IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 April 2023, accepted 7 June 2023, date of publication 13 June 2023, date of current version 20 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3285432

RESEARCH ARTICLE

Reducing the Learning Time of Reinforcement Learning for the Supervisory Control of Discrete Event Systems

JUNJUN YANG[1], KAIGE TAN[2], (Member, IEEE), LEI FENG[2], (Member, IEEE), AHMED M. EL-SHERBEENY[3], AND ZHIWU LI[1], (Fellow, IEEE)

[1]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[2]Department of Machine Design, KTH Royal Institute of Technology, 10044 Stockholm, Sweden
[3]Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia

Corresponding author: Lei Feng (lfeng@kth.se)

The work is supported by the Chinese Scholarship Council (CSC), KTH Centre of Excellence in Production Research (XPRES), and in part by the National Key R&D Program of China under Grant 2018YFB1700104, the National Natural Science Foundation of China under Grant 61873342, the Science Technology Development Fund, Macau Special Administrative Region (MSAR), under Grant 0064/2021/A2, the Special Fund for Scientific and Technological Innovation Strategy of Guangdong Province under Grant No. 2022A0505030025, and the Researchers Supporting Project (Number RSP2023R133), King Saud University, Riyadh, Saudi, Arabia.

ABSTRACT Reinforcement learning (RL) can obtain the supervisory controller for discrete-event systems modeled by finite automata and temporal logic. The published methods often have two limitations. First, a large number of training data are required to learn the RL controller. Second, the RL algorithms do not consider uncontrollable events, which are essential for supervisory control theory (SCT). To address the limitations, we first apply SCT to find the supervisors for the specifications modeled by automata. These supervisors remove illegal training data violating these specifications and hence reduce the exploration space of the RL algorithm. For the remaining specifications modeled by temporal logic, the RL algorithm is applied to search for the optimal control decision within the confined exploration space. Uncontrollable events are considered by the RL algorithm as uncertainties in the plant model. The proposed method can obtain a nonblocking supervisor for all specifications with less learning time than the published methods.

INDEX TERMS Discrete event system, linear temporal logic, supervisory control theory, reinforcement learning.

I. INTRODUCTION

Discrete event systems (DESs) [1], [2] encompass a wide variety of human-made systems, including power systems [3], unmanned aerial vehicle systems [4], healthcare service systems [5], traffic systems [6], communication protocols [7], [8], digital systems [9] and robotic systems [10], [11]. A DES typically uses deterministic finite automata (DFA) to describe a plant system and its control specification. Reference [12] propose the supervisory control theory (SCT) to compute a supervisor that prevents certain controllable events based on the state of the plant to ensure the controlled system satisfies all specifications, is nonblocking, and has maximal freedom.

An essential concept in SCT is that certain events are uncontrollable and hence cannot be prevented by the supervisor. If the occurrence of an uncontrollable event leads to the violation of a specification, the supervisor must prevent certain controllable events in advance to preempt the occurrence of the uncontrollable event. SCT develops many theories and algorithms for computing the maximally permissive and nonblocking supervisors with acceptable computational complexities. Both the plant and the specification are modeled by deterministic finite automata or regular languages [13]. Many common specifications, however, cannot be modeled by DFAs. For example,

The associate editor coordinating the review of this manuscript and approving it for publication was Abderrahmane Lakas.

''Eventually a workpiece is produced in a production line,'' ''It is always true that if condition A happens, result B will eventually happen.'' These specifications are difficult to model using DFA but can be easily expressed in linear temporal logic (LTL) [14]. Thistle and Lamouchi [15] synthesize a supervisor for a partially observed DES with DFA plant and LTL specifications. Lacerda [16] proposes a methodology to compute a supervisory for a DES with LTL specifications, where the plants are modeled by either DFAs or Petri nets. Sakakibara and Ushio [17] propose an online supervisory control scheme for DESs, where a control specification is described by a fragment of linear temporal logic. Despite theoretical contributions, these methods suffer from intractable computational complexity and are hence difficult for applications.

RL [18], [19], [20] is an effective method for solving control problems with LTL specifications. Smith et al. [21] shows a technique to find an optimal path of a robot to satisfy prespecified control requirements modeled by LTL formulas. Lindemann et al. [22] propose an approach to compute a controller of a coupled multi-robot system under linear temporal logic and signal temporal logic requirements. Sadigh et al. [23] consider a plant as a labeled MDP and the specification is given as an LTL formula. They propose a method for designing a controller based on temporal difference learning [18]. A reward is assigned to each state of the product MDP according to the acceptance condition of the DRA. The reinforcement learning method is employed to obtain an optimal controller that maximizes the expected discounted sum of rewards and satisfies the LTL formula. Hiromoto and Ushio [24] use a labeled MDP with a control cost for system modeling and propose a reinforcement learning algorithm to design an optimal controller. Their algorithm has two steps. First, it removes the actions that do not satisfy LTL specifications. Second, it selects an action that fulfills the LTL specification and minimizes the discount sum of the cost.

One common challenge of these methods is the conversion from an LTL formula to a finite automaton. The methods proposed in [23] and [25] use Safra's construction [26] to translate an LTL formula into a deterministic Rabin automaton (DRA). The conversion is implemented by the program ltl2dstar.[1] Compared with ltl2dstar, another tool Rabinizer3[2] is developed [27] to reduce the state size of the DRA corresponding to an LTL formula. Schillinger et al. [28] propose an approach to decompose a single finite LTL specification into several independent tasks with LTL specifications and construct a group of models to reduce the computational complexity. Wolff et al. [29] define a subset of LTL specifications, which significantly reduces the computational complexity of control synthesis for both the deterministic transition systems and Markov decision processes. The reduction is achieved by computing a controller directly from the original plant model instead of converting the LTL specification into a DRA. Wolff and Murray [30] propose a computationally efficient approach to obtain an optimal trajectory with LTL specification by encoding LTL formulas as mixed-integer linear constraints on the original system, which avoids the computationally expensive processes of creating a finite abstraction of plant and an $\omega-$automaton [31] for the specification. Tumova and Dimarogonas [32] propose a two-phase automaton method to synthesize a controller for multi-components with LTL specifications such that the large computational complexity is reduced. Hasanbeig et al. [33] convert an LTL formula to a limit-deterministic Büchi automaton and define a synchronous reward function to synthesize control policies for an MDP with LTL specifications. Cai et al. [34] divide a complex LTL specification into several simple modules, and propose a modular deep reinforcement learning method for an MDP environment with unknown transition probabilities. These approaches do not consider the controllability of events and hence are not suitable for computing supervisors.

The presence of uncontrollable events in DES control problems is often considered through Rabin games, where the uncontrollable events are modeled as an adversary player [35], [36], [37]. The control specification can be modeled by both finite automata and LTL formulas. However, it leads to a high computational complexity. Zielinski et al. [38] propose a general methodology that applies reinforcement learning to an industrial DES. They first synthesize a supervisor through SCT and then translate the controller to an MDP. Finally, they apply State-Action-Reward-State-Action algorithm [39] to the MDP and obtain an optimal supervisor that satisfies the specifications, where the controllability of events is considered in the proposed action selection algorithm. However, the method requires the assumption of some special transition probabilities of the original subsystem.

This study integrates SCT and an RL method to reduce the learning time of solving DES supervisory control problems with LTL specifications. We model a plant by a labeled DFA and the specifications as LTL formulas. Some types of LTL formulas can be modeled by normal DFAs. For example, a property must always hold. This category of LTL constraints is referred to as the invariant property [14]. Our method begins by utilizing SCT to generate a maximally permissive and nonblocking supervisor for the invariant constraints. Standard algorithms and tools for computation are available for synthesizing the supervisor [40]. The supervisor is extended to a labeled DFA, called labeled supervisor by considering the labels of the plant. The remaining LTL formula is then converted into a DRA, which is a special type of deterministic finite automaton capable of representing LTL formulas [41]. We construct a Rabin product DFA from the DRA and the labeled supervisor. The RL method is applied to find a deterministic state-based controller for the remaining LTL formula by considering the uncontrollable events as uncertain transitions. The controller determines

[1] https://www.ltl2dstar.de/

[2] https://www7.in.tum.de/ kretinsk/rabinizer3.html

one action at a state and is not yet a supervisor. All uncontrollable events must be added to the allowed action subset at every state to change the controller into a supervisor. The generated supervisor ensures the plant satisfies all specifications modeled by both DFA and LTL formulas.

The contributions of the paper are twofold as follows.

- To support control synthesis by RL, we propose an algorithm to determine the next state and the corresponding reward by considering uncontrollable events as uncertain transitions.
- We integrate SCT and RL for the supervisor synthesis of discrete event systems with LTL specifications. SCT can eliminate infeasible transitions, thereby reducing the learning time of RL. While the Q function of the RL algorithm in this paper is represented explicitly as a table for illustrating concepts, it can be easily replaced by a deep neural network to further reduce the computational complexity. All methods proposed in this paper apply without change to the RL algorithm using deep neural networks.

The advantages of the proposed method are verified by two case studies: 1) Control of an autonomous vehicle in a construction site. 2) Control of an industrial transfer line [1], [38]. For both cases, we compare the proposed method that integrates SCT and RL with the baseline method of only using RL. The proposed method is shown to require less training time to compute the supervisor. Moreover, we use a model-based deep reinforcement learning algorithm to realize the proposed method in the second case.

II. PRELIMINARIES

A. AUTOMATA AND LINEAR TEMPORAL LOGIC

A labeled deterministic finite automaton (DFA) is defined as $\mathbf{G} = \langle Q, A, \xi, q_0, Q_m, \Pi, L\rangle$, where

- Q is a finite nonempty state set;
- A is a finite nonempty set, the alphabet of transition labels;
- $\xi : Q \times A \to Q$ is the partially defined transition function;
- $q_0 \in Q$ is the initial state;
- $Q_m \subseteq Q$ is the set of marker states;
- Π is a nonempty set of atomic propositions;
- $L : Q \to 2^{\Pi}$ is the label function on states.

