}(x) = \min\{0, \min_{x'}\{h^{(i-1)}_{q^{aw}}(x') + \omega(e_o) : \exists x' \in \mathcal{E}(q'^w),\\
&\text{s.t. } f(x', e_o) = x\}\} \text{ if } i \geq 2, h^{(i-1)}_{q^w}(x) \neq 0, h^{(i-1)}_{q^w}(x) \neq \perp\\
&h^{(i)}_{q^w}(x) = 0 \text{ if } i \geq 2, \forall x' \in \mathcal{E}(q'^w), [f(x', e_o) = x] \Rightarrow\\
&[h^{(i-1)}_{q^{aw}}(x') = 0]\\
&h^{(i)}_{q^w}(x) = \perp \text{ if } i \geq 2, \forall x' \in \mathcal{E}(q'^w), [f(x', e_o) = x] \Rightarrow\\
&[h^{(i-1)}_{q^{aw}}(x') = \perp] \qquad (2)
\end{aligned}
$$

After a control decision is made, the enabled observable events (if any) occur and trigger the supervisor to issue another command, which leads to an e_o-successor. Here the state estimate is updated to the observable reach under e_o. We also calculate the values of windowed belief functions associated with each state in the estimate. Similarly with γ-successor, we count the minimum accumulative weight upon the occurrence of e_o. In essence, whenever the accumulative weight becomes nonnegative at certain dimension by some string reaching a state of the estimate, the local mean payoff also becomes nonnegative and a desirable fuzzy window is formed before the window reaches the maximum length N. Then we immediately reset the counter to 0 and start tracking the weight of a new string, as illustrated by the equations in Definition 5. Otherwise, if the current weight sum is still negative and the fuzzy window has not reached length N, we keep tracking the minimum accumulative weight incurred by e_o.

Intuitively speaking, this is because by Definition 1, when the local mean payoff within a fuzzy window becomes nonnegative before the window reaches its maximum length N, then we do not need to further consider the mean payoff within "longer windows" before N. Notice that the index i increases by one since a new observable event has occurred.

Definition 6. (Control-observation sequence). We define a control-observation sequence as a sequence of alternating states, events and control decisions of the form: $\rho = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \cdots \xrightarrow{\gamma_n} q^{aw}_n \xrightarrow{e^o_n} q^w_{n+1}$ or $\rho' = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \xrightarrow{\gamma_2} q^{aw}_2 \cdots \xrightarrow{e^o_n} q^w_{n+1} \xrightarrow{\gamma_{n+1}} q^{aw}_{n+1}$ where $q^w_i \in Q^W$, $q^{aw}_i \in Q^W \times \Gamma$, $\gamma_i \in \Gamma$, $e^o_i \in E_o$, q^{aw}_i is a γ_i-successor of q^w_i and q^w_{i+1} is an e^o_i-successor of q^{aw}_i for all $1 \leq i \leq n$.

Given ρ or ρ' in Definition 6, we write $\rho_k = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \cdots \xrightarrow{\gamma_{k-1}} q^{aw}_{k-1} \xrightarrow{e^o_{k-1}} q^w_k$ and $\rho'_k = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \xrightarrow{\gamma_2} q^{aw}_2 \cdots \xrightarrow{e^o_{k-1}} q^w_k \xrightarrow{\gamma_k} q^{aw}_k$, for $1 \leq k \leq n$. Then the set of strings *generated* by ρ or ρ' is defined recursively as:

$$
\begin{aligned}
Str(\rho_1) =& \{\epsilon\}\\
Str(\rho'_1) =& \{\xi_1 \in E^*_{uo} : \exists x \in \mathcal{E}(q^w_1), x' \in \mathcal{E}(I_W(q^{aw}_1)),\\
&\xi_1 \in (\gamma_1 \cap E_{uo})^* \text{ s.t. } f(x,\xi_1) = x'\}\\
Str(\rho_{k+1}) =& \{s'_k e^o_k : \exists x \in \mathcal{E}(q^w_1), x' \in \mathcal{E}(I_W(q^{aw}_k)), x'' \in\\
&\mathcal{E}(q^w_{k+1}), s'_k \in Str(\rho'_k),\ \text{s.t. } f(x,s'_k) = x',\\
&f(x', e^o_k) = x''\}\\
Str(\rho'_{k+1}) =& \{s_{k+1}\xi_{k+1} : \exists x \in \mathcal{E}(q^w_1), x' \in \mathcal{E}(q^{aw}_{k+1}), x'' \in\\
&\mathcal{E}(I_W(q^{aw}_{k+1})), s_{k+1} \in Str(\rho_{k+1}), \xi_{k+1} \in\\
&(\gamma_{k+1} \cap E_{uo})^*,\ \text{s.t. } f(x, s_{k+1}) = x',\\
&f(x', \xi_{k+1}) = x''\}
\end{aligned}
$$

