Information Sciences

A Model-Based Deep Reinforcement Learning Approach to the Nonblocking Coordination of Modular Supervisors of Discrete Event Systems

--Manuscript Draft--

Manuscript Number:	INS-D-22-5595
Article Type:	Full length article
Keywords:	Discrete event system; supervisory control theory; local modular control; deep reinforcement learning
Corresponding Author:	Lei Feng, PhD KTH Royal Institute of Technology Department of Machine Design Stockholm, SWEDEN
First Author:	Junjun Yang
Order of Authors:	Junjun Yang
	Kaige Tan
	Lei Feng, PhD
	Zhiwu Li
Abstract:	Modular supervisory control may lead to conflicts among the modular supervisors for large-scale discrete event systems. The existing methods for ensuring nonblocking control of modular supervisors either exploit favorable structures in the system model to guarantee the nonblocking property of modular supervisors or employ hierarchical model abstraction methods for reducing the computational complexity of designing a nonblocking coordinator. The nonblocking modular control problem is in general NP-hard. This study integrates supervisory control theory and a model-based deep reinforcement learning method to synthesize a nonblocking coordinator for the modular supervisors. The deep reinforcement learning method significantly reduces the computational complexity by avoiding the computation of synchronization of multiple modular supervisors and the plant models. The supervisory control function is approximated by the deep neural network instead of a large-sized finite automaton. Furthermore, the proposed model-based deep reinforcement learning method is more efficient than the standard deep Q network algorithm.
Suggested Reviewers:	Michel Reniers Eindhoven University of Technology m.a.reniers@tue.nl Prof. Reniers is an expert on supervisory control theory of discrete-event systems.
	Martin Fabian Chalmers University of Technology fabian@chalmers.se Prof. Fabian is an expert on supervisory control of discrete-event systems. He is also interested in AI methods for DES.

Dear editors and reviewers

It is our great pleasure to submit the original research paper to *Information Sciences*. The main contribution is to integrate modular supervisory control methods and deep reinforcement learning (DRL) methods to synthesize a nonblocking coordinator for a group of individually computed modular supervisors. The DRL method significantly reduces the computational complexity by avoiding the computation of synchronization of multiple modular supervisors and their related plant models. The supervisory control function is approximated by a deep neural network instead of a large-sized finite automaton. Furthermore, the proposed DRL method is more efficient than the standard deep Q network (DQN) learning method.

We confirm that the ideas of the paper are originally developed by all authors and the paper has never been published nor under review elsewhere. All authors have checked the manuscript and agreed to the submission. We also acknowledge all organizations that funded the research.

We look forward to your review feedbacks. Thank you!

All authors

A Model-Based Deep Reinforcement Learning Approach to the Nonblocking Coordination of Modular Supervisors of Discrete Event Systems

Junjun Yang[a], Kaige Tan[b], Lei Feng[b,*], Zhiwu Li[a,c]

[a] *School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China*
[b] *Department of Machine Design, KTH Royal Institute of Technology, Stockholm 10044, Sweden*
[c] *Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau*

Abstract

Modular supervisory control may lead to conflicts among the modular supervisors for large-scale discrete event systems. The existing methods for ensuring nonblocking control of modular supervisors either exploit favorable structures in the system model to guarantee the nonblocking property of modular supervisors or employ hierarchical model abstraction methods for reducing the computational complexity of designing a nonblocking coordinator. The nonblocking modular control problem is in general NP-hard. This study integrates supervisory control theory and a model-based deep reinforcement learning method to synthesize a nonblocking coordinator for the modular supervisors. The deep reinforcement learning method significantly reduces the computational complexity by avoiding the computation of synchronization of multiple modular supervisors and the plant models. The supervisory control function is approximated by the deep neural network instead of a large-sized finite automaton. Furthermore, the proposed model-based deep reinforcement learning method is more efficient than the standard deep Q network algorithm.

Keywords: Discrete event system, supervisory control theory, local modular control, deep reinforcement learning.

1. Introduction

Discrete event systems (DESs) [49, 3] encompass a wide variety of human-made systems, including power systems [32], healthcare service systems [55], traffic systems [5], communication protocols [24], digital systems [44] and robotic systems [43]. Ramadge and Wonham [30] initiate the supervisory control theory (SCT) to compute a supervisor that prevents certain controllable events based on the state of a DES to guarantee that the controlled system satisfies all specifications, is nonblocking, and has maximal freedom of behavior. SCT [49, 30, 50, 25] develops many theories and algorithms to compute maximally permissive and nonblocking supervisors with acceptable computational overheads. The fundamental solution is to synthesize a monolithic supervisor to satisfy all control requirements, which is an NP-hard problem [15] for large-scale DESs consisting of multiple asynchronous plant components and specifications. To reduce the computational complexity, Ramadge and Wonham [51] propose a modular control framework for the DES with multiple control specifications. A modular supervisor is dedicated to a single specification, and the conjunction of these modular supervisors enforces the satisfaction of all specifications. Modular supervisory control theory reduces the complexity of control synthesis for large-scale DESs [47]. Queiroz and Cury [29] study modular control for DESs with multiple concurrent plant components. In their work, each specification is associated with a group of components that share events with it. To synthesize the supervisor for the specification, only these plant components with shared events are needed. The supervisor synthesized from the specification and the related plant components is called a *local modular supervisor*. The final system behavior under all local modular supervisors is the synchronous product of these modular supervisors and all plant components.