Given a word $\omega = uv$ with $u, v \in A^*$, u is called a prefix of ω, denoted by $u \preceq \omega$. Given $q \in Q, \sigma \in A$, if $\xi(q, \sigma)$ is defined, write $\xi(q, \sigma)!$. ξ is extended to a function: $Q \times A^* \to Q$. Accordingly, given $str \in A^*$, write $\xi(q, str)!$ if $\xi(q, str)$ is defined. $\mathcal{L}(\mathbf{G}) = \{str \in A^* \mid \xi(q_0, str)!\}$ is the generated language and $\mathcal{L}_m(\mathbf{G}) = \{str \in A^* \mid \xi(q_0, str) \in Q_m\}$ is the marked language of $\mathbf{G}$. Let $\mathcal{L} \subseteq A^*$ be a language, $\overline{\mathcal{L}} = \{u \in A^* \mid (\exists \omega \in \mathcal{L}) u \preceq \omega\}$ is the prefix closure of $\mathcal{L}$. We say that $\mathbf{G}$ is nonblocking if $\mathcal{L}(\mathbf{G}) = \overline{\mathcal{L}_m(\mathbf{G})}$. The alphabet A is composed of two subsets A_c and A_u, where A_c and A_u respectively represent the sets of controllable and uncontrollable events in A.

An atomic proposition $a \in \Pi$ is a logic statement about the states in Q and returns true or false. A propositional logic formula is a proposition that combines atomic propositions using Boolean operators $\wedge$ (and), $\vee$ (or), ! (negation), and $\to$ (implication). A state formula is a proposition whose value is completely determined by the combination of states and the labels on these states. An LTL formula is composed of state formulas and temporal operators $\Box$ (always), $\Diamond$ (eventually), X (next), U_W (weak until), and U (strong until). Given a state sequence $v = q_0 q_1 q_2 \ldots$ of $\mathbf{G}$ and a set of state formulas $\{\phi, \varphi, \ldots\}$, let $v_i = q_i$ be the i-th state along the state sequence and $v[i] = q_i q_{i+1} \ldots$ be the suffix starting from the i-th state. The semantics of the most common temporal logic formulas are described as follows.

- $v \vDash \phi$ iff $v_0 \vDash \phi$,
- $v \vDash X\phi$ iff $v_1 \vDash \phi$,
- $v \vDash \Box\phi$ iff $(\forall i \geq 0)\ v_i \vDash \phi$,
- $v \vDash \Diamond\phi$ iff $(\exists i \geq 0)\ v_i \vDash \phi$,
- $v \vDash \phi \cup_W \varphi$ iff $v_0 \vDash \varphi \vee (v_0 \vDash \phi \wedge v[1] \vDash \phi \cup_W \varphi)$,
- $v \vDash \phi \cup \varphi$ iff $v \vDash \phi \cup_W \varphi \wedge \Diamond\varphi$.

The difference of $\phi \cup \varphi$ (strong until) and $\phi \cup_W \varphi$ (weak until) is that φ in $\phi \cup \varphi$ must become true in the future but φ in $\phi \cup_W \varphi$ may or may not become true. More complex LTL formulas are composed of simple LTL formulas and logic connectives.

A deterministic Rabin automaton (DRA) is defined as $\mathcal{R} = \langle S, \Pi, \beta, s_0, F\rangle$, where

- S is a finite set of states;
- Π is a finite set of atomic propositions;
- $\beta : S \times 2^{\Pi} \to S$ is the state transition function;
- $s_0 \in S$ is the initial state;
- $F = \{(D_1, B_1), \ldots, (D_k, B_k)\}$ is the acceptance condition where $D_i, B_i \subseteq S$ for $i = 1, \ldots, k$ and (D_i, B_i) is called an acceptance pair of $\mathcal{R}$.

A DRA is used to model an LTL specification [41]. An LTL formula ϕ can be translated into a corresponding DRA, and this study employs Rabinizer 3 [27], [42] to do this conversion. Most LTL specifications only have one acceptance pair, which simplifies the control synthesis algorithm. Thus the acceptance condition is written as $F = (D, B)$ throughout the rest of the paper.

Let $v = s_0 s_1 s_2 \ldots$ be an infinite sequence of states generated by $\mathcal{R}$ and $\inf(v)$ denote the set of states that appear infinitely often in v. If v satisfies

$$\inf(v) \cap D \neq \emptyset \text{ and } \inf(v) \cap B = \emptyset, \tag{1}$$

then we say that v is accepted by $\mathcal{R}$. Note that given a pair (D, B), B can be the empty set but D is always not.

B. THE OPTIMAL CONTROLLER FOR AN MDP

A Markov decision process (MDP) is defined as $\mathbf{M} = \langle Q, A, P, q_0, t\rangle$, where

- Q is a finite nonempty state set;
- A is a finite nonempty set of actions;
- $P : Q \times A \times Q \to [0, 1]$ is the transition probability function;

- $q_0 \in Q$ is the initial state;
- $t : Q \times A \times Q \to \mathbb{R}$ is the reward function;

A controller of an MDP $\mathbf{M} = \langle Q, A, P, q_0, t\rangle$ is a function $f : Q \to A$. Applying a fixed f to $\mathbf{M}$, we obtain a Markov chain $\mathbf{M}_f$, whose state space is Q and the probability from state q_i to state q_j is denoted by $P(q_i, f(q_i), q_j)$. A *run* of $\mathbf{M}_f$ is an ordered, infinite sequence of transitions $\mathbf{r} = (q_0, \sigma_0, q_1), (q_1, \sigma_1, q_2), (q_2, \sigma_2, q_3)\ldots$ such that for all $i \geq 0, P(q_i, \sigma_i, q_{i+1}) > 0$ and $\sigma_i = f(q_i)$. Each *run* corresponds to a sequence of states $\mathbf{v} = sta(\mathbf{r}) = q_0q_1q_2q_3\cdots$ and a word $\omega = act(\mathbf{r}) = \sigma_0\sigma_1\sigma_2\cdots$. For a transition from state q to q' by the event σ, the reward is denoted by $t(q, \sigma, q')$.

Given an MDP $\mathbf{M} = \langle Q, A, P, q_0, t\rangle$ and a controller f, the discounted total reward for a *run* $\mathbf{r}$ of the Markov chain $\mathbf{M}_f$ is defined as

$$R(\mathbf{r}) = \sum_{n=0}^{\infty} \gamma^n t(q_n, f(q_n), q_{n+1}), \tag{2}$$

where $0 < \gamma \leq 1$ is the discount factor that reduces the weight of rewards in the future, and $t(q_n, \sigma_n, q_{n+1})$ is the step reward of the transition (q_n, σ_n, q_{n+1}), where $q_n, q_{n+1} \in Q, \sigma_n = f(q_n)$, and $P(q_n, \sigma_n, q_{n+1}) \neq 0$.

Let $q \in Q$ be the initial state of a *run* $\mathbf{r}$ of $\mathbf{M}_f$, denoted by $q = sta(\mathbf{r})_0$. The expectation of the discounted rewards of all infinite runs of $\mathbf{M}_f$ starting from the state q is defined as a value function

$$U_f(q) = E_f\{R(\mathbf{r}) \mid q = sta(\mathbf{r})_0\} \tag{3}$$

of $\mathbf{M}$ under f, where $E_f\{\cdot\}$ is the expected value. According to the Bellman equation [18], the value function of state q satisfies

$$U_f(q) = \sum_{q' \in Q} P(q, f(q), q')[t(q, f(q), q') + \gamma U_f(q')]. \tag{4}$$

A controller for $\mathbf{M}$ that maximizes the expected value for every state $q \in Q$ is an optimal controller f^*, which is formulated as

$$f^* = \arg\max_f \sum_{q' \in Q} P(q, f(q), q')[t(q, f(q), q') + \gamma U_f(q')]. \tag{5}$$

If the MDP model including the state transition probabilities is available, the optimal controller f^* can be found by stochastic dynamic programming with an infinite horizon. If the MDP model is not available, then RL is applicable for approximating the optimal controller via randomly generated simulations.

C. SUPERVISORY CONTROL THEORY (SCT)

SCT is a branch of systems control regarding the logical behavior of a DES [12]. A plant is modeled by a DFA $\mathbf{G}$ with an input set A that consists of two disjoint subsets A_c and A_u, where A_c is the set of controllable events and A_u is that of uncontrollable events. In SCT of DESs, the specification that the plant necessarily satisfies is modeled by a DFA, denoted by $\mathbf{E}$, which has the same event set as $\mathbf{G}$.

If $\mathbf{G}$ does not satisfy the specification, a supervisor needs to be synthesized to prevent the occurrences of some controllable events such that the plant satisfies the control requirement. The supervisory control function is $V : \mathcal{L}(\mathbf{G}) \to \Gamma$, where $\Gamma = \{\gamma \in 2^A | A_u \subseteq \gamma \subseteq A\}$ is a set of allowed event subsets. The definition implies that uncontrollable events must always be allowed to happen. The definition also emphasizes a fundamental difference between the controller obtained by (5) and the supervisor. A controller determines one action to take at the current state, but a supervisor determines a subset of events that may occur at the current state and the occurrences of these events are uncertain for the supervisor. The events outside the determined subset must be prevented by the supervisor. The function V restricts the behavior of $\mathbf{G}$ to a supervisor DFA $\mathbf{S}$, such that $\mathcal{L}_m(\mathbf{S}) \subseteq \mathcal{L}_m(\mathbf{G}) \cap \mathcal{L}_m(\mathbf{E})$, $\mathbf{S}$ is nonblocking, and the language $\mathcal{L}(\mathbf{S})$ is controllable with respect to $\mathbf{G}$ and A_u. The SCT theory ensures the existence of the maximally permissive supervisor $\mathbf{S}^*$ such that the language of any other supervisor $\mathbf{S}$ for $\mathbf{G}$ and $\mathbf{E}$ is a subset of the language of $\mathbf{S}^*$, i.e., $\mathcal{L}_m(\mathbf{S}) \subseteq \mathcal{L}_m(\mathbf{S}^*)$.