Remark 1. Consider a control-observation sequence $\rho = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \xrightarrow{\gamma_2} q^{aw}_2 \cdots \xrightarrow{\gamma_n} q^{aw}_n \xrightarrow{e^o_n} q^w_{n+1}$, we say a fuzzy window of observed length ℓ ($\ell \leq n$ and $\ell \leq N$) is *bad* at q^w_{n+1} if $\exists s \in Str(q^w_{n-\ell+1} \xrightarrow{\gamma_{n-\ell+1}} q^{aw}_{n-\ell+1} \xrightarrow{e^o_{n-\ell+1}} q^w_{n-\ell} \cdots \xrightarrow{\gamma_n} q^{aw}_n \xrightarrow{e^o_n} q^w_{n+1})$ such that $\omega(s) < 0$. Otherwise, we say it is *good* at q^w_{n+1} if $\forall s \in Str(q^w_{n-\ell+1} \xrightarrow{\gamma_{n-\ell+1}} q^{aw}_{n-\ell+1} \xrightarrow{e^o_{n-\ell+1}} q^w_{n-\ell} \cdots \xrightarrow{\gamma_n} q^{aw}_n \xrightarrow{e^o_n} q^w_{n+1})$, $\omega(s) \geq 0$.

Theorem 1. Given a control-observation sequence $\rho = q^w_1 \xrightarrow{\gamma_1} q^{aw}_1 \xrightarrow{e^o_1} q^w_2 \xrightarrow{\gamma_2} q^{aw}_2 \cdots \xrightarrow{\gamma_n} q^{aw}_n \xrightarrow{e^o_n} q^w_{n+1}$, a fuzzy window of observed length ℓ where $\ell \leq n$ and $\ell \leq N$ is bad at q^w_{n+1} if $\exists x_{n+1} \in \mathcal{E}(q^w_{n+1})$ such that $h^{(\ell)}_{q^w_{n+1}}(x_{n+1}) < 0$.

We may prove Theorem 1 by induction, while the proof is omitted here due to space limitations. Theorem 1 implies that given ρ, a fuzzy window of observed length N ($N \leq n$) is bad at q^w_{n+1} if $\exists x \in \mathcal{E}(q^w_{n+1})$ such that $h^{(N)}_{q^w_{n+1}}(x) < 0$,

which further indicates that an undesirable fuzzy window is formed. As there is no unobservable loop in G, the values of windowed belief functions in ρ remain bounded.

4.2 Synthesize Supervisors

We proceed to introduce the two-player game structure called *windowed bipartite transition system*, which essentially captures the information flow under control.

Definition 7. (Windowed Bipartite Transition System). A WBTS with respect to G and maximum window size N is a tuple $T = (Q_Y, Q_Z, E, \Gamma, f_{yz}, f_{zy}, y_0)$ where $Q_Y \subseteq Q^W$ is the set of windowed information states; $Q_Z \subseteq Q^W \times \Gamma$ is the set of augmented windowed information states; E is the set of events; Γ is the set of control decisions; $f_{yz} : Q_Y \times \Gamma \to Q_Z$ is the transition function from Q_Y to Q_Z where for $y \in Q_Y$, $\gamma \in \Gamma$ and $z \in Q_Z$, we have $[f_{yz}(y,\gamma) = z] \Rightarrow [z \text{ is a } \gamma \text{ successor of } y]$; $f_{zy} : Q_Y \times E_o \to Q_Z$ is the transition function from Q_Z to Q_Y where for $z = (y,\gamma) \in Q_Z$, $e_o \in E_o$ and $y' \in Q_Y$, we have $[f_{zy}(z, e_o) = y'] \Leftrightarrow [y' \text{ is an } e_o \text{ successor of } z]$; $y_0 \in Q_Y$ is the initial state and we set $y_0 = \{x_0, (\bot, \cdots, \bot)\}$.