Since the local modular supervisors are designed for individual specifications without considering the influence of other modular supervisors of a DES, the conjunction of them may cause blocking. The straightforward solution to nonblocking modular control is to compute another nonblocking supervisor, namely the *coordinator* [47], for the synchronization of all modular supervisors and the DES plant. The computational complexity, however, is exponential with the number of modular supervisors and plant components [28]. Computing the nonblocking coordinator with affordable complexity remains a major research problem for SCT.

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

Traditional SCT applies two approaches to develop nonblocking modular supervisory control efficiently. The first exploits system structural properties that can guarantee the nonblocking property of local modular supervisors [8, 10, 16]. The second applies system decomposition and hierarchal model abstraction techniques to significantly reduce the computational complexity of synthesizing the coordinator [11, 12, 48, 17, 33, 34, 7, 20, 21]. For the first approach of exploiting favorable system structures, Feng and Wonham [8, 10] propose a control-flow-network method that models the interconnecting structure of local modular supervisors and plant components, which identifies network structures that can guarantee the nonblocking property of certain modular supervisors. Goorden et al. [16] identify three model properties that ensure the nonblocking property of modular supervisors.

For the second approach of using model abstraction methods, a network of local modular supervisors and the related plant components is organized as multiple connected subsystems. The methods to obtain favorable structures of subsystems are studied in [8, 16]. The coordinator of each subsystem can be synthesized efficiently because of the reduced size of the subsystem. To further find the coordinator among the subsystems, they are abstracted into smaller models, and the abstraction must keep the nonblocking property of the original subsystem [11]. The model abstraction method is typically based on the observer property [12, 48]. Hill and Tilbury [17] suggest an incremental hierarchical approach to produce a set of nonblocking modular supervisors for large-scale DESs. Schmidt et al. [33, 34] study the hierarchical control and apply it to concurrent DESs. Feng et al. [11, 7] propose a similar architecture that ensures maximally permissive control of local modular supervisors through model abstractions using the natural observer property. The hierarchical interface-based supervisory control approach [20, 21] decomposes a DES into high-level subsystems, low-level subsystems, and interfaces that restrict the communications among these subsystems. The interfaces hide unnecessary information such that the computational complexity is limited. Moreover, Seow [35] presents an organizational control approach to the supervisory control of a class of hierarchical DESs using dynamic programming recursion.

In addition, distributed control [46] and state-tree-based control [42] for DESs are also investigated to reduce the computational complexity of synthesizing a supervisor. All these methods rely on favorable structures in the system model. In general applications, the worst complexity is still exponential with the total number of plant components and specifications.

Reinforcement learning (RL) [39, 22, 23] is an emerging method for the decision and control of large-scale DESs. The method learns the control decisions through a large number of simulations of the studied system and hence avoids computing the synchronous product of multiple local modular supervisors and the plant. State explosion in conventional coordinator design approaches is avoided. Yamasaki and Ushio [41, 52, 18, 19] propose a decentralised supervisory control framework based on RL to synthesize a supervisor for a DES. The reward function is related to both the disabled and the enabled events during the simulations. If an uncontrollable event is disabled, the reward is assigned to negative infinity, which guarantees the permission of uncontrollable events. Moreover, the nonblocking property of modular supervisors is ensured by maximizing the accumulative rewards. Zielinski et al. [56] propose a method that combines SCT and RL for the flexible control of an industrial system. They obtain an optimal supervisor for a DES by State-Action-Reward-State-Action (SARSA) and N-step SARSA [54] based on a known supervisor that satisfies the control specifications through SCT. In [56], the uncontrollable events are addressed by the proposed controllability ϵ-greedy policy, and the nonblocking property is ensured by the supervisor. However, the algorithms used in these researches are only suitable for the problems with low complexity since they require the multi-dimensional Q-table to represent the state-action function. The size of the Q-table is the same as the size of the reachable states of a DES.

This work integrates SCT and a model-based deep reinforcement learning (DRL) algorithm to synthesize nonblocking modular supervisors for large-scale DESs. SCT is employed to synthesize the local modular supervisors related to the individual specifications and their local plant components. A model-based DRL algorithm is proposed to find a coordinator for these local modular supervisors. The proposed learning algorithm selects an event subset that includes all feasible uncontrollable events at a state to guarantee that no uncontrollable event is disabled. Moreover, a reward of the selection at the state is associated with the rewards of transitions induced by all events in the selected event subset. The nonblocking property is also satisfied by maximizing the accumulated rewards of the allowed sequences of state transitions.