Many computational tools are available for computing the maximally permissive nonblocking supervisor for a given pair of plant DFA $\mathbf{G}$ and the specification DFA $\mathbf{E}$. This study applies TCT [40].

III. INTEGRATION OF SCT AND RL

This section introduces a method that integrates SCT and RL to compute a supervisor for a DES to satisfy the LTL specifications.

A. PROBLEM FORMULATION

Given a labeled DFA $\mathbf{G} = \langle Q, A, \xi, q_0, Q_m, \Pi, L\rangle$ with the LTL specification $\phi = \phi_1 \wedge \phi_2$, where ϕ_1 describes the invariant property and ϕ_2 describes others. Our objective is to compute a supervisor that ensures $\mathbf{G}$ to satisfy ϕ.

Our solution is shown in Fig. 1. To compute the supervisor, we divide the procedure into three parts: satisfaction of the invariant property ϕ_1, computing the optimal controller for the remaining LTL specification ϕ_2, and construction of the final supervisor. The details of each part are described as follows.

B. SATISFACTION OF THE INVARIANT SPECIFICATIONS

As shown in Fig. 1, a part of the LTL specification, namely ϕ_1 describes the invariant constraints. An invariant constraint describes a property that the system must satisfy at all reachable states. For instance, the requirement that an autonomous vehicle must avoid the obstacles in its path is modeled by an LTL formula $\phi_1 = \Box\neg obstacle$. This type of LTL constraint can be modeled by DFAs, and can be guaranteed by SCT.

Given the plant $\mathbf{G} = \langle Q, A, \xi, q_0, Q_m, \Pi, L\rangle$ and the DFA $\mathbf{E}_1$ that describes ϕ_1, TCT generates a maximal permissive nonblocking supervisor $\mathbf{S}'_1 = \langle Q_1, A, \xi_1, q_{1,0}, Q_{1,m}\rangle$ without labels. To connect the supervisor and the LTL formula, we need to construct a labeled supervisor that has the same

language as $\mathbf{S}_1'$ and consistent labels as $\mathbf{G}$. Denote the labeled supervisor $\mathbf{S}_1 = \langle Q_1, A, \xi_1, q_{1,0}, Q_{1,m}, \Pi, L_1 \rangle$. It must have the following properties

- $\mathbf{S}_1$ is nonblocking,
- $\mathcal{L}_m(\mathbf{S}_1) \subseteq \mathcal{L}_m(\mathbf{G}) \cap \mathcal{L}_m(\mathbf{E}_1)$,
- $\mathbf{S}_1$ is a supervisor w.r.t. $\mathbf{G}$ and A_u.
- $(\forall str \in \mathcal{L}(\mathbf{S}_1))L_1(\xi_1(q_{1,0}, str)) = L(\xi(q_0, str))$.

The labeled supervisor $\mathbf{S}_1$ can be obtained from the product $\mathbf{G} \times \mathbf{S}_1'$, and the label at every state (q, q_1) is $L_1(q, q_1) = L(q)$.

C. SATISFACTION OF THE REMAINING LTL SPECIFICATIONS

This part is devoted to the computation of an optimal controller that satisfies the remaining LTL specification by an RL algorithm.

As depicted in Fig. 1, the remaining portion of the LTL specification, namely ϕ_2 is converted to a DRA $\mathcal{R}_2 = \langle S, \Pi, \beta, s_0, F \rangle$ by the open-source software tool Rabinizer 3, but other tools for converting LTL to DRA are applicable. To obtain the supervisor for $\mathbf{G}$ that satisfies both the invariant specification ϕ_1 and ϕ_2, a Rabin product DFA from the labeled supervisor $\mathbf{S}_1$ and the DRA $\mathcal{R}_2$ is created as follows.

Given $\mathbf{S}_1 = \langle Q_1, A, \xi_1, q_{1,0}, Q_{1,m}, \Pi, L_1 \rangle$ and $\mathcal{R}_2 = \langle S, \Pi, \beta, s_0, F \rangle$, a Rabin product DFA is defined as a tuple $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2 = \langle X, A, T, x_0, \mathcal{F}, W \rangle$, where

- $X \subseteq Q_1 \times S$ is the set of states;
- $A = A_c \cup A_u$ is identical to it in $\mathbf{S}_1$, where A_c is the control event set and A_u is the uncontrollable event set;
- $T : X \times A \to X$ is the partial transition function. For $x = (q, s)$ and $\sigma \in A$,

$$T(x, \sigma) = \begin{cases} (q', s') & \text{if } q' = \xi_1(q, \sigma) \\ & \quad \wedge\, s' = \beta(s, L_1(q')) \\ \text{undefined} & \text{if } \xi_1(q, \sigma) \text{ is undefined} \end{cases}$$

- $x_0 = (q_0, s_0) \in X$ is the initial state of $\mathbf{P}$;
- $\mathcal{F} = (\mathcal{G}, \mathcal{B})$ is the acceptance condition, where $\mathcal{G} = Q_1 \times D, \mathcal{B} = Q_1 \times B$, and $(D, B) = F$ is defined in $\mathcal{R}_2$;
- $W : X \to \mathbb{R}$ is the reward function. For any state $x \in X$,

$$W(x) = \begin{cases} w_G & \text{if } x \in \mathcal{G} \\ w_B & \text{if } x \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

where $w_G > 0$ is a positive reward and $w_B < 0$ is a negative reward.

An event $\sigma \in A$ is called enabled at a state $x \in X$ if $(\exists x' \in X)x' = T(x, \sigma)$. A set that includes all enabled events at a state x is defined as $\mathbf{Enb}(x) = \{\sigma \in A \mid \sigma \text{ is enabled at } x\}$. For the Rabin product DFA $\mathbf{P}$, the one-step reward for transition (x_{n-1}, σ, x_n) is defined as $t(x_{n-1}, \sigma, x_n) = W(x_n)$.

An essential assumption of SCT is that the occurrence of uncontrollable events is not preventable. If a controllable event $\sigma \in \mathbf{Enb}(x)$ is selected at a state x by a controller, all other controllable events defined at state x are necessarily disabled; however, the uncontrollable events defined at the state x may still occur. Similarly, if the controller selects an uncontrollable event σ at x, then all controllable events defined at x are disabled but other uncontrollable events defined at x may still occur. The uncertainty of the uncontrollable events makes the Rabin product DFA $\mathbf{P}$ a Markov Decision Process (MDP) with an uncertain state transition relationship. At a state $x = (q, s)$, the occurrence of event $\sigma \in \mathbf{Enb}(x)$ randomly determines the next state by Algorithm 1.

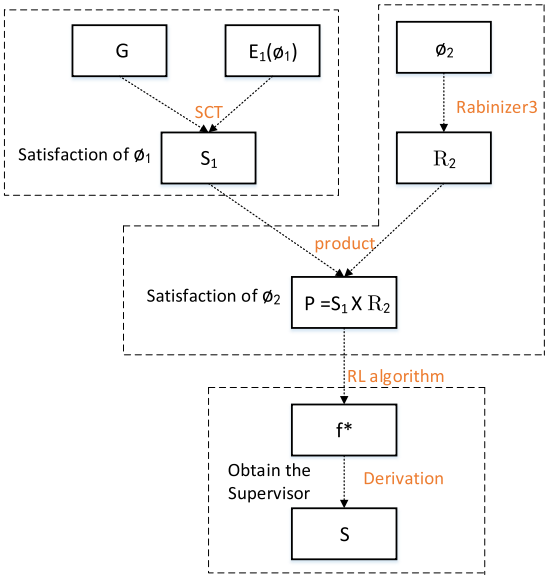


FIGURE 1. Framework for the integration of SCT and reinforcement learning.

Algorithm 1 Random Selection of the Next State

Input: $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2, x = (q, s), \sigma \in \mathbf{Enb}(x)$
Output: $x', t(x, \sigma, x')$

1: $\Sigma \leftarrow (\{\sigma\} \cup A_u) \cap \mathbf{Enb}(x)$;
2: Randomly select an event $\sigma' \in \Sigma$;
3: $x' \leftarrow T(x, \sigma')$;
4: $t(x, \sigma, x') \leftarrow W(x')$.

Algorithm 1 essentially augments the Rabin product DFA P to an MDP. Similar to (5), the optimal controller for $\mathbf{P}$ that satisfies ϕ_2 can be found by (6).

$$f_{\mathbf{P}}^* = \arg\max_{f_{\mathbf{P}}} \sum_{x' \in X} P(x, f_{\mathbf{P}}(x), x')[t(x, \sigma, x') + \gamma U(x')]. \quad (6)$$

If a controller for $\mathbf{P}$ satisfying the remaining LTL property ϕ_2 exists, $f_{\mathbf{P}}^*$ is a satisfying controller after a proper selection of the reward function W [23], [43]. On the other hand, even though $f_{\mathbf{P}}^*$ maximizes the value function of $\mathbf{P}$, it may not ensure the $\mathbf{P}$ to satisfy ϕ_2 because of the uncertainty of the MDP. Therefore, the optimal controller $f_{\mathbf{P}}^*$ needs further verification against ϕ_2 by simulations or model checking [14].