The supervisor makes control decisions at Q_Y-states (Y-states) and the game moves to Q_Z-states (Z-states) where the environment lets the enabled observable events occur. They take turns to play. Accordingly, f_{yz} and f_{zy} are defined following the definition of γ-successor and e_o-successor, respectively. Definition 7 is consistent with the mechanism of supervisory control under partial observation where the supervisor's decisions get updated after the occurrence of observable events. The initial state is set to be $\{x_0, (\bot, \cdots, \bot)\}$, which means no fuzzy window is formed before the first observable event occurs.

In a WBTS T, the supervisor should form desirable fuzzy windows and avoid such states by Theorem 1:

$$\mathcal{U}(T) = \{y \in Q_Y : \exists x \in \mathcal{E}(y) \text{ s.t. } h_y^{(N)}(x) < 0\}$$

We call a state in $\mathcal{U}(T)$ as an *undesired* state, which indicates that a undesirable fuzzy window is formed by certain string. Since the supervisor should only reach safe Z-states and $Q_Y \setminus \mathcal{U}(T)$ in T, we have a *safety* game.

Given a WBTS T, for a Y-state y, we define $C_T(y) = \{\gamma \in \Gamma : f_{yz}(y,\gamma)!\}$ as the set of control decisions defined at y. Also a Y-state y is called *complete* if $C_T(y) \neq \emptyset$ and a Z-state z is called *complete* if $\forall e_o \in E_o$, $f_{zy}(z, e_o)! \Leftrightarrow [Next_{e_o}(\mathcal{E}(I_W(z))) \neq \emptyset] \wedge [e_o \in \Gamma(z)]$. That is to say, a Y-state is complete if there is at least one control decision defined and a Z-state is complete if all enabled observable events are defined. Being complete is necessary for supervisor synthesis since the supervisor should always be able to make decisions and no enabled event is blocked. We call T complete if all states in T are complete.

A run in T is a control-observation sequence of the form $r = y_1 \xrightarrow{\gamma_1} z_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{\gamma_n} z_n \xrightarrow{e_n} y_{n+1}$. Here we denote by $Run_y(T)$ (respectively $Run_z(T)$) the set of runs whose last states are Y-states (respectively Z-states).

Both players have *strategies* in the game. Generally, they make decisions based on their history of observation captured by the runs. In a WBTS T, we define the *supervisor's strategy (control strategy)* as $\pi_s : Run_y(T) \to \Gamma$ and the *environment's strategy* as $\pi_e : Run_z(T) \to E_o$. A player selects a transition at its position following its strategy. It turns out that a control strategy works in the same way as a standard supervisor Cassandras and Lafortune [2008]. In the following discussion, we use the terms "supervisor" and "supervisor's strategy (control strategy)" interchangeably. Since the supervisor's decisions are updated on the occurrence of observable events executed by the environment, the control strategies are called *observation based*.

If the supervisor plays π_s while the environment plays π_e from the initial state y_0, then a unique initial run, is generated. We also define $Run(\pi_s, y) = \{y \xrightarrow{\gamma_1} z_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{e_{n-1}} y_n : \forall i < n, \gamma_i = \pi_s(y \xrightarrow{\gamma_1} z_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{\gamma_{i-1}} z_{i-1} \xrightarrow{e_{i-1}} y_i)\}$ as the set of runs starting from y and *consistent* with the control strategy π_s, i.e., the control decisions in the run are specified by π_s.

A strategy $\pi_i \in \Pi_i$ for player $i \in \{s, e\}$ in a WBTS T is *positional* if the decisions only depend on the current windowed or augmented windowed information state. It is known that positional strategies are sufficient for a player to win safety games, see, e.g. Apt and Grädel [2011], thus we will restrict our attention to positional control strategies in the remainder of the work.

