Optimal Stabilization of Discrete Event Systems with Guaranteed Worst Cost [⋆]

Yiding Ji [*] **Xiang Yin** [**]

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

Abstract:
This work investigates optimal stabilization with guaranteed worst-case performance of stochastic discrete event systems by supervisory control. We formulate the problem on probabilistic weighted automata. The system is driven to a specified set of target states after a finite number of transitions, thus stabilized. The cost of stabilization is concerned with the accumulative weight of transitions reaching target states. Our goal is to optimize the expected cost of reaching target states, while ensuring that the worst-case individual cost is bounded by a given threshold. Then we transform the supervisory control problem to a two-player stochastic game between the supervisor and the environment, which properly encodes the worst-case requirement. Finally an algorithm is presented to synthesize the optimal supervisor by leveraging results from Markov Decision Processes, which turns out to provably solve the original problem.

Keywords: Discrete event systems, optimal supervisory control, stabilization, stochastic game

1. INTRODUCTION

Supervisory control in discrete event systems (DES) has been well investigated since it was proposed. In this framework, a supervisor controls the plant by enabling and disabling events dynamically, so that the plant fulfills a given specification; see, e.g., Cassandras and Lafortune [2008], Wonham and Cai [2018].

In practice, the supervisor may not have perfect observation of the system, which brings in uncertainty and gives rise to supervisory control under partial observation. There is a rich literature on this topic, see, e.g., Alves et al. [2019], Shu and Lin [2015], Yin and Lafortune [2016a,b, 2017], Komenda and Masopust [2017], Mohajerani et al. [2017], Ji et al. [2018], Ma et al. [2019, 2018], Rashidinejad et al. [2018], Lin et al. [2019] for some recent discussions.

The uncertainties in system modeling also lead to supervisory control of *stochastic DES*, where events are associated with probabilities of occurrence. Both deterministic and probabilistic supervisory control frameworks were discussed, depending on whether events are enabled for sure or with certain probabilities, see, e.g., Kumar and Garg [2001], Pantelic et al. [2009]. More recently, learning-based supervisor synthesis was considered in Wu et al. [2019] and Meira-Góes et al. [2019] discussed deception attacks for supervisors in probabilistic settings.

In many applications, some states of the system called *target states* are of particular interest as they indicate the fulfillment of some important tasks. Meanwhile, one may apply the supervisory control theory to drive the system to those target states within a bounded number of transitions and stay there afterwards. This problem is referred to as *stabilization* or *state attraction*, which was originally proposed in Brave and Heymann [1990] and generalized by Schmidt and Breindl [2014]. A matrix-based approach of optimal stabilization for nondeterministic DES was provided in Han et al. [2019] recently. Additionally, *optimal stabilization problem* considers minimizing the cost of driving the system to target states. This topic was initialed by Brave and Heymann [1993] for the full observation case and extended in Marchand et al. [2002], Pruekprasert and Ushio [2016b] for the partial observation case. As a variant, Pruekprasert and Ushio [2016a] considered optimal stabilization under the influence of disturbances.

All existing works on optimal stabilization focus on optimizing the system's worst possible performance when being stabilized. On the other hand, optimal supervisory control of probabilistic discrete event systems aims to ensure a good *expected* overall performance. Both frameworks suffer from some drawbacks. The worst-case analysis may ignore overall performance, while good expectation may not rule out some extreme individual behaviors that are not very likely but indeed possible. The conventional settings of optimal supervisory control may be *insufficient* to provide formal guarantees under multiple scenarios.

Let us consider an everyday situation when a student commutes between home and campus. There are three possible options: subway, car and on foot. Usually subway is the fastest option with potential risk of severe delay. Self-driving has more expected arriving time, while guarantees to reach campus within a tolerable amount of time under all traffic circumstances. Then it is more reasonable to

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

drive instead of taking the risk of the subway, when the student could not afford to be late for some activity.

Motivated by the above situation, we investigate optimal stabilization under an acceptable upper bound for worst-case cost. We formulate this problem in the context of probabilistic weighted automata, where each event is assigned a cost and a probability of occurrence. Next we propose the *probabilistic weighted bipartite transition system (PWBTS)* and transform the problem to a two-player stochastic game between the supervisor and the environment. Then we construct the largest PWBTS which contains all enforcing the worst cost, reformulated as a safety condition over the new information space. Furthermore, it turns out that the game is an Markov Decision Process (MDP). Finally we synthesize the optimal supervisor from the game, which provably solves the proposed stabilization problem. Our solution procedure leverages results from beyond worst-case synthesis in Bruyere et al. [2017] and optimal control of MDPs in Filar and Vrieze [2012].