Based on the Rabin product DFA $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$, a state-action value function $Q_f : X \times A \to \mathbb{R}$ for the controller f is

defined as the expected return if the starting state is x and the first action at x is σ, and the controller f is applied from the next state:

$$Q_f(x,\sigma) = E_f\{\sum_{n=0}^{\infty}\gamma^n t(x_n,\sigma_n,x_{n+1}) \mid x_0 = x, \sigma_0 = \sigma, (\forall n > 0)\sigma_n = f(x_n)\}.$$

According to the Bellman equation [44] and the Markov property, the state-action value under f can be written as

$$\begin{aligned} Q_f(x,\sigma) &= \sum_{x'\in X} P(x,\sigma,x')[t(x,\sigma,x') \\ &\quad + \gamma E_f\{\sum_{n=0}^{\infty}\gamma^n t(x_n,f(x_n),x_{n+1})|x_0 = x'\}] \\ &= \sum_{x'\in X} P(x,\sigma,x')[t(x,\sigma,x') + \gamma Q_f(x',f(x'))]. \\ Q^*(x,\sigma) &= \sum_{x'\in X} P(x,\sigma,x')[t(x,\sigma,x') + \gamma \max_{\sigma'} Q^*(x',\sigma')]. \end{aligned} \tag{7}$$

Since the state transition probabilities of the Rabin product DFA $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$ are unknown, $P(x,\sigma,x')$ in (7) is also unknown. The optimal controller for (7) may be found by model-free methods. Q-learning [18], [45] is an off-policy reinforcement learning method popular and effective for unknown MDPs with discrete control actions. The temporal difference learning method updates the Q function along with a sampled episode. Based on the Rabin product DFA $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$, the temporal difference error is defined as

$$e = t(x,\sigma,x') + \gamma \max_{\sigma'} Q^*(x',\sigma') - Q^*(x,\sigma),$$

and the optimal Q function is updated as

$$\begin{aligned} Q^*(x,\sigma) \leftarrow Q^*(x,\sigma) &+ \alpha[t(x,\sigma,x') \\ &+ \gamma \max_{\sigma'} Q^*(x',\sigma') - Q^*(x,\sigma)], \end{aligned}$$

where α is the learning rate.

Algorithm 2 is the Q-learning algorithm used in this study. The balance between exploitation and exploration is coordinated by the ϵ-greedy algorithm. The probability threshold is $\epsilon \in (0,1)$. After each episode, ϵ decreases by a decay rate *Decay*, and the minimal value of ϵ is 0.01. At a state x, the Q-learning algorithm has the probability ϵ for randomly selecting an event in $\mathbf{Enb}(x)$, i.e., all events defined at x. To this end, a random number $r \in [0,1]$ is generated by the even probability distribution. If $r \leq \epsilon$, a random event $\sigma \in \mathbf{Enb}(x)$ is selected; otherwise, $\sigma = \arg\max_{\sigma} Q^*(x,\sigma)$.

After selecting an event σ at a state x, Algorithm 2 calls Algorithm 1 at Line 12 to further consider the uncertainties caused by uncontrollable events. Then the value function is updated as Line 13 of Algorithm 2.

There are three kinds of terminal situations in an episode. The first one is that x is a blocking state with $\mathbf{Enb}(x) = \emptyset$. The second one is that $x = (q,s)$ and the state s satisfies $s \in B$ and $(\forall l \in 2^{\Pi})\beta(s,l) = s$, where B is a component of the acceptance pair of the DRA. Here we use **Bad** to denote the collection of the states that satisfy the second condition. The third situation is that the step counter *StepNumber* in each episode is larger than the step threshold *MaxiStep*. When one of the three situations mentioned above occurs, the current episode is terminated and the algorithm starts a new episode from the initial state x_0.

Algorithm 2 is designed for finite Markov decision processes and the convergence has been proved in [45]. The three variables *EpisodeReward*, *EpisodeNumber* and *AverageReward* respectively represent the reward obtained for each episode, the episode number, and the average reward over the previous *Length* episodes. If the value of *AverageReward* is larger than the threshold Δ or *EpisodeNumber* is larger than the limit *MaxiEpi*, we

Algorithm 2 Q-Learning for Computing an Optimal Controller

Input: $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$, Discount factor γ, learning rate α, probability ϵ, decay rate *Decay*, average reward threshold Δ, maximal steps *MaxiStep*, maximal episodes *MaxiEpi* and queue length *Length*

Output: An optimal controller $f^*_{\mathbf{P}}$

1: Initialize $Q(x,\sigma)$ with random values;
2: *EpisodeNumber* $\leftarrow 0$;
3: *AverageReward* $\leftarrow 0$;
4: Let *Que* be a circular queue with length *Length*;
5: Initialize *Que* as empty;
6: **while** *AverageReward* $< \Delta$ and *EpisodeNumber* $<$ *MaxiEpi* **do**
7: *EpisodeReward* $\leftarrow 0$;
8: $x \leftarrow x_0$;
9: *StepNumber* $\leftarrow 0$;
10: **while** $\mathbf{Enb}(x) \neq \emptyset$ and $x \notin$ **Bad** and *StepNumber* $\leq$ *MaxiStep* **do**
11: Choose an event $\sigma \in \mathbf{Enb}(x)$ by ϵ-greedy policy;
12: Call Algorithm 1;
13: $Q(x,\sigma) \leftarrow Q(x,\sigma) + \alpha[t(x,\sigma,x') + \gamma \max_{\sigma'} Q(x',\sigma') - Q(x,\sigma)]$;
14: *EpisodeReward* $\leftarrow$ *EpisodeReward* $+ t(x,\sigma,x')$;
15: $x \leftarrow x'$;
16: *StepNumber* $\leftarrow$ *StepNumber* $+ 1$;
17: **end while**
18: **if** $\epsilon > 0.01$ **then**
19: $\epsilon \leftarrow \epsilon \times (1 - Decay)$;
20: **end if**
21: *EpisodeNumber* $\leftarrow$ *EpisodeNumber* $+ 1$;
22: Add *EpisodeReward* to *Que*;
23: *SumReward* $\leftarrow \sum_{n\in Que} n$;
24: *AverageReward* $\leftarrow$ *SumReward*$/len($*Que*$)$;
25: **end while**
26: $f^*_{\mathbf{P}}(\cdot) \leftarrow \arg_{\sigma}\max Q(\cdot,\sigma)$;

terminate the iteration and return an optimal controller for $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$ that satisfies the remaining LTL specification. If there are multiple actions at state x with the identical value of $\max Q(x, \sigma)$, then the algorithm chooses one randomly. To calculate the average reward of the last *Length* episodes, we define a circular queue *Que* with the fixed length *Length* and initialize it as empty. An episode reward is added to *Que* when an episode is finished.

D. COMPUTATION OF THE FINAL SUPERVISOR

The sections above obtain an optimal controller that satisfies the invariant constraint and the remaining LTL specification. We cannot use the optimal controller as a supervisor because it only selects one event at a state. This controller obtained by Algorithm 2 does not contain all feasible uncontrollable events at the state. As a result, the controller is not a supervisor in SCT. This part considers the uncontrollable events and computes a supervisor restricting the plant to satisfy the whole control specification.

Given the Rabin product DFA $\mathbf{P} = \langle X, A, T, x_0, \mathcal{F}, W\rangle$ from $\mathbf{S}_1 = \langle Q_1, A, \xi_1, q_{1,0}, Q_{1,m}, \Pi, L_1\rangle$ and $\mathcal{R}_2 = \langle S, \Pi, \beta, s_0, F\rangle$ and the optimal controller $f^*_{\mathbf{P}}$, we construct the final supervisor $\mathbf{S} = \langle X', A, T', x_0, X_m\rangle$ as follows.

- $X' \subseteq X$ is the set of reachable states;
- A and x_0 are identical to them in $\mathbf{S}$ and $\mathbf{P}$, respectively;
- $T' : X' \times A \to X'$ is the partial transition function. For a state $x \in X'$ and an event $\sigma \in A$,

$$T'(x, \sigma) = \begin{cases} T(x, \sigma) & \text{if } \sigma \in \{f^*_{\mathbf{P}}(x)\} \cup A_u \\ \text{undefined} & \text{otherwise;} \end{cases}$$

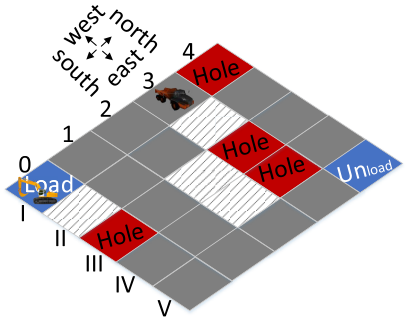
- $X_m = (Q_{1,m} \times S) \cap X'$ is the marker state set.

IV. CASE STUDIES

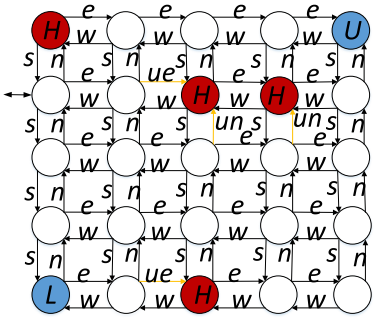
In this part, we apply our method to two cases and compare the results of our method with the approach that only uses the RL method. The two methods are compared in terms of the sizes of the state spaces, the learning times, and the number of learning iterations. The consequence is that our proposed method spends less time obtaining an optimal policy. In addition, the proposed method is realized by a model-based deep reinforcement learning algorithm in the second case.

A. CONTROL OF AN AUTONOMOUS VEHICLE IN A CONSTRUCTION SITE

We explain the proposed method in this paper through an illustrative example. As shown in Fig. 2(a), an autonomous truck travels in a square construction site to move earth from the load station to the unload station. The construction site is partitioned into a 5 by 5 grid world. The rows are numbered from 0 to 4 from south to north and the columns are numbered from I to V from west to east. A block located in row r and column c is denoted by (r, c). The truck is initialized at (3, I), and the blocks in this grid world are divided into four types: the grids with labels *Load*, *Unload*, *Hole*, and none. The



(a) The construction site.



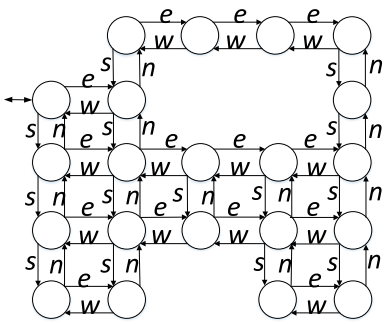
(b) The labeled DFA model $\mathbf{G}$ of the construction site.