Given a WBTS T, we call it *deterministic* if $\forall y \in Q_Y$, $C_T(y)$ is a singleton, i.e., the control decision at each Y-state is unique, which is denoted by $c_T(y)$. Thus there exists a unique control strategy (supervisor) in T and we denote the supervisor by S_T. Furthermore, S_T may be *realized* by an automaton $G_T = (Q_Y, E, \xi, q_0)$ where Q_Y is the set of Y-states in deterministic T; $\xi : Q_Y \times E \to Q_Y$ is the (partial) transition function defined as: $\forall q \in Q_Y$, $\forall e \in E$, we have (i) $\xi(q, e) = q$ if $e \in c_T(y) \cap E_{uo}$ (ii) $\xi(q, e) = f_{zy}(f_{yz}(y, c_T(y)), e)$ if $e \in c_T(y) \cap E_o$. Then we may compute the language of the supervised system as $\mathcal{L}(S_T/G) = \mathcal{L}(G_T \times G)$ where $\times$ is the product operation between automata, see Cassandras and Lafortune [2008].

Then we compare the "size" of WBTSs in the sense of graph merging. Given $T_1 = (Q_Y^1, Q_Z^1, E, \Gamma, f_{yz}^1, f_{zy}^1, y_0^1)$ and $T_2 = (Q_Y^2, Q_Z^2, E, \Gamma, f_{yz}^2, f_{zy}^2, y_0^2)$, we say T_1 is a *subgame* of T_2, denoted by $T_1 \sqsubseteq T_2$, if $Q_Y^1 \subseteq Q_Y^2$, $Q_Z^1 \subseteq Q_Z^2$ and for all $y \in Q_Y^1$, $z \in Q_Z^1$, $\gamma \in \Gamma$, $e \in E$, we have $f_{yz}^1(y,\gamma) = z \Rightarrow f_{yz}^2(y,\gamma) = z$ and $f_{zy}^1(z,e) = y \Rightarrow f_{zy}^2(z,e) = y$. Given a WBTS T and a set of states $Q \subseteq Q_Y \cup Q_Z$, we denote by $T' = T \lfloor Q$ if $T' \sqsubseteq T$ and Q is the state space of T', i.e., the game on T is restricted to a subgame with states in Q. Intuitively we may imagine that all states in $Q_Y \cup Q_Z \setminus Q$ are deleted from T and the game is played on the remaining structure.

Now we conjecture that given system G and maximum length N of fuzzy windows, if we construct a complete WBTS T that is "as large as possible", only contains safe Z-states and has $\mathcal{U}(T) = \emptyset$, then the control strategies in the resulting structure should solve Problem 1. Formally, we denote it by $T_m = (Q_Y^m, Q_Z^m, E, \Gamma, f_{yz}^m, f_{zy}^m, y_0)$ where (i) it is complete; (ii) $\forall z \in Q_Z^m$: z is safe; (iii) $\mathcal{U}(T_m) = \emptyset$; (iv) for all WBTSs T satisfying (i), (ii) and $\mathcal{U}(T) = \emptyset$, $T \sqsubseteq T_m$ holds. Algorithm 1 is presented to build T_m.

Algorithm 1 follows Definition 1 to constructs T_m in two steps. First, $DoDFS$ performs a depth-first search from

Algorithm 1 Build T_m

Input: G, N
Output: $T_m = (Q_Y^m, Q_Z^m, E_o, \Gamma, f_{yz}^m, f_{zy}^m, y_0)$
1: $Q_Y^m = \{y_0\} = \{x_0, \mathbf{0}\}$, $Q_Z^m = \emptyset$;
2: $T_m^{pre} = DoDFS(y_0, G)$;
3: **while** there exist Y-states or Z-states that have no successors **do**
4: **if** $\exists y \in Q_Y^m$ that has no successors **then**
5: remove y and all its predecessor Z-states;
6: **if** $\exists z \in Q_Z^m$ that has no successors **then**
7: remove z;
8: **return** T_m;
9: **procedure** $DoDFS(y, G)$
10: **for** $\gamma \in \Gamma$ **do**
11: $z = f_{yz}(y, \gamma)$ by Definition 7;
12: **if** z is safe **then**
13: add transition $y \xrightarrow{\gamma} z$ to f_{yz}^m;
14: **if** $z \notin Q_Z^m$ **then**
15: $Q_Z^m = Q_Z^m \cup \{z\}$;
16: **for** $e_o \in \gamma \cap E_o$ **do**
17: $y' = f_{zy}(z, e)$ by Definition 7;
18: **if** $\nexists x \in \mathcal{E}(y')$ s.t. $h_{y'}^{(N)}(x) < 0$ **then**
19: add transition $z \xrightarrow{e_o} y'$ to f_{zy}^m;
20: **if** $y' \notin Q_Y^m$ **then**
21: $Q_Y^m = Q_Y^m \cup \{y'\}$;
22: $DoDFS(y', G)$;