A supervisor determines an event subset at a reachable state in a DES. However, due to the Markov property, an RL algorithm only searches a single control action at a system state and updates the value function according to the selected action. Therefore, we need to code every event subset as a distinct control action. The number of control actions is then exponential with the number of events of a DES. If this number is large, then the number of control actions to be explored by RL algorithms become intractable. To limit the computational complexity of RL, our method allows the user to set an upper bound on the size of the event subset permitted by the coordinator. Hence, the supervisor obtained by the proposed learning algorithm may not be maximally permissive. In summary, the paper has the following two main contributions.

- A model-based DRL is designed to suit the supervisory control theory of DESs. Compared with the standard

DQN algorithm, the proposed DRL algorithm generates a reward by considering the state transitions of all events in a selected event subset instead of only one sample of one possible state transition. Thus, the proposed DRL algorithm has higher training efficiency than DQN.

- The computational complexity of the proposed method is significantly less than the conventional method since the synchronous product of multiple local modular supervisors and plant models is avoided. The global state of the controlled system is generated on-the-fly during the learning process, and the supervisory control function is represented by the deep neural network instead of a large-scale automaton or a state-based lookup table.

The proposed method is verified by two case studies: a manufacturing system with five automated guided vehicles (AGVs) and a railway network with shared tracks. The results show that the proposed algorithm obtains the nonblocking coordinators for the cases and is more efficient than the standard DQN algorithm.

The structure of the paper is organized as follows. Section 2 reviews some necessary preliminaries. Section 3 displays the scheme of the proposed approach. Section 4 describes the details of the integration of SCT and DRL. Section 5 provides two cases to show that the proposed approach successfully and efficiently obtains the correct coordinators. We conclude the paper in Section 6.

2. Preliminaries

Definition 1. A deterministic finite automaton (DFA) is defined as $\mathbf{G} = \langle Y, \Sigma, \eta, y_0, Y_m \rangle$, where

- Y is a finite nonempty state set;
- Σ is a finite nonempty event set, called the alphabet of transition labels;
- $\eta : Y \times \Sigma \to Y$ is the partial transition function;
- $y_0 \in Y$ is the initial state;
- $Y_m \subseteq Y$ is the set of marker states.

Given $y \in Y$ and $\sigma \in \Sigma$, if $\eta(y, \sigma)$ is defined, write $\eta(y, \sigma)!$. For convenience, η is extended to the partial function: $Y \times \Sigma^* \to Y$, defined as $\eta(y, \varepsilon) = y$ and $\eta(y, s\sigma) = \eta(\eta(y, s), \sigma)$ for $s \in \Sigma^*$ and $\sigma \in \Sigma$. Accordingly, given $s \in \Sigma^*$, write $\eta(y, s)!$ if $\eta(y, s)$ is defined. All events defined at state y is denoted by the set $\mathbf{Enb}(y) = \{\sigma \in \Sigma \mid \eta(y, \sigma)!\}$. If $\mathbf{Enb}(y) = \emptyset$, then y is a deadlock state. The generated language and the marked language of $\mathbf{G}$ are $\mathcal{L}(\mathbf{G}) = \{s \in \Sigma^* \mid \eta(y_0, s)!\}$ and $\mathcal{L}_m(\mathbf{G}) = \{s \in \Sigma^* \mid \eta(y_0, s) \in Y_m\}$, respectively. A state $y \in Y$ is *reachable* if there exists an event sequence $s \in \Sigma^*$ such that $\eta(y_0, s)!$ and $y = \eta(y_0, s)$; the state y is *coreachable* if there exists $s \in \Sigma^*$ such that $\eta(y, s) \in Y_m$; $\mathbf{G}$ is *nonblocking* if every reachable state is coreachable.

Let $\Sigma_s \subseteq \Sigma$ be a set of special events. For example, an event symbolizing the completion of a task may be a special event. For any event $\sigma \in \Sigma$, an event-reward function $r_e : \Sigma \to \mathbb{R}$ is defined as

$$r_e(\sigma) = \begin{cases} w_1 & \text{if } \sigma \in \Sigma_s \\ w_2 & \text{otherwise,} \end{cases}$$

where w_1 and w_2 are two positive real numbers satisfying $w_2 < w_1$. For a state $y \in Y$, a state-reward function $r_s : Y \to \mathbb{R}$ is defined as

$$r_s(y) = \begin{cases} w_3 & \text{if } y \in Y_m \\ w_4 & \text{if } y \notin Y_m \text{ and } \mathbf{Enb}(y) = \emptyset \\ 0 & \text{otherwise,} \end{cases}$$

where w_3 is a positive number and w_4 a negative number.