FIGURE 2. Environment.

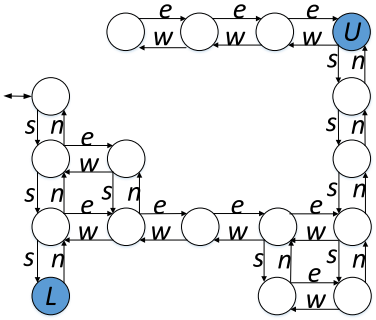
grids for *Load* and *Unload* are colored in blue. The girds with *Hole* are colored in red. At some grids adjacent to the holes, the ground is uneven and has steep slopes into the *Hole* area. These grids are colored in white. The labels *Load*, *Unload*, and *Hole* respectively represent the load station, the unload station, and the holes in the construction site. The truck can execute four actions: *north*, *south*, *west* and *east* in this construction site. The action *north* makes the truck move north. Similarly, the vehicle respectively moves south, west, and east when it executes *south*, *west*, and *east*. The truck is allowed to choose an arbitrary action except at the boundary of the environment. For instance, at location (4, I), the vehicle is only allowed to select an action from *south* and *east* since actions *north* and *west* will make it hit the boundary. The truck must satisfy the following constraints:

1. Never reach the grids with a hole.
2. Travel between the load and unload stations infinitely often.

The grid world is modeled as the labeled DFA $\mathbf{G}$ in Fig. 2(b). The nodes and directed edges respectively denote the states and transitions of $\mathbf{G}$. The state with a double arrow is the initial state and the arrows in yellow describe the uncontrollable events. For convenience, we use n, s, w, e, un, and ue to respectively denote actions *north*, *south*, *west*, *east*, *unorth* and *ueast*, where *unorth* and *ueast* are two uncontrollable events generated when the vehicle goes to the white locations. The event set, controllable event set, and uncontrollable set of $\mathbf{G}$ are respectively expressed by $\Sigma = \{n, s, w, e, un, ue\}$, $A_c = \{n, s, w, e\}$ and $A_u = \{un, ue\}$.



(a) The DFA $\mathbf{E}$ corresponds to the constraint $\square\neg H$.



(b) The labeled supervisor $\mathbf{S}_1$.

FIGURE 3. Specification and the supervisor.

Capital letters L, H and U in nodes separately represent the labels *Load*, *Hole*, and *Unload* of the grid world. The labels of nodes in white are $\emptyset$.

1) THE PROPOSED METHOD

This part uses the proposed new method that combines SCT and a reinforcement learning method to compute a supervisor that satisfies constraints 1 and 2 for the truck. The plant is shown in Fig. 2(b) and the specification is described as an LTL formula

$$\varphi = \square\Diamond L \wedge \square\Diamond U \wedge \square\neg H. \tag{8}$$

The constraint $\square\neg H$ can be enforced by a regular language specification, and the DFA $\mathbf{E}$ that recognizes the language is shown in 3(a).

The first step is to compute a nonblocking labeled supervisor $\mathbf{S}_1$ from the plant $\mathbf{G}$ and the specification $\mathbf{E}$. This step is realized by SCT and the result is illustrated in Fig. 3(b). With the supervisory control of $\mathbf{S}_1$, the movement of the truck satisfies $\square\neg H$, i.e., the truck never goes to the *Hole* areas. For example, the supervisor disables action n at the initial state since n leads to a *Hole* block. The action e is also disabled at the initial state as the truck may enter a *Hole* block from the shaded block (3, II).

The second step is to compute an optimal controller for the LTL specification ϕ. The remaining LTL formula ϕ is translated into a DRA $\mathcal{R}_\phi$ in Fig. 4. The acceptance condition of $\mathcal{R}_\phi$ is $F = (\{s_2\}, \emptyset)$. The state in blue should be visited

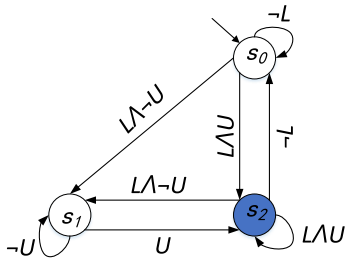
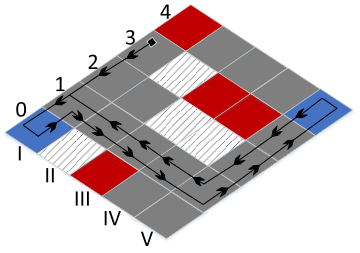
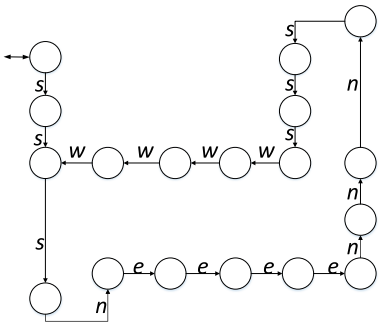


FIGURE 4. The DRA $\mathcal{R}_\phi$ corresponding to $\phi = \square\Diamond L \wedge \square\Diamond U$.



(a) The trajectory induced by $f^*_{\mathbf{P}}$.



(b) The supervisor $\mathbf{S}$.

FIGURE 5. Trajectory and the supervisor.

TABLE 1. Values of parameters.

Parameter	α	γ	ε	w_G	w_B	Δ	$Decay$	$MaxiStep$	$Length$
Value	0.9	0.99	0.9	200	-	209	10^{-4}	25	100

infinitely often, and other states are in white. Then we obtain a Rabin product DFA $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_\phi$ with 33 states.

Applying Algorithm 2 to $\mathbf{P}$, we obtain an optimal controller $f^*_{\mathbf{P}}$. The trajectory induced by $f^*_{\mathbf{P}}$ is shown in Fig. 5(a). The arrows in blocks are the actions selected by the optimal controller. When the truck enters (1, I) from (1, II), the Rabin automaton is at state s_0 and the optimal action is $f^*_{\mathbf{P}}((1, \text{II}), s_0) = west$. On the other hand, when the truck enters (1, III) from (1, II), the Rabin automaton reaches s_1 and the corresponding action is $f^*_{\mathbf{P}_1}((1, \text{II}), s_1) = east$. That is the reason why there is more than one action in a block under the optimal controller.

The values of the parameters used are from Table 1. The negative reward w_B is not used since there does not exist a state that is required to be visited finitely.

Then we use SCT to generate a supervisor $\mathbf{S}$ that allows all feasible uncontrollable events, and $\mathbf{S}$ is shown in Fig. 5(b).

At the initial state, the truck cannot take the controllable action n since $\mathbf{S}$ disables it to avoid the *Hole* block (4, I). Similarly, the action e is disabled at the block (3, I). Although the string se does not lead to any *Hole* block, it is also disabled since the learning algorithm selects the event s at the block (2, I).

The trajectory under the supervisor is shown in Fig. 5(a). The truck starts from the initial state (3, I) and arrives at the loading location (0, I) by executing the action sequence sss. Then it takes action sequence $neeeennn$ and arrives at the unloading location (4, V). After dumping the soil, the truck navigates back to the upload location by the action sequence $sssw wws$. Therefore, we obtain a trajectory

$$\begin{aligned} tra = &(3, \mathrm{I}), (2, \mathrm{I}), ((1, \mathrm{I}), (0, \mathrm{I}), (1, \mathrm{I}), (1, \mathrm{II}), (1, \mathrm{III}), (1, \mathrm{IV}) \\ &(1, \mathrm{V}), (2, \mathrm{V}), (3, \mathrm{V}), (4, \mathrm{V}), (3, \mathrm{V}), (2, \mathrm{V}), (1, \mathrm{V}) \\ &(1, \mathrm{IV}), (1, \mathrm{III}), (1, \mathrm{II}))^{\omega} \end{aligned}$$

for the vehicle. In this trajectory, the truck travels between the load and the unload stations infinitely often and never reaches the *Hole* areas. Therefore, under the supervision of $\mathbf{S}$, the trajectory of the truck satisfies constraints 1 and 2.

2) RL METHOD WITHOUT SCT

To highlight the advantages of the proposed method, this section shows the results of two methods based on RL for computing an optimal controller for the plant $\mathbf{G}$ to satisfy the constraint φ in (8). One directly generates an optimal controller for $\mathbf{G}$. The other considers the uncertainties by uncontrollable events based on the plant model $\mathbf{G}$.

a: RL METHOD WITHOUT CONSIDERING UNCONTROLLABLE EVENTS

The plant model is shown in Fig. 2(b) and the specification is described by (8). The LTL formula φ is translated to a DRA $\mathcal{R}_\varphi$ in Fig. 6, where the acceptance condition of $\mathcal{R}_\varphi$ is $(\{s_2\}, \{s_3, s_4\})$, and s_0 is the initial state. The state in blue is required to be visited infinitely often and the states in red should be avoided. Other states are in white.

We obtain a Rabin product DFA $\mathbf{P}' = \mathbf{G} \times \mathcal{R}_\varphi$ with 102 states. Applying Algorithm 2 to $\mathbf{P}'$, we obtain an optimal controller $f^*_{\mathbf{P}'}$ for $\mathbf{P}'$ to satisfy φ. The trajectory of the truck under $f^*_{\mathbf{P}'}$ is shown in Fig. 7(a). Compared with the trajectory induced by the optimal controller $f^*_{\mathbf{P}}$ in Fig. 5(a), the truck in Fig. 7(a) enters the block (3, II). The values of $\alpha, \gamma, \varepsilon, \Delta, w_G, w_B, Decay, MaxiStep$ and $Length$ used in Algorithm 2 are shown in Table 1, where $w_B = -100$.

The supervisor $\mathbf{S}'$ derived from $\mathbf{P}' = \mathbf{G} \times \mathcal{R}_\varphi$ and $f^*_{\mathbf{P}'}$ is shown in Fig. 7(b). Fig. 7(c) shows the trajectories under $\mathbf{S}'$, in which the truck enters the white block (3, II) and may slip into the *Hole* block (3, III). Consequently, the supervisor cannot ensure that the truck meets constraint 1.