the initial state y_0 to build a WBTS that enumerates all potential states and control decisions. Line 12 checks whether a new Z-state (augmented windowed information state) is safe and desirable. If so, we continue to add all successor Y-states and check whether they are desirable in line 18. Then we make a recursive call of $DoDFS$ in line 22, which allows us to traverse the whole state space. The resulting structure is denoted by T_m^{pre} and it is a WBTS.

In T_m^{pre}, there may be Y-states and Z-states without successors, which requires an iterative pruning at line 3. Notice that when a Y-state is removed, all its predecessor Z-states are removed as well, since enabled observable events should not be blocked from occurring. This process may be interpreted as calculating the supremal controllable sublanguage in supervisory control theory if we view states without successors as undesired, transitions for f_{yz} as controllable and transitions for f_{zy} as uncontrollable.

Theorem 2. A control strategy in T_m solves Problem 1.

Proof. Consider a control strategy π_s in T_m. First it is safe following Algorithm 1 since every Z-state in T_m is safe. It is also live as every state in T_m has successors so the supervisor is always able to make decisions perpetual and every enabled event occurs. Then consider a run $r \in Run(\pi_s, y_0)$ consistent with π_s. For any Y-state y in r, we know $\nexists x \in \mathcal{E}(y)$ s.t. $h_y^{(N)}(x) < 0$ since $\mathcal{U}(T_m) = \emptyset$. By negation of Theorem 1, the supervisor only forms desirable fuzzy windows along r and it is the case for any $r \in Run(\pi_s, y_0)$. Therefore, π_s solves Problem 1. □

Thus, we may select a supervisor by our preference. For example, we choose γ_y^* at every $y \in Q_Y^m$ in T_m where $\nexists \gamma_y \in C_{T_m}(y)$ such that $\gamma_y^* \subset \gamma_y$. This leads to a (locally) *maximal permissive* supervisor in terms of enabling events.

Remark 2. Denote by $N_u = \max\{|\xi| : \xi \in E_{uo}^*, \exists x \in X \text{ s.t. } f(x, \xi)!\}$ the maximum length of unobservable string between two observable events in G. It is finite as there is no unobservable loop in G. The state space of T_m is bounded by $[W \cdot (N + N_u)]^{2^{|X|} \cdot (N+1)}$ due to constructing state estimates and calculating windowed belief functions.

Example 2. We continue Example 1 and solve Problem 1. First we follow Algorithm 1 to build T_m. After $DoDFS$, the resulting T_m^{pre} is shown in Figure 2. For simplicity, we only draw state estimates and control decisions in Figure 2, where square states represent Y-states and oval states represent Z-states. The construction is initiated from y_0, where the supervisor may choose to issue γ_0 or γ_0'. However, if event c_1 is disabled, then Z-state z_0' does not have successors. If the supervisor issues γ_0, then we reach y_1 after observable event o_1 occurs. The supervisor and the environment take turns to play until T_m^{pre} is built.