2.1. Supervisory control theory

Supervisory control theory is a branch of systems control theory studying the logical behavior of a DES [30]. A plant is modeled by a DFA $\mathbf{G}$ with an event set Σ that consists of two disjoint subsets Σ_c and Σ_u, where Σ_c is the set of controllable events and Σ_u is that of uncontrollable events. In SCT of DESs, the specification that a plant must satisfy is modeled by a DFA, denoted by $\mathbf{E}$, which usually 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 $V : \mathcal{L}(\mathbf{G}) \rightarrow \Gamma$, where $\Gamma = \{\gamma \in 2^{\Sigma} \mid \Sigma_u \subseteq \gamma \subseteq \Sigma\}$ is a set of allowed event subsets. An event subset that contains the uncontrollable event set is called a *control pattern*. The definition implies that the uncontrollable events must always be allowed to happen whenever they are feasible. When the plant reaches a state via a string $s \in \mathcal{L}(\mathbf{G})$, the supervisory control function allows the events in the set $V(s)$ to occur, and other events must be prevented. 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 Σ_u. If it exists, SCT synthesizes a 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., $\mathcal{L}_m(\mathbf{S}) \subseteq \mathcal{L}_m(\mathbf{S}^*)$. The supervisor $\mathbf{S}$ contains the state transition relationship of both the plant model and the specification, and is usually large in state size. A reduced supervisor [36, 37, 38] **RS** removes the state transition relationship already contained in the plant model and in general has much smaller state size than the original supervisor $\mathbf{S}$, and hence more clearly exhibits the essential control logic.

Many computational tools such as TCT [9], Supremica [26], IDES [31] and CIF [2] are available for synthesizing maximally permissive nonblocking supervisor for a given pair of plant DFA $\mathbf{G}$ and the specification DFA $\mathbf{E}$. This study applies TCT.

2.2. Local modular control

Let $\mathbf{G}_i$ ($i \in \{1, \ldots, A\}$) and $\mathbf{E}_j$ ($j \in \{1, \ldots, B\}$) be the sets of DFAs that describe the concurrent plant components and specifications defined over Σ_i and Σ'_j, respectively, where A and B are two integers greater than or equal to one. A set of plant components that share an event $\sigma \in \Sigma_i$ is defined as $I_\sigma = \{i \in \{1, \ldots, A\} \mid \sigma \in \Sigma_i\}$; the set of components that share events with $\mathbf{E}_j$ is $Share(\mathbf{E}_j) = \cup_{\sigma \in \Sigma'_j} I_\sigma$. The definition of the synchronous product of two DFAs is reviewed prior to the introduction of the local modular SCT principle. Note that Taş et al. [40] propose a GPU based approach to compute synchronization of automata with billion-scale. This study uses the original definition of the synchronization.

Definition 2. Given two DFAs $\mathbf{G}_1 = \langle Y_1, \Sigma_1, \eta_1, y_{10}, Y_{1m} \rangle$ and $\mathbf{G}_2 = \langle Y_2, \Sigma_2, \eta_2, y_{20}, Y_{2m} \rangle$, their synchronous product is defined as $\mathbf{G}_1 \| \mathbf{G}_2 = \langle Y, \Sigma, \eta, y, Y_m \rangle$, where

- $Y = Y_1 \times Y_2$ is the state set;
- $\Sigma = \Sigma_1 \cup \Sigma_2$ is the set of events;
- $\eta : Y \times \Sigma \rightarrow Y$ is the partial transition function: for $y = (y_1, y_2) \in Y$ and $\sigma \in \Sigma$,

$$\eta(y, \sigma) = \begin{cases} (y'_1, y_2) & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } y'_1 = \eta_1(y_1, \sigma) \\ (y_1, y'_2) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } y'_2 = \eta_2(y_2, \sigma) \\ (y'_1, y'_2) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2, y'_1 = \eta_1(y_1, \sigma) \text{ and } y'_2 = \eta_2(y_2, \sigma) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- $y_0 = (y_{10}, y_{20})$ is the initial state;
- $Y_m = Y_{1m} \times Y_{2m}$ is the set of marker states.

The local plant related to the specification $\mathbf{E}_j$ is defined as $\mathbf{H}_j = \|_{i \in Share(\mathbf{E}_j)} \mathbf{G}_i$, and the corresponding modular supervisor $\mathbf{S}_j$ that satisfies the individual specification $\mathbf{E}_j$ is synthesized from $\mathbf{H}_j$ and $\mathbf{E}_j$ through SCT. The synchronous product of all these local modular supervisors and the plant model describes the final behavior of the DES. As stated in Section 1, the controlled system may be blocking. A coordinator is usually required to ensure the nonblocking property [49, 27].