The rest of the work is organized as follows. Section 2 introduces the system model. Section 3 formulates the problem of optimal stabilizing supervisory control with guaranteed worst cost. In Section 4, a two-player stochastic game is defined to encode the worst-case requirement. Then in Section 5, the optimal supervisor is synthesized to solve the problem. Finally, Section 6 concludes the paper.

2. SYSTEM MODEL

We consider the weighted finite-state automaton model:

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

where X is the finite state space, E is the finite set of events, $f : X \times E \to X$ is the (partial) deterministic transition function and event e is *active* at a state x if $f(x, e)!$ where ! means "is defined", $x_0 \in X$ is the initial state and $\omega : E \to \mathbb{N}^+$ is the weight function which represents the event costs. The domain of f can be extended to $X \times E^*$ in the standard manner Cassandras and Lafortune [2008] and we still denote the extended function by f. The language generated by G is defined as $\mathcal{L}(G) = \{s \in E^* : f(x_0, s)!\}$. We denote by $s \leq u$ if string s is a prefix of u, and $s < u$ if $s \leq u$ and $s \neq u$. The function ω is additive and its domain can be extended to E^* by letting $\omega(\epsilon) = 0$, $\omega(se) = \omega(s) + \omega(e)$ for all $s \in E^*$ and $e \in E$. Given $s = e_1 e_2 \cdots e_n \in \mathcal{L}(G)$, its *(accumulative) cost* is the sum of each event's weight (cost), i.e., $\omega(s)$.

Given an automaton 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 it may be infinitely long. We denote by $Run(G)$ the set of runs in G. A run is *initial* if its initial state is the initial state of the system. We say a run forms a *cycle* if $x_1 = x_n$ and we call a system *acyclic* if it does not contains cycles, otherwise, it is called *cyclic*.

In system G, $x \in X$ is a *terminating* state if $\nexists e \in E$ s.t. $f(x, e)!$, i.e., no outgoing transition is defined. We let $X_T \subseteq X$ be the set of *target states* where $\forall x \in X_T$, x is terminating. We further denote by $L(X_T, G) = \{t \in \mathcal{L}(G) : f(x_0, t) \in X_T\}$ the set of strings reaching X_T. The target states indicate the completion of some essential tasks and ideally, the system should be attracted to them.

A stochastic discrete event system is modeled by a probabilistic weighted finite-state automaton (G, p) where G is a weighted finite-state automaton and $p : X \times E \to [0, 1]$ is the probability of event occurrence. Specifically, for any $x \in X$, $e \in E$, we denote by $p(e|x)$ the probability that event e occurs from state x. In this work, we assume that: (i) for every non-terminating state x, $\sum_{e \in E} p(e|x) = 1$; (ii) $\forall x \in X$, $e \in E$: $p(e|x) > 0 \Leftrightarrow f(x, e)!$.

For a string $s \in \mathcal{L}(G)$, we introduce $Pr(s)$ as the probability that s occurs, which is defined recursively as: $Pr(\epsilon) = 1$ and $Pr(se) = Pr(s)p(e|f(x_0, s))$ for all $s \in E^*$, $e \in E$. Thus, the occurrence probability of a set of strings L is $Pr(L) = \sum_{s \in L} Pr(s)$ and L is the so called *probabilistic language* in Kumar and Garg [2001].

A standard supervisor is a function $S : \mathcal{L}(G) \to 2^E$ that controls the system to achieve certain specification by dynamically enabling/disabling events. 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, all events are *observable* in this context. We denote by $\mathbb{S}$ the set of supervisors. Notice that a supervisor never disables uncontrollable events and a control decision $\gamma \in 2^E$ is *admissible* if $E_{uc} \subseteq \gamma$. We also say a control decision γ is *non-redundant* at state x if $\forall e \in \gamma$, $f(x, e)!$ in G, i.e., all enabled events are defined at x. Denote by Γ the set of all admissible and non-redundant control decisions, then we only consider Γ in this work. We also use S/G to represent the supervised system under S, where $\mathcal{L}(S/G)$ and $Run(S/G)$ stand for the language and the set of runs in S/G, respectively.

In this work, the supervisors are *deterministic* in the sense that control commands are issued *for sure*. Here we make the same assumption on the supervisor as in Kumar and Garg [2001] for the sake of following discussion.

Assumption 1. Given system (G, p) and supervisor S, after S issues $\gamma \neq \emptyset$ at $x \in X$, the occurrence probability of event $e \in \gamma$ becomes $\frac{p(e|x)}{\sum_{e' \in \gamma} p(e'|x)}$ in the supervised system.