The state estimates of the four red Z-states z_3', z_4', z_7', z_8' contain the unsafe state x_6. Thus they are not included in T_m^{pre}. Next we calculate the values of windowed belief functions. For example, at y_8, we have $h_{y_8}^{(0)}(x_2) = 0$, $h_{y_8}^{(1)}(x_2) = 0$, $h_{y_8}^{(2)}(x_2) = -1$, $h_{y_8}^{(3)}(x_2) = 0$. Since z_9 is a γ_1-successor of y_8, we have $h_{z_9}^{(0)}(x_2) = 0$, $h_{z_9}^{(1)}(x_2) = 0$, $h_{z_9}^{(2)}(x_2) = -1$, $h_{z_9}^{(3)}(x_2) = 0$ and $h_{z_9}^{(0)}(x_3) = \min\{\omega(u_1), \omega(c_2)\} = -5$, $h_{z_9}^{(1)}(x_3) = 0$, $h_{z_9}^{(2)}(x_3) = h_{z_9}^{(2)}(x_2) + \omega(u_2) = -6$, $h_{z_9}^{(3)}(x_3) = 0$. Furthermore, since y_9 is an o_2-successor of z_9, we have $h_{y_9}^{(0)}(x_4) = 0$, $h_{y_9}^{(1)}(x_4) = h_{z_9}^{(0)}(x_3) + \omega(o_2) = -3$, $h_{y_9}^{(2)}(x_4) = 0$, $h_{y_9}^{(3)}(x_4) = h_{z_9}^{(2)}(x_3) + \omega(o_2) = -4$. It turns out that $h_{y_9}^{(3)}(x_4) < 0$ and $h_{y_4}^{(3)}(x_2) < 0$, so y_9 and y^4 are not included in T_m^{pre}. The calculation of windowed belief functions for other states is similar and omitted here.

Then the While loop in Algorithm 1 recursively prunes away states, resulting in T_m in Figure 4. Finally we choose γ_1 ($\gamma_2 \subset \gamma_1$) at y_5 and the resulting supervisor S is in Figure 4. S has "memory" as it alternatively enables and disables c_2 at x_2. We may verify that S solves Problem 1.

5. CONCLUSION

This work discussed local mean payoff supervisory control under partial observation for the first time in DES. The mean payoff within the fuzzy windows should always stay nonnegative. By introducing windowed information states, we transformed the partial observation supervisory control problem to a two-player game and defined the windowed bipartite transition system. Further analysis revealed that it is a safety game. We presented how to synthesize supervisors from the game and showed they are sound solutions to the original problem. For future extension, it is interesting to study whether our method is complete and explore efficient implementation of supervisor synthesis.

REFERENCES

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

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

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to discrete event systems – 2nd Edition*. Springer.

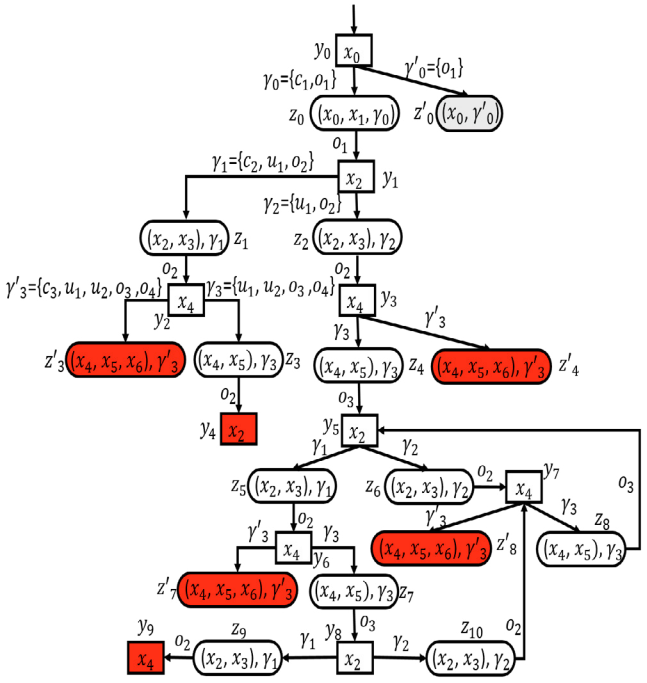

Fig. 2. T_m^{pre} after DoDFS (without the red states)