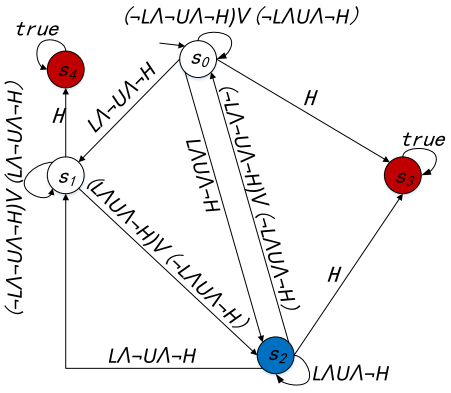
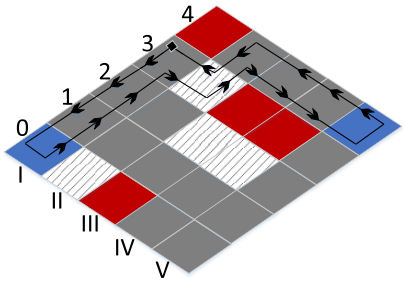
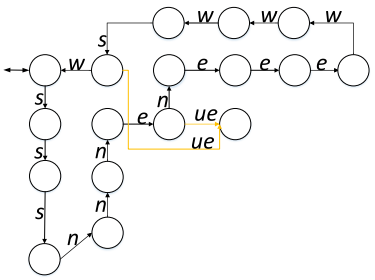


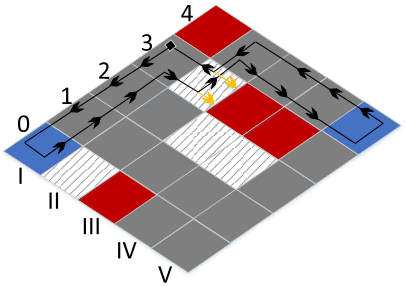
FIGURE 6. Deterministic Rabin automaton $\mathbf{R}_\varphi$ corresponding to φ.



(a) The trajectory induced by $f^*_{\mathbf{P}'}$.



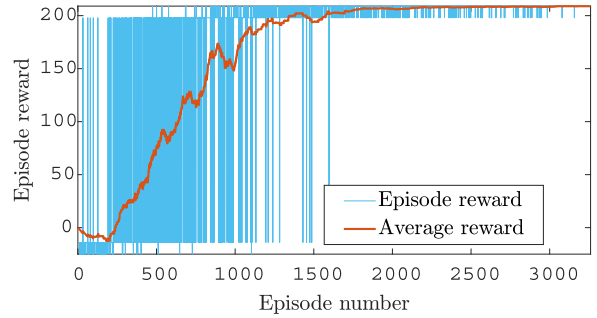
(b) The supervisor $\mathbf{S}'$.



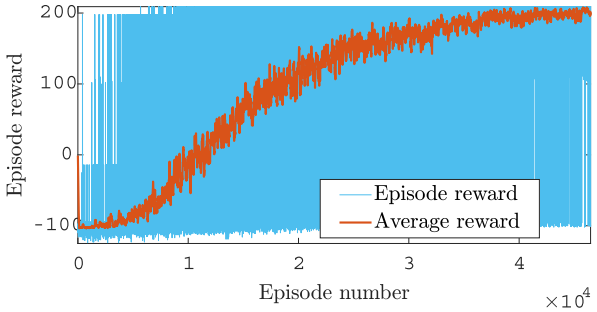
(c) The trajectory induced by $\mathbf{S}'$.

FIGURE 7. Trajectories and the supervisor.

Therefore, the RL method fails to compute a correct supervisor for the truck without recognizing the controllability of events. Next, we investigate the situation that considers the uncontrollable events.



(a) The proposed method.



(b) RL method without using SCT.

FIGURE 8. Plots of episode reward and average reward.

b: RL METHOD CONSIDERING UNCONTROLLABLE EVENTS

This section considers the uncertainties induced by uncontrollable events. The product Rabin DFA $\mathbf{P} = \mathbf{G} \times \mathcal{R}_\varphi$ is the same as it in the last method. Given the same values of parameters in the last method, we obtain an optimal controller $f_{\mathbf{P}}^*$ that satisfies the LTL formula φ by Algorithm 2. The trajectory induced by $f_{\mathbf{P}}^*$ is illustrated in Fig. 5(a). Then we obtain the supervisor $\mathbf{S}$, which is the same as the supervisor illustrated in Fig. 5(b). As analyzed in the proposed method of the case, the supervisor meets the control requirements.

Considering the controllability of events, the correct supervisor of the plant is obtained by both the proposed method and the RL method without using SCT. Fig. 8 shows the plots of changes of *EpisodeReward* and *AverageReward* introduced in Algorithm 2 through the proposed method and the RL method without using SCT. The proposed method requires 3257 episodes to converge the optimal controller, while the RL method without using SCT needs 46682 episodes. The oscillation in the figures comes from the uncertainties caused by both the ϵ-greedy algorithm and the uncontrollable events.

Table 2 displays the differences between the two methods in the number of states, learning episodes, and learning times before converging. The RL method without using SCT requires 102 states of the product MDP while the proposed method requires only 33 states. The implementations are realized by MATLAB with a PC with Intel(R) Core(TM) i5-8265U CPU @1.60GHz, 1.80GHz, and 8.00GB installed memory (RAM). The proposed method requires 7.2 seconds while the RL method without using SCT needs 90.8 seconds.

TABLE 2. Comparison of computing overhead.

Method	State	Episode	Time
RL	102	46682	90.8
SCT+RL	33	3257	7.2

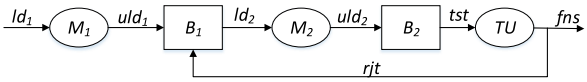


FIGURE 9. Configuration of transfer line.

In summary, the proposed method searches a smaller state space, obtains the converged values earlier, and requires less time than the RL method without using SCT.

B. CONTROL OF AN INDUSTRIAL TRANSFER LINE SYSTEM

The second example is an industrial transfer line consisting of two machines $\mathbf{M}_1$, $\mathbf{M}_2$ and a test unit **TU**, linked by buffers $\mathbf{B}_1$ and $\mathbf{B}_2$ in Fig. 9. The capacities of buffers $\mathbf{B}_1$ and $\mathbf{B}_2$ are, for example, 3 and 1, respectively. Machine $\mathbf{M}_1$ takes a part and is loaded. After the processing, $\mathbf{M}_1$ is unloaded and the finished part is moved to buffer $\mathbf{B}_1$. Machine $\mathbf{M}_2$ takes a part from buffer $\mathbf{B}_1$ and is loaded. After the processing, $\mathbf{M}_2$ is unloaded and the finished part is moved to buffer $\mathbf{B}_2$. The test unit **TU** takes a part from $\mathbf{B}_2$ for quality test. If the part is accepted by **TU**, it is released from the system; if rejected, it is returned to $\mathbf{B}_1$ for reprocessing by $\mathbf{M}_2$. Thus the system incorporates 'material feedback'. We assume that the returned part has a priority to be taken from $\mathbf{B}_1$. The control requirements of the system are:

1. Buffers $\mathbf{B}_1$ and $\mathbf{B}_2$ must be protected against underflow and overflow.

2. If a workpiece is rejected by the test unit, $\mathbf{B}_1$ is not allowed to receive any workpiece from $\mathbf{M}_1$ until the rejected workpiece is processed by $\mathbf{M}_2$.

The labeled DFA models of $\mathbf{M}_1, \mathbf{M}_2$, **TU**, $\mathbf{B}_1$ and $\mathbf{B}_2$ are displayed in Fig. 10, where $A_c = \{ld_1, ld_2, tst\}$ and $A_u = \{uld_1, uld_2, fns, rjt\}$. The atomic proposition set is $\Pi = \{pro_1, pro_2, RJT, und_1, over_1, und_2, over_2\}$. Events pro_1 and pro_2 respectively denote the processing of $\mathbf{M}_1$ and $\mathbf{M}_2$, and RJT means that **TU** rejects a workpiece. The labels und_1 and und_2 respectively denote the underflow of $\mathbf{B}_1$ and $\mathbf{B}_2$, and the labels $over_1$ and $over_2$ denote the overflow of $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively. The labels of nodes in white are empty.

The specification is described as an LTL formula

$$\phi = \Box\neg und_1 \wedge \Box\neg und_2 \wedge \Box\neg over_1 \wedge \Box\neg over_2 \wedge \Box(RJT \rightarrow \neg pro_1 U pro_2). \quad (9)$$

1) THE PROPOSED METHOD

The proposed method substitutes the LTL specification $\phi_1 = \Box\neg und_1 \wedge \Box\neg und_2 \wedge \Box\neg over_1 \wedge \Box\neg over_2$ in (9) by a DFA specification $\mathbf{E}$, which is the synchronization of **B1SPEC** and **B2SPEC** in Fig. 11.

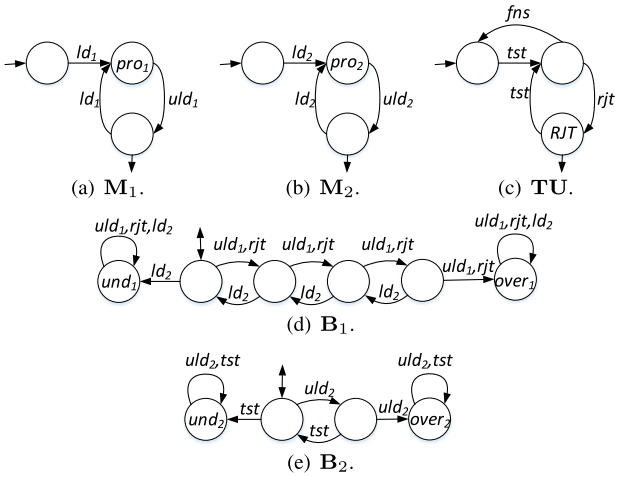


FIGURE 10. Component DES.