3. Sketch of the overall approach

Consider a DES with A plant components and B specifications, where both the components and the specifications can be expressed by DFAs, our objective is to design nonblocking modular supervisors to satisfy the specifications. As illustrated in Fig. 1, our approach has the following four steps.

- Compute B nonblocking modular reduced supervisors $\mathbf{S}_1, \ldots, \mathbf{S}_B$ from the plant components $\mathbf{G}_1, \ldots, \mathbf{G}_A$ and the individual specifications $\mathbf{E}_1, \ldots, \mathbf{E}_B$ through local modular control (LMC).

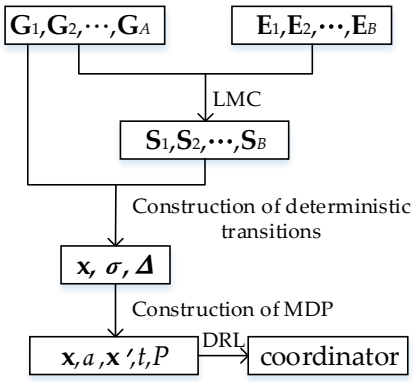


Figure 1: Scheme of the proposed approach.

- Represent the global state of the synchronous product of all plant components and modular supervisors as a vector of dimensionality $A + B$. Each element in the state vector represents a state in the plant components or the modular supervisors. An algorithm is developed to generate the global state on-the-fly. After executing an event σ at a global state $\mathbf{x}$, the next state is unique if the transition is defined. If no events is available at a global state and the global state is not marked, then the global state is a deadlock, and the system blocks, which indicates that a conflict occurs among the modular supervisors.
- Define a function to code a control pattern that is an event subset, as a discrete control action to facilitate the DRL algorithm. Then we obtain a Markov decision process (MDP) whose state set is identical to the state set of the global system. The action set is defined by the coding function and the set of all control patterns. The state-transition probability of the MDP is assumed to be an even distribution of all events allowed by the control pattern.
- Employ a model-based DRL algorithm to synthesize a coordinator for the modular supervisors such that the specifications are satisfied, and there is no conflict among the modular supervisors.

4. The integration of SCT and DRL

4.1. Construction of deterministic global state transitions

Let $\mathbf{G}_i$ $(i = 1, \ldots, A)$ be the DFA model of the i-th plant component and $\mathbf{E}_j$ $(j = 1, \ldots, B)$ be the DFA that models the j-th specification. The local plant that shares events with $\mathbf{E}_j$ is $\mathbf{H}_j = \|_{i \in Share(\mathbf{E}_j)} \mathbf{G}_i = \langle Z_j, \Pi_j, \xi_j, z_{j0}, z_{jm} \rangle$, and the nonblocking local modular supervisor $\mathbf{S}_j = \langle Q_j, \Pi_j, \delta_j, q_{j0}, Q_{jm} \rangle$ is synthesized by SCT from $\mathbf{H}_j$ and $\mathbf{E}_j$. The supervisor $\mathbf{S}_j$ may also be reduced by the method [38], but the proposed procedure applies equally regardless of whether it is reduced. We extend the event set Π_j of each local modular supervisor to the complete event set $\Sigma = \cup_{i=1}^{A} \Sigma_i$ of the system as $\delta'_j : Q_j \times \Sigma \to Q_j$. Given a state $q \in Q_j$ and an event $\sigma \in \Sigma$, define

$$\delta'_j(q, \sigma) = \begin{cases} \delta_j(q, \sigma) & \text{if } \sigma \in \Pi_j \text{ and } \delta_j(q, \sigma)! \\ q & \text{if } \sigma \notin \Pi_j \\ \text{undefined} & \text{otherwise.} \end{cases}$$

As a result, the system's behavior satisfies the specification $\mathbf{E}_j$ $(j = 1, \ldots, B)$ with the restriction of $\mathbf{S}_j$. Nonetheless, each local modular supervisor $\mathbf{S}_j$ is obtained without the knowledge of other modular supervisors, which may cause a system blocking. To solve this problem, we define a global state from the modular supervisors and the plant components as follows.

Definition 3. Given the models of A components $\mathbf{G}_i = \langle Y_i, \Sigma_i, \eta_i, y_{i0}, Y_{im}, \Sigma_{i,s} \rangle$ and B local modular supervisors $\mathbf{S}_j = \langle Q_j, \Sigma, \delta'_j, q_{j0}, Q_{jm} \rangle$, a global state is defined as a vector $\mathbf{x} = (y_1, y_2, \ldots, y_A, q_1, q_2, \ldots, q_B)$, where $y_i \in Y_i$ and $q_j \in Q_j$.