In other words, after some events are disabled at a state, the probability of each enabled event increases proportionally so that the probabilities of enabled events at non-terminating states still sum up to 1. Additionally, if the supervisor disables all events at a state, then that state will become a terminating state. From the results in Kumar and Garg [2001], the language in the supervised system is still a probabilistic language under this assumption.

A finite discrete Markov decision process (MDP) is a tuple:

$$M = (X_M, A, \Delta, \omega_m)$$

Here X_M is the finite state space partitioned as $X_M^{ns} \cup X_M^s$ where X_M^{ns} is the set of non-stochastic states and X_M^s is the set of stochastic states. A is the set of actions from X_M^{ns} to X_M^s. $\Delta : X_M^s \times X_M^{ns} \to [0, 1]$ is the probability of the transition between a stochastic state and a non stochastic state. $\omega_m : X_M^s \times X_M^{ns} \to \mathbb{R}$ is the reward or cost function.

An MDP may be seen as a special two-player stochastic game: one player plays from X_M^{ns} against a probabilistic adversary which makes randomized decisions characterized by Δ from X_M^s. Hence, MDPs are sometimes termed "$1\frac{1}{2}$-player games" in the literature, e.g., Bruyere et al. [2017].

3. PROBLEM FORMULATION

Stability is critical for control systems. Intuitively, a system is stable if all its trajectories are driven to certain region after a while and remain there afterwards. Additionally, certain cost may be entailed by those trajectories. For DES, the open-loop system G may not end up in target states or the cost of reaching target states is excessively high, either for a single string or the expectation of multiple strings. In this section, we introduce the stabilizing supervisor and formulate the key problem of this work.

Definition 1. (Stabilizing Supervisor). Given G and target states X_T, a supervisor S is stabilizing if $\forall s \in \mathcal{L}(S/G)$, $\exists t \in E^*$ and $\exists N \in \mathbb{N}^+$ s.t. $|t| \geq N \Rightarrow f(x_0, st) \in X_T$.

A stabilizing supervisor drives every string in the supervised system to target states and it should never disable all events at a non-target state by definition. We call a system as *stabilizable* if there exists a stabilizing supervisor and denote by $L(X_T, S/G) = \{s \in \mathcal{L}(S/G) : f(x_0, s) \in X_T\}$ the set of strings reaching target states under S.

Then we evaluate the expected cost of strings reaching X_T as $\mathbb{E}_S(X_T) = \sum_{s \in L(X_T, S/G)} \omega(s) Pr(s)$, which is also called the *expectation of stabilization* under S. Since all strings in $L(X_T, G)$ are finite and their costs are bounded, the expectation is well defined. Lower $\mathbb{E}_S(X_T)$ is preferred in practice. We also claim that $Pr(L(X_T, S/G)) = 1$ holds for a stabilizing supervisor under Assumption 1, while we omit the argument here due to space limitation.

However, one severe shortage of the expected cost is the failure to rule out certain extreme cases. A stabilizing supervisor with good expectation of stabilization may still permit strings with exceptionally high cost to occur, which is troublesome. Thus we have to ensure an acceptable worst-case performance, i.e., upper bound of individual string cost when optimizing the expectation. That is, our supervisors is *risk aware* of high individual cost. This gives rise to the following problem of *Optimal Stabilization by Supervisory Control with Guaranteed Worst Cost.*

Problem 1. Given probabilistic system (G, p) with target states X_T and cost threshold $\mu \in \mathbb{N}^+$, design a supervisor S^* such that $\mathbb{E}_{S^*}(X_T) = \min_{S \in \mathbb{S}_a} \mathbb{E}_S(X_T)$ where $\mathbb{S}_a = \{S \in \mathbb{S} : S \text{ is stabilizing and } \forall s \in L(X_T, S/G), \omega(s) \leq \mu\}$

Here the condition $\forall s \in L(X_T, S/G), \omega(s) \leq \mu$ is equivalent with the maximum (worst) string cost of reaching X_T being bounded. Thus, we synthesize the optimal supervisor under the constraint of worst-case cost. The stabilizing supervisor should settle for trade-offs between optimal expectation and acceptable worst-case cost. Since event costs are always positive and target states are terminating, the resulting stabilizing supervisor leads to an acyclic system.

Example 1. The system is shown in Figure 1. Here $X_T = \{x_6\}$ so x_6 is the only target state. The set of controllable events is $E_c = \{a, b, c, d, e, f\}$ and the set of uncontrollable events is $E_{uc} = \{u_1, u_2, u_3, u_4, u_5\}$. The event weights and probabilities are labeled with the events. For example, $(a, 1, 0.5)$ at x_0 means $\omega(a) = 1$ and $p(a|x_0) = 0.5$.