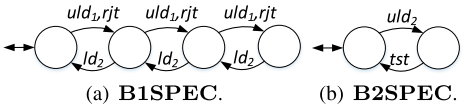


FIGURE 11. Buffer specifications.

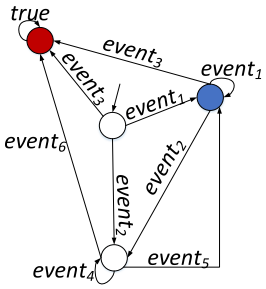


FIGURE 12. The DRA $\mathcal{R}_2$ corresponding to ϕ_2.

The plant $\mathbf{G}$ is the synchronization of $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{TU}$. First of all, we obtain a supervisor $\mathbf{S_1}$, with 28 states and 65 transitions that satisfies $\mathbf{E}$ for the plant $\mathbf{G}$ through SCT. We translate the remaining LTL formula $\phi_2 = \Box(RJT \rightarrow \neg pro_1 U pro_2)$ to a DRA $\mathbf{R}_2$ in Fig. 12, where

$$event_1 = (\neg RJT \wedge pro_1 \wedge pro_2) \vee (RJT \wedge \neg pro_1 \wedge pro_2) \vee (\neg RJT \wedge \neg pro_1 \wedge \neg pro_2) \vee (\neg RJT \wedge \neg pro_1 \wedge pro_2) \vee (RJT \wedge pro_1 \wedge pro_2) \vee (\neg RJT \wedge pro_1 \wedge \neg pro_2),$$

$event_2 = RJT \wedge \neg pro_1 \wedge \neg pro_2$, $event_3 = RJT \wedge pro_1 \wedge \neg pro_2$, $event_4 = (\neg RJT \wedge \neg pro_1 \wedge \neg pro_2) \vee (RJT \wedge \neg pro_1 \wedge \neg pro_2)$, $event_5 = pro_2$, and $event_6 = (RJT \wedge pro_1 \wedge \neg pro_2) \vee (\neg RJT \wedge pro_1 \wedge \neg pro_2)$.

We get the labeled supervisor $\mathbf{S}_1$ and obtain a Rabin product DFA $\mathbf{P} = \mathbf{S}_1 \times \mathcal{R}_2$ with 112 states. Applying Algorithm 2 to $\mathbf{P}$, we obtain an optimal controller for ϕ_2. The parameters used in this case are displayed in Table 3.

Then we construct the final supervisor $\mathbf{S}$ in Fig. 13. It is tested by Monte Carlo simulations that the supervisor indeed satisfies the specifications $\mathbf{B}_1$ and $\mathbf{B}_2$. As an illustration, during any trajectory of transitions allowed by $\mathbf{S}$, after the occurrence of event rjt, the event ld_1 to load the first machine is disabled until the second machine is loaded by event ld_2. Thus the LTL specification ϕ_2 is satisfied.

TABLE 3. Values of parameters.

Parameter	α	γ	ε	w_G	w_B	Δ	$Decay$	$MaxiStep$	$Length$
Value	0.9	0.96	0.98	5	-5	73	10^{-2}	50	30

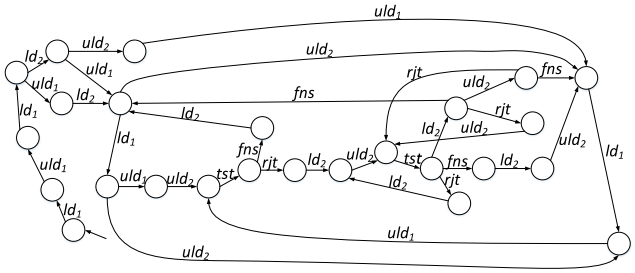


FIGURE 13. Supervisor S of transfer line.

TABLE 4. Comparison of computing overhead.

Method	State	Episode	Time
RL	864	3498	11.2
SCT+RL	112	228	0.7

2) RL METHOD WITHOUT SCT

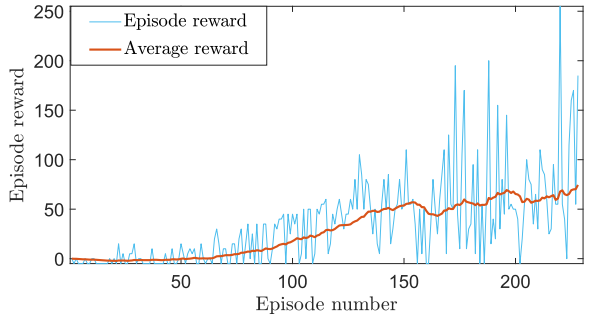
The plant model $\mathbf{G}$ is the synchronization of components in Fig. 10. The label of each state is the union of the labels of the states in modular components. The specification is expressed in (9) and is translated into a DRA $\mathcal{R}_\phi$ with 4 states and 128 transitions. We construct a product DFA $\mathbf{P}' = \mathbf{G} \times \mathcal{R}_\phi$ with 864 states. Applying Algorithm 2 to $\mathbf{P}'$, we obtain an optimal controller for ϕ, and then we extend the optimal controller to the corresponding supervisor $\mathbf{S}$ in Fig. 13. As analyzed in the last section, the supervisor satisfies the control requirements.

Table 4 displays the differences between the two methods in the number of states, learning episodes, and learning times before converging. The RL method without using SCT requires 864 states of the product DFA while the proposed method requires only 112 states. The implementations are realized by the same PC as the first case. The proposed method requires 0.7 seconds while the RL method without using SCT needs 11.2 seconds.

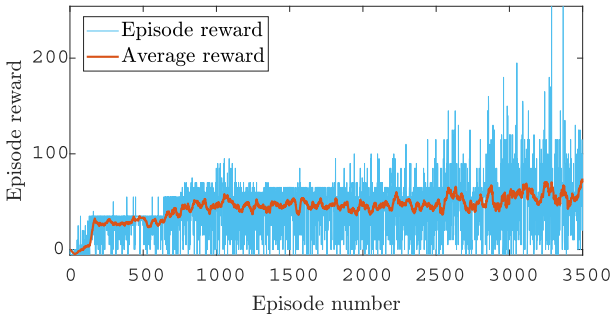
Fig. 14 shows the plots of changes of episode reward and average reward introduced in Algorithm 2 through the two methods. The proposed method requires 228 episodes, while the RL method without using SCT needs 3498 episodes.

3) DEEP REINFORCEMENT LEARNING FRAMEWORK

To improve the scalability of the proposed method, the Q-learning algorithm in Algorithm 2 using the explicit Q



(a) The proposed method.



(b) RL method without using SCT.

FIGURE 14. Plots of episode reward and average reward.

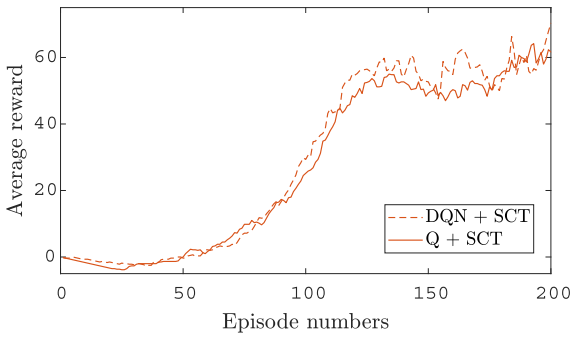


FIGURE 15. Plots of average rewards from the methods that use the DQN algorithm and the Q-learning algorithm.

table can be replaced by a deep Q-network (DQN) algorithm. The Q table for the transfer line example is approximated by a fully connected three-layer neural network, where the numbers of neurons of the input layer, the middle layer, and the output layer are 6, 8, and 8, respectively. The details of the DQN algorithm are elaborated in [46].

The input of the neural network is a vector of the state numbers of the plant components, the modular supervisors, and the DRA translated from the LTL specification, where the modular supervisors are obtained from the plant components and the individual specifications. In this case, the input is $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$, where x_1, x_2 and x_3 correspond to the state numbers of $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{TU}$, x_4 and x_5 represent the state numbers of two modular supervisors that fulfill $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively, and x_6 denotes the state number of $\mathcal{R}_2$. The reward function for DQN is the same as the one for the Q-learning algorithm. and the values of rewards are shown in Table 3. Other parameters for the DQN algorithm are the same as those in Table 3 in [46].

Fig. 15 displays the plots of the average rewards by the methods employing the Q-learning and the DQN algorithm, respectively. It shows that the learning efficiency is improved when the Q table is replaced by a neural network. The primary advantage of the DQN algorithm is the reduction of memory space. The Q table for this example has 112 states and 8 actions. The total number of elements is $112 \times 8 = 896$, but the total number of parameters for the DQN is only $6 \times 8 + 8 \times 8 = 112$. The optimal policy is modeled by the obtained neural network, and the final supervisor constructed is the same as the one shown in Fig. 13.

V. CONCLUSION AND FUTURE WORK

We investigate the controllability of events and propose a method that combines SCT and RL framework to compute a supervisor that allows all uncontrollable events in DESs. Moreover, the proposed method can be applied to a DRL framework to solve the control problems for a large-scale DES. In this paper, SCT is utilized to remove the obviously unacceptable states and hence saves learning time.

In the future, we plan to apply DRL methods to design a supervisory controller for a DES with multiple LTL specifications. The proposed method is employed to obtain multiple modular supervisors that satisfy the individual LTL specifications, and a high-level supervisor is approximated by a neural network to coordinate these modular supervisors such that the LTL specifications are satisfied, and the learning time is reduced since SCT eliminates a lot of obviously unacceptable states.

REFERENCES

[1] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Berlin, Germany: Springer, 2019.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. New York, NY, USA: Springer, 2008.

[3] T. Saravanakumar, S. M. Anthoni, and Q. Zhu, "Resilient extended dissipative control for Markovian jump systems with partially known transition probabilities under actuator saturation," *J. Franklin Inst.*, vol. 357, no. 10, pp. 6197–6227, Jul. 2020.