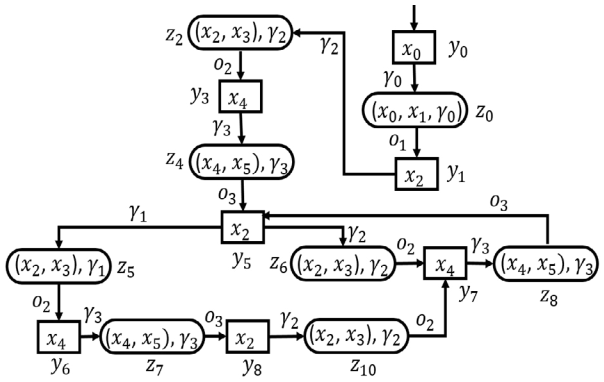

Fig. 3. T_m in Example 2

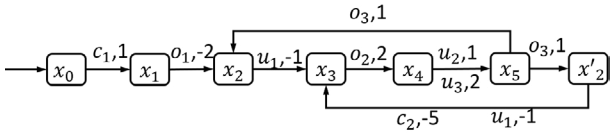

Fig. 4. A synthesized supervisor S that solves Problem 1

Hunter, P., Pérez, G.A., and Raskin, J.F. (2018). Looking at mean payoff through foggy windows. *Acta Informatica*, 55(8), 627–647.

Ji, Y., Yin, X., and Lafortune, S. (2018). Mean payoff supervisory control under partial observation. In *57th IEEE Conference on Decision and Control*, 3981–3987.

Ji, Y., Yin, X., and Lafortune, S. (2019a). Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, 108, 108476.

Ji, Y., Yin, X., and Lafortune, S. (2019b). Supervisory control under local mean payoff constraints. In *58th IEEE Conference on Decision and Control*, 1043–1049.

Komenda, J. and Masopust, T. (2017). Computation of controllable and coobservable sublanguages in decentralized supervisory control via communication. *Disc. Event Dyn. Systems: Theory and App.*, 27(4), 585–608.

Li, J. and Takai, S. (2019). Maximally permissive similarity enforcing supervisors for nondeterministic discrete event systems under partial observation. In *58th IEEE Conference on Decision and Control*, 1037–1042.

Lin, L., Thuijsman, S., Zhu, Y., Ware, S., Su, R., and Reniers, M. (2019). Synthesis of supremal successful normal actuator attackers on normal supervisors. In *American Control Conference*, 5614–5619.

Ma, Z., He, Z., Li, Z., and Giua, A. (2018). Design of monitor-based supervisors in labelled Petri nets. In *14th Intl. Workshop on Discrete Event Systems*, 374–380.

Meira-Góes, R., Kang, E., Kwong, R., and Lafortune, S. (2017). Stealthy deception attacks for cyber-physical systems. In *56th IEEE Conference on Decision and Control*, 4224–4230.

Mohajerani, S., Malik, R., and Fabian, M. (2017). Compositional synthesis of supervisors in the form of state machines and state maps. *Automatica*, 76, 277–281.

Pruekprasert, S. and 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., and Kanazawa, T. (2016). Quantitative supervisory control game for discrete event systems. *IEEE Transactions on Automatic Control*, 61(10), 2987–3000.

Rashidinejad, A., Reniers, M., and Feng, L. (2018). Supervisory control of timed discrete-event systems subject to communication delays and non-FIFO observations. In *14th Intl. Workshop on Discrete Event Syst.*, 456–463.

Ricker, S.L., Lidbetter, T., and Marchand, H. (2017). Inferencing and beyond: further adventures with parity-based architectures for decentralized discrete-event systems. In *20th IFAC World Congress*, 13447–13452.

Shu, S. and Lin, F. (2015). Supervisor synthesis for networked discrete event systems with communication delays. *IEEE Trans. on Auto. Contr.*, 60(8), 2183–2188.

Wang, Y. and Pajic, M. (2019). Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In *58th IEEE Conference on Decision and Control*, 5350–5355.

Wonham, W.M. and Cai, K. (2019). *Supervisory control of discrete-event systems*. Springer.

Wu, B., Zhang, X., and Lin, H. (2019). Permissive supervisor synthesis for Markov decision processes through learning. *IEEE Transactions on Automatic Control*, 64(8), 3332 – 3338.

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

Yin, X. and Lafortune, S. (2017). Synthesis of maximally-permissive supervisors for the range control problem. *IEEE Trans. on Automatic Control*, 62(8), 3914–3929.

Yin, X. (2017). Supervisor synthesis for Mealy automata with output functions: A model transformation approach. *IEEE Trans. on Auto. Cont.*, 62(5), 2576–2581.