We set the worst-case cost threshold as $\mu = 100$. Apparently, some strings do not reach the target state while some others violate the threshold. Our goal is to design an optimal supervisor to solve Problem 1 on the system.

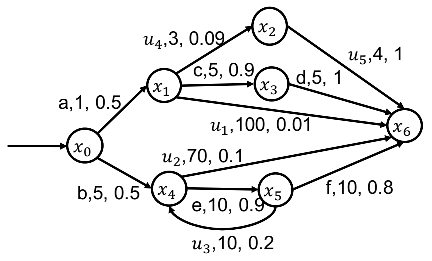

Fig. 1. The probabilistic system in Example 1

4. PROBABILISTIC WEIGHTED BIPARTITE TRANSITION SYSTEM

In this section, we transform Problem 1 to a two-player stochastic game between the supervisor and the environment.For this purpose, a novel information structure called *probabilistic weighted bipartite transition system (PWBTS)* was proposed to characterize the game.

Definition 2. (PWBTS). A PWBTS w.r.t. (G, p) is a tuple

$$T = (Q_Y, Q_Z, E, \Gamma, \omega, f_{yz}, f_{zy}, p_w, y_0)$$

where: $Q_Y \subseteq X \times \mathbb{N}$ is the set of states where the supervisor plays, for $y \in Q_Y$, we write $y = (Sta(y), Lev(y))$; $Q_Z \subseteq X \times \mathbb{N} \times \Gamma$ is the set of states where the environment plays, for $z \in Q_Z$, we write $z = (Sta(z), Lev(z), \Gamma(z))$; E is the set of events;Γ is the set of control decisions; $\omega : E \to \mathbb{N}^+$ is the weight function of G and labels transitions between Q_Z and Q_Y states; $f_{yz} : Q_Y \times \Gamma \to Q_Z$ is the transition function from Q_Y states to Q_Z state where for $y \in Q_Y$, $\gamma \in \Gamma$ and $z \in Q_Z$, we have: $f_{yz}(y, \gamma) = z \Rightarrow [z = (y, \gamma)]$; $f_{zy} : Q_Z \times E \to Q_Y$ is the transition function from Q_Z states to Q_Y states where for $z = (y, \gamma) \in Q_Z$, $e \in E$ and $y' \in Q_Y$, we have: $[f_{zy}(z, e) = y'] \Leftrightarrow [e \in \gamma] \wedge [y' = f(y, e)] \wedge [Lev(y') = Lev(z) + \omega(e)]$; $p_w : Q_Z \times E \to (0, 1]$ is the probability function for transitions where for $z = (y, \gamma) \in Q_Z$, $e \in \gamma$, we have: $f_{zy}(z, e)! \Rightarrow p_w(e|z) = \dfrac{p(e|Sta(z))}{\sum_{e' \in \gamma} p(e'|Sta(z))}$; $y_0 = (x_0, 0) \in Q_Y$ is the initial state.

The game on a PWBTS starts from the initial state y_0 and the two players (supervisor and environment) take turns to play. We refer to a Q_Y-state as a Y-state for simplicity, where the supervisor issues control decisions. A Y-state contains a state $Sta(y)$ from G and a positive integer $Lev(y)$ indicating the cost of a string reaching $Sta(y)$. We also refer to a Q_Z-state as a Z-state where the environment executes the events enabled by the supervisor. A Z-state contains a state $Sta(z)$, a positive integer component $Lev(z)$ and a control decision $\Gamma(z)$. f_{zy} are defined from Y-states to Z-states to reflect the control decisions issued by the supervisor. For $y \in Q^Y$ in a PWBTS T, we let $C_T(y)$ be set of control decisions defined at y. While f_{zy} transitions are defined from Z-states to Y-states to indicate the occurrence of enabled events. We put "$\Leftrightarrow$" in defining f_{zy} due to the fact that the supervisor is unable to choose which enabled event to occur and we should have them all defined at a Z-state. Notice that along with f_{zy}, we recalculate the occurrence probability of each enabled event $e \in \gamma$ following Assumption 1, which is reflected in probability function p_w. Counterparts of the PWBTS for nonstochastic DES are defined in Ji et al. [2019a,b].

Given a PWBTS T, we say it is *complete* if $\forall y \in Q_Y$, $C_T(y) \neq \emptyset$, i.e., every Y-state has a successor. While a Z-

state z is *deadlock-free* if either $\exists e \in E$ such that $f_{zy}(z,e)!$ or $Sta(z) \in X_T$. In other words, a deadlock-free Z-state either has a outgoing transition or it has a terminating states from G. Otherwise, we call z a *deadlocking* state.