We call $\mathbf{x} = (y_1, y_2, \ldots, y_A, q_1, q_2, \ldots, q_B)$ a marker state, denoted by $\mathbf{x} \in \mathbf{X}_m$, if $(\forall i \in \{1, \ldots, A\}, \forall j \in \{1, \ldots, B\}) y_i \in Y_{im} \wedge q_j \in Q_{jm}$. The DFA models of the plant components and the local modular supervisors are at the initial states at

the beginning; thus the global state is initialized as

$$\mathbf{x}_0 = (y_{10}, y_{20}, \ldots, y_{A0}, q_{10}, q_{20}, \ldots, q_{B0}),$$

where $y_{i0} \in Y_i$ and $q_{j0} \in Q_j$.

A global state $\mathbf{x} = (y_1, y_2, \ldots, y_A, q_1, q_2, \ldots, q_B)$ consists of the states of the plant components and the states of the local modular supervisors for the individual specifications. Whether an event $\sigma \in \Sigma$ is allowed at $\mathbf{x}$ is decided by two aspects. One is that the event must be defined at the current state in the corresponding plant components, and the other is the permission of these local modular supervisors. Based on the two constraints, we define the set of enabled events at a global state as follows.

Definition 4. An event $\sigma \in \Sigma$ is said to be enabled at a global state $\mathbf{x}$, written as $\Delta(\mathbf{x}, \sigma)!$, if

$$(\forall i \in I_\sigma)\eta_i(y_i, \sigma)! \wedge (\forall j \in \{1, \ldots, B\})\delta'_j(q_j, \sigma)!,$$

where I_σ is the set of plant components that share the event σ, as defined in Section 2.2. The enabled event set at $\mathbf{x}$ is then defined as $\mathbf{Enb}(\mathbf{x}) = \{\sigma \in \Sigma \mid \Delta(\mathbf{x}, \sigma)!\}$. Accordingly, $\mathbf{x}$ is a deadlock global state that describes a conflict among the modular supervisors if $\mathbf{Enb}(\mathbf{x}) = \emptyset$ and $\mathbf{x}$ is not a marker global state.

Given a global state $\mathbf{x} = (y_1, y_2, \ldots, y_A, q_1, q_2, \ldots, q_B)$ and an event $\sigma \in \Sigma$, the next state $\mathbf{x}'$ is computed as

$$\Delta(\mathbf{x}, \sigma) = \begin{cases} (y'_1, y'_2, \ldots, y'_A, q'_1, q'_2, \ldots, q'_B) & \text{if } \sigma \in \mathbf{Enb}(\mathbf{x}) \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $(\forall i \in I_\sigma)y'_i = \eta_i(y_i, \sigma) \wedge (\forall i \notin I_\sigma)y'_i = y_i \wedge (\forall j \in \{1, \ldots, B\})q'_j = \delta'_j(q_j, \sigma)$, and $\Delta : \mathbf{X} \times \Sigma \to \mathbf{X}$ is the state transition function, with the set $\mathbf{X}$ that contains all global states. Note that our method does not require the enumeration of all states in $\mathbf{X}$. The global states are generated on-the-fly during the learning process.

4.2. Conversion between event subsets and actions for RL

In SCT, a supervisor assigns an event subset with all feasible uncontrollable events to a state, while an action is applied to a state in RL simulations. This section describes the conversion between the event subsets and the actions.

We first rank all events in the controllable event set Σ_c from 0 to $|\Sigma_c| - 1$ by a function $\iota : \Sigma_c \to \{0, \ldots, |\Sigma_c| - 1\}$, and the index of an event $\sigma \in \Sigma_c$ is $\iota(\sigma)$. Then we define a conversion function $\Omega : 2^{\Sigma_c} \to \mathbb{N}$ for a DES with the controllable event set Σ_c. Given an event subset $\gamma \subseteq \Sigma_c$, define

$$\Omega(\gamma) = \begin{cases} \sum\limits_{\sigma \in \gamma} 2^{\iota(\sigma)} & \text{if } \gamma \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

For instance, if a ranked controllable event set is $\Sigma_c = \{\sigma_0, \sigma_1\}$, then $\Omega(\{\sigma_1\}) = 2^1 = 2$ and $\Omega(\{\sigma_0, \sigma_1\}) = 2^0 + 2^1 = 3$. Therefore, an event subset $\gamma \subseteq \Sigma_c$ corresponds to an unique integer $a = \Omega(\gamma)$, where a is called the action of γ. Meanwhile, the event set γ can be determined as $\gamma = \Omega^{-1}(a)$. The set that collects the actions corresponding to all event subsets of Σ_c is defined as $\mathcal{A} = \{a = \Omega(\gamma) \mid \gamma \in 2^{\Sigma_c}\}$.