[4] P. Pouya and A. M. Madni, "Expandable-partially observable Markov decision-process framework for modeling and analysis of autonomous vehicle behavior," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3714–3725, Sep. 2021.

[5] B. P. Zeigler, "Discrete event system specification framework for self-improving healthcare service systems," *IEEE Syst. J.*, vol. 12, no. 1, pp. 196–207, Mar. 2018.

[6] S. Coogan, M. Arcak, and C. Belta, "Formal methods for control of traffic flow: Automated control synthesis from finite-state transition models," *IEEE Control Syst. Mag.*, vol. 37, no. 2, pp. 109–128, Apr. 2017.

[7] J. Tan, F. Liu, and Z. Dziong, "Active opacity of discrete-event systems," *Int. J. Control*, vol. 95, pp. 1–10, Jun. 2022.

[8] O. Himrane, A. Ourghanlian, and S. Amari, "Response time evaluation of industrial-scale distributed control systems by discrete event systems formalisms," *Int. J. Control*, vol. 95, no. 2, pp. 419–431, Feb. 2022.

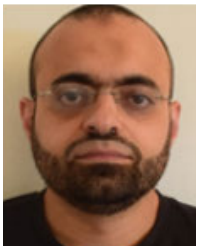
[9] J.-M. Yang and D.-E. Lee, "Robust corrective control against a class of actuator attacks in input/state asynchronous sequential machines," *J. Franklin Inst.*, vol. 358, no. 2, pp. 1403–1421, Jan. 2021.

[10] C. R. De Lima, S. G. Khan, S. H. Shah, and C. Lu, "Deliberative/reactive architecture of a multirobot patrol system based on supervisory control theory," *IEEE Sensors J.*, vol. 22, no. 11, pp. 11023–11033, Jun. 2022.
[11] L. Cheng, L. Feng, and Z. Li, "Model abstraction for discrete-event systems by binary linear programming with applications to manufacturing systems," *Sci. Prog.*, vol. 104, no. 3, 2021, Art. no. 00368504211030833.
[12] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.
[13] W. M. Wonham, K. Cai, and K. Rudie, "Supervisory control of discrete-event systems: A brief history," *Annu. Rev. Control*, vol. 45, pp. 250–256, 2018.
[14] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[15] J. G. Thistle and H. M. Lamouchi, "Effective control synthesis for partially observed discrete-event systems," *SIAM J. Control Optim.*, vol. 48, no. 3, pp. 1858–1887, Jan. 2009.
[16] B. Lacerda, "Supervision of discrete event systems based on temporal logic specifications," Ph.D. thesis, Instituto Superior Tecnico, Univ. Tecnica de Lisboa, Lisbon, Portugal, 2013.
[17] A. Sakakibara and T. Ushio, "On-line permissive supervisory control of discrete event systems for scLTL specifications," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 530–535, Jul. 2020.
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2015.
[19] S. Qiu, Z. Li, Z. Pang, Z. Li, and Y. Tao, "Multi-agent optimal control for central chiller plants using reinforcement learning and game theory," *Systems*, vol. 11, no. 3, p. 136, Mar. 2023.
[20] S. Hu, J. Gao, D. Zhong, R. Wu, and L. Liu, "Real-time scheduling of pumps in water distribution systems based on exploration-enhanced deep reinforcement learning," *Systems*, vol. 11, no. 2, p. 56, Jan. 2023.
[21] S. L. Smith, J. Tumová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Robot. Res.*, vol. 30, no. 14, pp. 1695–1708, Dec. 2011.
[22] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 2, pp. 858–865, Mar. 2021.
[23] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 1091–1096.
[24] M. Hiromoto and T. Ushio, "Learning an optimal control policy for a Markov decision process under linear temporal logic specifications," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Dec. 2015, pp. 548–555.
[25] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1244–1257, May 2014.
[26] S. Safra, "On the complexity of ω-automata," in *Proc. 29th Annu. Symp. Found. Comput. Sci.*, 1988, pp. 319–327.
[27] Z. Komarkova and J. Kretinsky, "Rabinizer 3: Safraless translation of LTL to small deterministic automata," in *Proc. Int. Symp. Automated Technol. Verification Anal.*, 2014, pp. 235–241.
[28] P. Schillinger, M. Burger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems*. Cham, Switzerland: Springer, 2018, pp. 253–267.
[29] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5033–5040.
[30] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *Proc. Int. Symp. Robot. Res.*, 2016, pp. 21–37.
[31] R. Dimitrova, B. Finkbeiner, and H. Torfah, "Approximate automata for omega-regular languages," in *Proc. 17th Int. Symp. Automated Technol. Verification Anal.*, 2019, pp. 334–349.
[32] J. Tumova and D. V. Dimarogonas, "Decomposition of multi-agent planning under distributed motion and task LTL specifications," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Dec. 2015, pp. 7448–7453.
[33] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 5338–5343.
[34] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021.
[35] A. Sakakibara and T. Ushio, "Decentralized supervision and coordination of concurrent discrete event systems under LTL constraints," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 7–12, 2018.
[36] E. Gradel, W. Thomas, and W. Thomas, *Automata, Logics, and Infinite Games*. Berlin, Germany: Springer, 2002.
[37] A. Sakakibara and T. Ushio, "Hierarchical control of concurrent discrete event systems with linear temporal logic specifications," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 10, no. 2, pp. 313–321, 2018.
[38] K. M. C. Zielinski, L. V. Hendges, J. B. Florindo, Y. K. Lopes, R. Ribeiro, M. Teixeira, and D. Casanova, "Flexible control of discrete event systems using environment simulation and reinforcement learning," *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107714.
[39] Y. Yuan, Z. L. Yu, Z. Gu, X. Deng, and Y. Li, "A novel multi-step reinforcement learning method for solving reward hacking," *Int. J. Speech Technol.*, vol. 49, no. 8, pp. 2874–2888, Aug. 2019.
[40] L. Feng and W. M. Wonham, "TCT: A computation tool for supervisory control synthesis," in *Proc. 8th Int. Workshop Discrete Event Syst.*, 2006, pp. 3–8.
[41] J. Brunner, B. Seidl, and S. Sickert, "A verified and compositional translation of LTL to deterministic rabin automata," in *Proc. 10th Int. Conf. Interact. Theorem Proving*, 2019, pp. 1–19.
[42] J. Esparza, J. Kretínský, and S. Sickert, "From LTL to deterministic automata," *Formal Methods Syst. Des.*, vol. 49, no. 3, pp. 219–271, Dec. 2016.
[43] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Apr. 2019, pp. 237–248.
[44] A. Giuseppi and A. Pietrabissa, "Bellman's principle of optimality and deep reinforcement learning for time-varying tasks," *Int. J. Control*, vol. 95, no. 9, pp. 2448–2459, 2021.
[45] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst. Mag.*, vol. 32, no. 6, pp. 76–105, Dec. 2012.
[46] J. Yang, K. Tan, L. Feng, and Z. Li, "A model-based deep reinforcement learning approach to the nonblocking coordination of modular supervisors of discrete event systems," *Inf. Sci.*, vol. 630, pp. 305–321, Jun. 2023.

JUNJUN YANG received the B.S. degree in mechanical design manufacturing and automation from Lanzhou Jiaotong University, Lanzhou, China, in 2016. He is currently pursuing the Ph.D. degree in mechatronic engineering with Xidian University, Xi'an, China. His current research interests include discrete event systems, systems control theory, reinforcement learning, and smart scheduling.

KAIGE TAN (Member, IEEE) received the B.Sc. degree in mechatronics from the Harbin Institute of Technology, Harbin, China, in 2018, and the M.Sc. degree in mechatronics from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Machine Design. His research interests include optimization, reinforcement learning, and optimal filtering.

LEI FENG (Member, IEEE) received the B.S. and M.S. degrees from Xi'an Jiaotong University, Xi'an, China, in 1998 and 2001, respectively, and the Ph.D. degree from the Systems Control Group, University of Toronto, Toronto, ON, Canada, in 2007. In 2012, he joined the Mechatronics and Embedded Control System Division, KTH Royal Institute of Technology, Stockholm, Sweden, where he is currently an Associate Professor. His main research interests include energy management control of mechatronic systems, autonomous driving, verification and control synthesis of cyber-physical systems, and supervisory control of discrete event systems.



AHMED M. EL-SHERBEENY received the master's and Ph.D. degrees in mechanical engineering from West Virginia University (WVU), in 2001 and 2006, respectively. He was a Graduate Teacher and a Research Assistant with WVU. He has been an Assistant Professor with the Industrial Engineering Department, King Saud University, since 2010. He was the former Head of the Alumni and Employment Unit, College of Engineering, King Saud University, from 2013 to 2018. His research interests include human factors engineering, manufacturing engineering, and engineering education.

ZHIWU LI (Fellow, IEEE) received the B.S. degree in mechanical engineering, the M.S. degree in automatic control, and the Ph.D. degree in manufacturing engineering from Xidian University, Xi'an, China, in 1989, 1992, and 1995, respectively. In 1992, he joined Xidian University. He is with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau. Over the past decade, he was a Visiting Professor with the University of Toronto, Technion (Israel Institute of Technology), Martin-Luther University, Conservatoire National des Arts et Métiers, Meliksah Universitesi, the University of Cagliari, the University of Alberta, and King Saud University. His current research interests include petri net theory and application, supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining.

He is a member of the Technical Committee on Discrete Event Systems of the IEEE Systems, Man, and Cybernetics Society. He was a member of the IFAC Technical Committee on Discrete Event and Hybrid Systems, from 2011 to 2014. He was a recipient of the Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation, Germany, and Research in Paris, France. He is the Founding Chair of the Xi'an Chapter of the IEEE Systems, Man, and Cybernetics Society. He serves as a Frequent Reviewer for more than 90 international journals, including *Automatica* and a number of the IEEE TRANSACTIONS and many international conferences. He was listed in Marquis Who's Who in the World, 27th Edition, in 2010.