In T, a run is of the form $r = y_1 \xrightarrow{\gamma_1} z_1 \xrightarrow{e_1} y_2 \xrightarrow{\gamma_2} z_2 \xrightarrow{e_2} \cdots \xrightarrow{\gamma_n} z_n \xrightarrow{e_n} y_{n+1}$. We let $Last_Y(r)$ and $Last_Z(r)$ be the last Y-state and Z-state of r. Then we denote by $Run_y(T)$ (respectively $Run_z(T)$) the set of initial runs whose last states are Y-states (respectively Z-states). Also, we let $l_g(r)$ be the string *generated* by run r, i.e., $l_g(r) = e_1 \cdots e_n$.

Then we define the *supervisor's strategy (control strategy)* in T as $\pi_s : Run_y(T) \to \Gamma$ and the *environment's strategy* defined as $\pi_e : Run_z(T) \to E_o$. Each player selects transitions according to its strategy. We let Π_s (respectively Π_e) be the set of supervisor's (respectively environment's) strategies. 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 control strategy π_s, i.e., the control decisions in the run are specified by π_s. Specifically, for $s \in \mathcal{L}(G)$, we write $r(\pi_s, y_0, s)$ as the run in $Run(\pi_s, y_0)$ with $l_g(r) = s$, i.e., $r(\pi_s, y_0, s)$ generates s.

Notice that there is no difference between a control strategy and a standard supervisor in their mechanisms. For this reason, we will not distinguish between them from now on. Given a control strategy π_s and string s, we use notations π_s/G and $\pi_s(s)$ to stand for the supervised system under π_s and the control decision of π_s on occurrence of s, respectively. Then π_s is *included* in T if $\forall s \in \mathcal{L}(\pi_s/G)$, $\pi_s(s) \in C_T(Last_Y(r(\pi_s, y_0, s)))$, in other words, the control decisions of π_s on any string are defined in T. Besides, π_s is *memoryless* if $\forall r, r' \in Run_y(T)$ with $Last_Y(r) = Last_Y(r')$, $\pi_s(r) = \pi_s(r')$ holds. That is, a memoryless strategy only depends on the last state of runs.

5. SYNTHESIZE THE OPTIMAL SUPERVISOR

We proceed to solve Problem 1 sequentially based on the PWBTS in this section. First we resolve the worst-case cost requirement, then synthesize the optimal supervisor.

We may compare the "size" of PWBTSs in graph sense. Consider $T_1 = (Q_Y^1, Q_Z^1, E, \Gamma, \omega^1, f_{yz}^1, f_{zy}^1, p_w^1, y_0^1)$ and $T_2 = (Q_Y^2, Q_Z^2, E, \Gamma, \omega^2, f_{yz}^2, f_{zy}^2, p_w^2, y_0^2)$, 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 that $f_{yz}^1(y,\gamma) = z \Rightarrow f_{yz}^2(y,\gamma) = z$ and $f_{zy}^1(z,e) = y \Rightarrow f_{zy}^2(z,e) = y$.

We call a set of states Q in T as a *closed set* to the supervisor if we have: (i) for all $y \in (Q \cap Q_Y)$ and for all $\gamma \in \Gamma$, $f_{yz}(y,\gamma)! \Rightarrow f_{yz}(y,\gamma) \in Q$, i.e., all successors of the supervisor's states in Q are also in Q; (ii) for all $z \in (Q \cap Q_Z)$, there exists $e \in E$ such that $f_{zy}(z,e) \in Q$, i.e., every environment's state in Q has a successor in Q. Intuitively, if the supervisor reaches a closed set, then the environment is able to "trap" its opponent there forever. A closed set to the environment is defined analogously.

Afterwards, we construct the maximum complete PWBTS where (i) all Z-states are deadlock-free and (ii) all the control strategies lead to the target states without exceeding the worst-case cost threshold μ. We denote this structure by $T_m^\mu = (Q_Y^m, Q_Z^m, E, \Gamma, \omega, f_{yz}^m, f_{zy}^m, p_w^m, y_0)$ and formally define it by construction in Algorithm 1. Here being maximum means that for all PWBTS T satisfying the above two conditions, $T \sqsubseteq T_m^\mu$ always holds.