A state-action-event subset function $\Psi : \mathbf{X} \times \mathcal{A} \to 2^\Sigma$ is defined to assign a global state-action pair to a control pattern enabled at the state. Given a global state $\mathbf{x} \in \mathbf{X}$ and an action $a \in \mathcal{A}$, define

$$\Psi(\mathbf{x}, a) = (\Omega^{-1}(a) \cup \Sigma_u) \cap \mathbf{Enb}(\mathbf{x}), \tag{1}$$

where Σ_u is the uncontrollable event set of a DES, $\mathbf{Enb}(\mathbf{x})$ is used to remove the undefined events at $\mathbf{x}$, and $(\Omega^{-1}(a) \cup \Sigma_u) \in \Gamma$ is the control pattern corresponding to the control action a. In addition, we set an upper bound on the size of the event subset to restrict the cardinality of the action set, and hence the search space of DRL is reduced. Let $K \in \mathbb{R}$ be the upper bound. We first select the set of the controllable event subsets

$$\Sigma^K = \{\gamma \in 2^{\Sigma_c} \mid |\gamma| \leq K\}, \tag{2}$$

where $|\gamma|$ denotes the cardinality of the set γ. Second, we compute the set of actions corresponding to these event subsets

$$\mathcal{A}^K = \{a = \Omega(\gamma) \mid \gamma \in \Sigma^K\}. \tag{3}$$

4.3. Model-based deep reinforcement learning

A *run* is an ordered and infinite sequence of global transitions $\mathbf{r} = (\mathbf{x}_0, \sigma_0, \mathbf{x}_1), (\mathbf{x}_1, \sigma_1, \mathbf{x}_2), (\mathbf{x}_2, \sigma_2, \mathbf{x}_3)\ldots$, where $\mathbf{x}_{k+1} = \Delta(\mathbf{x}_k, \sigma_k)$. A deterministic static control policy is a function $f : \mathbf{X} \to \mathcal{A}^K$ that maps a global state to an action. After applying a policy f to a state $\mathbf{x}$, we obtain an action $a = f(\mathbf{x})$ and the corresponding event subset $\Psi(\mathbf{x}, a)$. To apply DRL, the event occurrence probabilities of all events in $\Psi(\mathbf{x}, a)$ should be known. These probability values, however, are not available in the DFA models of the plant and the local modular supervisors. To design the logical supervisor of the given DFA model, we assume that all events in $\Psi(\mathbf{x}, a)$ have the identical occurrence probability, i.e.,

$$P(\sigma \mid \mathbf{x}, a) = \begin{cases} \frac{1}{|\Psi(\mathbf{x},a)|} & \text{if } \sigma \in \Psi(\mathbf{x}, a) \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Then we define the reward function that reflects the control objectives and the nonblocking property. The one-step reward for a global transition $(\mathbf{x}, \sigma, \mathbf{x}')$ is defined as

$$t(\mathbf{x}, \sigma, \mathbf{x}') = r_e(\sigma) + r_s(\mathbf{x}'),$$

where r_e and r_s are defined in Definition 1. Given the initial global state $\mathbf{x}_0$ and a control function f, the discounted total reward for a run $\mathbf{r}$ induced by f is defined as

$$R(\mathbf{r}) = \sum_{k=0}^{\infty} \lambda^k t_k,$$

where $0 < \lambda \leq 1$ is the discount factor for the future reward. At $k \geq 0$, the corresponding control action is $a_k = f(x_k)$. If $\Psi(x_k, a_k) \neq \emptyset$, then $t_k = t(x_k, \sigma_k, x_{k+1})$, where $\sigma_k \in \Psi(x_k, a_k)$ and $x_{k+1} = \Delta(x_k, \sigma_k)$. If $\Psi(x_k, a_k) = \emptyset$, then $t_k = w_4$ (< 0) and for all $k' > k, t_{k'} = 0$. In other words, the run is terminated at step k.

By maximizing the expected return, we obtain a policy under which every reachable global state is coreachable, and the nonblocking coordinator of the obtained modular supervisors is obtained. The expectation of the discounted rewards of all infinite runs induced by f starting from a global state $\mathbf{x}$ is defined as a value function

$$V_f(\mathbf{x}) = \mathbb{E}_f\{\sum_{k=0}^{\infty} \lambda^k t_k \mid \mathbf{x}_0 = \mathbf{x}, \sigma_k \in \Psi(\mathbf{x}, a)\}, \tag{5}$$

where $\mathbb{E}_f\{\cdot\}$ is the expected value and $a = f(\mathbf{x})$. According to the Bellman equation [39] and the event occurrence probability in (4), (5) is written as

$$\begin{aligned} V_f(\mathbf{x}) &= \sum_{\sigma \in \Psi(\mathbf{x},a)} P(\sigma|\mathbf{x}, a)[t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda V_f(\Delta(\mathbf{x}, \sigma))] \\ &= \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x},a)} [t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda V_f(\Delta(\mathbf{x}, \sigma))]. \end{aligned}$$

The Bellman optimality equation is written as

$$V^*(\mathbf{x}) = \max_f \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x},a)} [t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda V^*(\Delta(\mathbf{x}, \sigma))].$$