Algorithm 1 Construct T_m^μ with respect to G

```
Input: $G, p, \mu$
Output: $T_m^\mu = (Q_Y^m, Q_Z^m, E, \Gamma, \omega, f_{yz}^m, f_{zy}^m, p_w^m, y_0)$
 1: $Q_Y^m = \{y_0\}$, $Q_Z^m = \emptyset$;
 2: $T_{pre} = DoDFS(y_0, G, p, \mu)$;
 3: while there exist $Y$-states that have no successor do
 4:     Remove all such $Y$-states and their predecessor $Z$-states, take the accessible part;
 5: return $T_m^\mu$;
 6: procedure $DoDFS(y, G, p, \mu)$
 7:     for $\gamma \in \Gamma$ do
 8:         $z = f_{yz}(y, \gamma)$;
 9:         if $z$ is deadlock-free then
10:             add transition $y \xrightarrow{\gamma} z$ to $f_{yz}^m$;
11:             if $z \notin Q_Z^m$ then
12:                 $Q_Z^m = Q_Z^m \cup \{z\}$;
13:                 for $e \in \gamma$ do
14:                     $y' = f_{zy}(z, e)$;
15:                     add $z \xrightarrow{e} y'$ to $f_{zy}^m$, calculate $p_w^m(e|z)$;
16:                     if $y' \notin Q_Y^m$ then
17:                         $Q_Y^m = Q_Y^m \cup \{y'\}$;
18:                         if $Lev(y') \le \mu$ then
19:                             $DoDFS(y', G, p, \mu)$;
20:                         else
21:                             stop searching;
```

Algorithm 1 builds T_m^μ in a depth-first search manner. We start construction from y_0. Then we run $DoDFS$ recursively to add new states and transitions following Definition 2. The costs of strings are integrated into the states and their values are updated. We track the cost until it exceeds μ, then stop construction in line 21. Otherwise, we repeat $DoDFS$ in line 19 until no states are added.

After $DoDFS$, there may be two types of Y-states that have no successors: Y-states whose integer components are above μ or whose only feasible control decision leads to deadlocking Z-states. We prune away such states together with their predecessor states since the supervisor may not choose which enabled event to occur. In other words, this results in a subgame of T_{pre}, whose states have cost levels no greater than μ and are closed to the environment.

Remark 1. Let $|\cdot|$ be the cardinality of a set. By construction, the complexity of T_m^μ is polynomial in the size of system G: the states in T_m^μ have components from X and 2^E. Furthermore, it is also polynomial in the threshold μ as we stop construction when the cost exceeds μ. So it takes pseudo-polynomial time $O(|X| \cdot \mu \cdot 2^{|E|})$ to build T_m^μ.

Example 2. We build T_m^μ with respect to the system in Example 1. After $DoDFS$ in Algorithm 1, part of T_{pre} is in Figure 2. The square states are Y-states and the oval states are Z-states. For simplicity, we only show the transition probabilities in the figure and omit the transition weights.

The game is initialed at Y-state $(x_0, 0)$. Since both events a and b are controllable, the supervisor has four choices: to enable a or b alone, to enable them both or to disable them. Thus, there are four f_{yz} transitions defined at $(x_0, 0)$. Here we show the enabled events under each control decision and use $\{\}$ for enabling no event. Only enabled events that are also active at that state are included in a control decision. Notice that if the supervisor chooses γ_0 from $(x_0, 0)$, then the successor Z-state $(x_0, 0, \{\})$ is a deadlocking state since no event may occur from it and x_0 is not a target state. For this reason, $(x_0, 0, \{\})$ is not added to the structure in $DoDFS$ and we make this state shaded. Similarly, another deadlocking Z-state $(x_3, 6, \{\})$ is also shaded and not included. If the supervisor chooses γ_2, then two f_{zy} transitions are defined at the Z-state $(x_0, 0, \gamma_2)$. Meanwhile, the transition probabilities for events a and b at $(x_0, 0, \gamma_2)$ are $p_w((x_0, 0, \gamma_2), a) = p_w((x_0, 0, \gamma_2), b) = 0.5$. We also update the cost levels when we reach new Y-states $(x_1, 1)$ and $(x_2, 5)$ since $\omega(a) = 1$ and $\omega(b) = 5$. The remaining game graph is interpreted in a similar way.

Notice that $DoDFS$ stops at the two red Y-states $(x_6, 101)$ and $(x_6, 115)$ since $Lev((x_6, 101))$ and $Lev((x_6, 115))$ are above $\mu = 100$. Also we calculate transition probabilities following Definition 2. For example, at Z-state $(x_1, 1, \gamma_4)$, since only u_1 and u_4 are enabled in γ_4, the probability of u_1 becomes $\frac{p(u_1|x_1)}{p(u_1|x_1)+p(u_4|x_1)} = 0.1$ and the probability of u_2 becomes $\frac{p(u_2|x_1)}{p(u_1|x_1)+p(u_4|x_1)} = 0.9$. While at Z-state $(x_1, 1, \gamma_5)$, since u_1, u_4 and c are all enabled, the transition probabilities in Figure 2 are the same as the occurrence probabilities of those events at state x_1 in Figure 1.

Then we prune states from Figure 2 by Algorithm 1. Actually, there should be more states after the dashed state $(x_5, 45)$ by $DoDFS$. However, all such states together with $(x_5, 45)$ become inaccessible after $(x_6, 115)$, $(x_4, 45, \gamma_8)$ and $(x_4, 45, \gamma_9)$ are removed. We do not draw them in Figure 2. Finally, T_m^μ is shown in Figure 3. As is seen, every leaf state in T_m^μ is with the target state x_6 and has cost level less that 100. Hence, each control strategy in T_m^μ is stabilizing and meets the worst-case cost threshold μ. □

Notice that it is possible that Algorithm 1 returns nothing, which indicates that no supervisor satisfies the worst-case cost requirement. If that is the case, there is no need to search for the optimal strategy and Problem 1 has no solution. When T_m^μ is not empty, we justify Algorithm 1.

Lemma 1. A control strategy π_s is stabilizing and $\forall s \in L(X_T, \pi_s/G), \omega(s) \leq \mu$ if and only if π_s is included in T_m^μ.

Lemma 1 implies that any control strategy in T_m^μ stabilizes the system without violating the worst-case cost threshold μ. The proof is omitted here. Also by construction, the leaf stats of T_m^μ are such that $\{z \in Q_Z^m : Sta(z) \in X_T, Lev(z) \leq \mu\}$. We are particularly interested in those states and the supervisor should only reach them on T_m^μ. Since the supervisor plays by making control decisions and the antagonistic environment plays by assigning probability distributions to the enabled events, the game on T_m^μ is actually an Markov Decision Process (MDP). Then our goal is to search for an optimal control strategy on T_m^μ and the dynamic/linear programming method for the *stochastic shortest path problem* suffices. By established results in Filar and Vrieze [2012], the optimal supervisor is memoryless. To proceed, we present Algorithm 2.

The following theorem captures the main result of this work. Since it takes polynomial time in the size of an MDP to search for an optimal strategy Filar and Vrieze [2012], Algorithm 2 requires polynomial time in the size of T_m^μ.

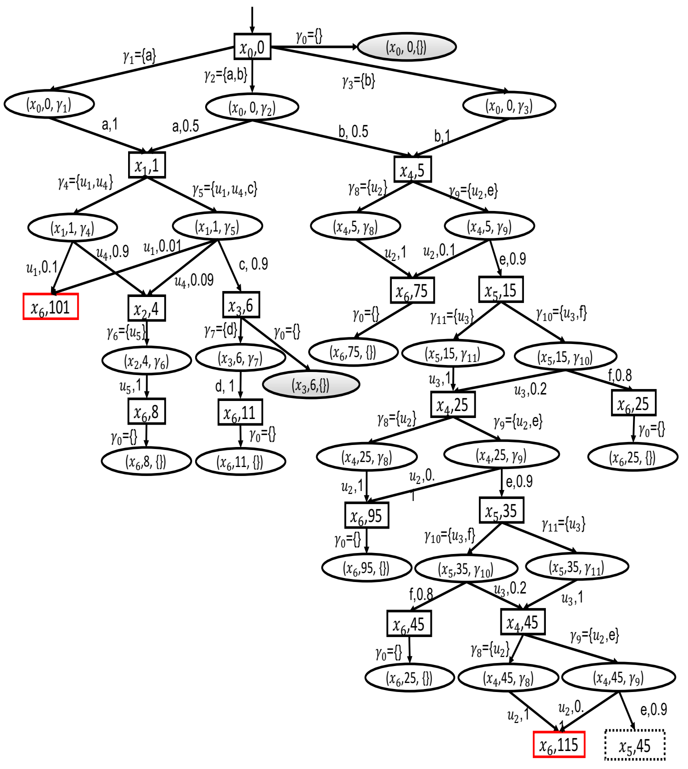

Fig. 2. Part of the T_{pre} (without the shaded states)

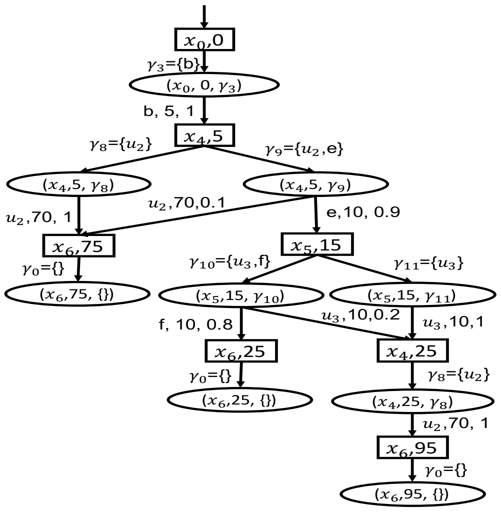

Fig. 3. The T_m^μ in Example 2

Algorithm 2 Find an optimal control strategy

```
Input: $T_m^\mu$
Output: An optimal control strategy $\pi_s^*$
1: Find $Q_T = \{z \in Q_Z^m : Sta(z) \in X_T, Lev(z) \leq \mu\}$;
2: View $Q_T$ as the set of states to reach in the MDP $T_m^\mu$;
3: Apply the method of solving stochastic shortest path
   problem to search for an optimal control strategy $\pi_s^*$;
```

Theorem 2. π_s^* returned by Algorithm 2 solves Problem 1.

Example 3. We continue Example 2 to solve Problem 1 following Algorithm 2. We view T_m^μ as an MDP and synthesize an optimal supervisor, which reaches the leaf states in Figure 3. The expectation of stabilization is 42.6 for the optimal supervisor shown in Figure 4.

It is interesting to notice that if the supervisor disables b but enables a at x_0, then we get an even lower expectation of stabilization, given that the probability of u_1 is fairly low. However, the worst-case cost is greater than 100 since u_1 is uncontrollable. This reveals the trade-off between the expected cost and the worst cost of stabilization.

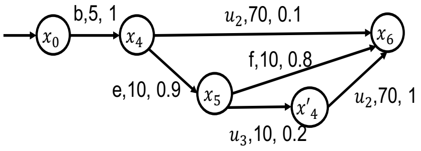

Fig. 4. The optimal supervisor solving Problem 1

6. CONCLUSION

This work was the first to study optimal stabilization on stochastic DES by supervisory control with a guarantee of worst-case performance. To transform the control problem to a two-player stochastic game between the supervisor and the environment, we proposed probabilistic weighted bipartite transition system (PWBTS). Then we constructed the largest PWBTS to encode the worst-case cost. We further illustrated that the game is an Markov Decision Process (MDP). Finally we synthesized the optimal control strategy by solving the MDP. For future work, we would consider more complicated performance measures and the potential application of our developed method.

REFERENCES

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

Brave, Y. and Heymann, M. (1990). Stabilization of discrete-event processes. *International Journal of Control*, 51(5), 1101–1117.

Brave, Y. and Heymann, M. (1993). On optimal attraction in discrete-event processes. *Info. sciences*, 67, 245–245.

Bruyere, V., Filiot, E., Randour, M., and Raskin, J.F. (2017). Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Information and Computation*, 254, 259–295.

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

Filar, J. and Vrieze, K. (2012). *Competitive Markov decision processes.* Springer.

Han, X., Chen, Z., and Su, R. (2019). Synthesis of minimally restrictive optimal stability-enforcing supervisors for nondeterministic discrete event systems. *Systems & Control Letters*, 123, 33–39.

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.

Kumar, R. and Garg, V.K. (2001). Control of stochastic discrete event systems modeled by probabilistic languages. *IEEE Trans. on Auto. Control*, 46(4), 593–606.

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., and Li, Z. (2019). Supervisory control in partially observable petri nets with sensor reduction. In *15th IEEE International Conference on Automation Science and Engineering*, 189–194.

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.

Marchand, H., Boivineau, O., and Lafortune, S. (2002). On optimal control of a class of partially observed discrete event systems. *Automatica*, 38(11), 1935–1943.

Meira-Góes, R., Kwong, R., and Lafortune, S. (2019). Synthesis of sensor deception attacks for systems modeled as probabilistic automata. In *American Control Conference*, 5620–5626.

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.

Pantelic, V., Postma, S.M., and Lawford, M. (2009). Probabilistic supervisory control of probabilistic discrete event systems. *IEEE Transactions on Automatic Control*, 54(8), 2013–2018.

Pruekprasert, S. and 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. and Ushio, T. (2016b). Optimal stabilizing supervisor of quantitative discrete event systems under partial observation. *IEICE Trans. on Fund. of Elect, Commu. and Computer Science*, 99(2), 475–482.

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 Systems.*, 456–463.

Schmidt, K.W. and Breindl, C. (2014). A framework for state attraction of discrete event systems under partial observation. *Information Sciences*, 281, 265–280.

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

Wonham, W.M. and Cai, K. (2018). *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.