The optimal function is

$$f^* = \arg\max_f \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x},a)} [t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda V^*(\Delta(\mathbf{x}, \sigma))],$$

and the final control pattern at $\mathbf{x}$ is $\Psi(\mathbf{x}, f^*(\mathbf{x}))$. To find the optimal control action with RL, the Q function [45] is more convenient. A global state-action value function $Q_f : \mathbf{X} \times \mathcal{A}^K \to \mathbb{R}$ is defined as the expected return starting from a

global state $\mathbf{x}$ and an action a:

$$\begin{aligned} Q_f(\mathbf{x}, a) &= \mathbb{E}_f\{\sum_{k=0}^{\infty} \lambda^k t_k \mid \mathbf{x}_0 = \mathbf{x}, a_0 = a, (k > 0) a_k = f(\mathbf{x}_k)\} \\ &= \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x}, a)} [t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda Q_f(\Delta(\mathbf{x}, \sigma), a')], \end{aligned} \tag{6}$$

where $a' = f(\Delta(\mathbf{x}, \sigma))$ is an action determined by f at $\Delta(\mathbf{x}, \sigma)$. If the optimal function f^* is used, then the Q function in (6) becomes the optimal Q function:

$$Q^*(\mathbf{x}, a) = \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x}, a)} [t(\mathbf{x}, \sigma, \Delta(\mathbf{x}, \sigma)) + \lambda \max_{a'} Q^*(\Delta(\mathbf{x}, \sigma), a')]. \tag{7}$$

This study employs the deep neural network [1] to approximate the Q function. We use a neural network parameterized by $\boldsymbol{\theta}$, denoted by $Q(\mathbf{x}, a; \boldsymbol{\theta})$, to approximate the optimal function $Q^*(\mathbf{x}, a)$ in (7). The number of input nodes of the neural network is equivalent to the dimension of the global transition system, i.e., $A + B$. The number of the output nodes is $|\mathcal{A}^K|$. Simultaneously, another neural network $\hat{Q}(\mathbf{x}, a; \boldsymbol{\phi})$ with the same network structure but different parameters is utilized as the target network. At the i-th iteration, the target value of $(\mathbf{x}_l, a_l)$ is predicted by

$$target_l = \frac{1}{\mid \Psi(\mathbf{x}_l, a_l) \mid} \sum_{\sigma \in \Psi(\mathbf{x}_l, a_l)} t(\mathbf{x}_l, \sigma, \Delta(\mathbf{x}_l, \sigma)) + \lambda \max_{a'} \hat{Q}(\Delta(\mathbf{x}_l, \sigma), a'; \boldsymbol{\phi}_{i-1}), \tag{8}$$

where l is the index of the global state and the corresponding action, and $\boldsymbol{\phi}_{i-1}$ are the parameters of the target network $\hat{Q}$ at the previous iteration. The loss function is

$$L(\boldsymbol{\theta}_i) = \mathbb{E}\{[target_l - Q(\mathbf{x}_l, a_l; \boldsymbol{\theta}_i)]^2\},$$

where $\boldsymbol{\theta}_i$ are the updated parameters of the estimate network Q at the i-th iteration to minimize the loss function.

Note that this study estimates $target_l$ by considering the influence of transitions induced by all events in $\Psi(\mathbf{x}_l, a_l)$ based on the global transition function Δ defined in Section 4.1. If an event $\sigma \in \Psi(\mathbf{x}_l, a_l)$ is executed at the global state $\mathbf{x}_l$, then the next state is $\mathbf{x}_{l+1} = \Delta(\mathbf{x}_l, \sigma)$. If $\mathbf{x}_{l+1}$ is deadlock, i.e., $\mathbf{Enb}(\mathbf{x}_{l+1}) = \emptyset$, then $\max_{a'} \hat{Q}(\Delta(\mathbf{x}_l, \sigma), a'; \boldsymbol{\phi}_{i-1})$ in (8) is set to 0. Compared with the proposed method, the standard DQN only considers one sample of execution of an event in $\Psi(\mathbf{x}_l, a_l)$ and computes the target value as $target_l = t(\mathbf{x}_l, \sigma, \Delta(\mathbf{x}_l, \sigma)) + \lambda \max_{a'} \hat{Q}(\Delta(\mathbf{x}_l, \sigma), a'; \boldsymbol{\phi}_{i-1})$, where σ is an event randomly selected from $\Psi(\mathbf{x}_l, a_l)$.

Algorithm 1 is the proposed model-based DRL algorithm. Each episode starts with the initial global state $\mathbf{x}_0$. For the l-th step of an episode, we observe the current global state $\mathbf{x}_l$ and select an action a_l from the K-bounded action set $\mathcal{A}^K$ by the ϵ-greedy algorithm, which is used to coordinate the balance between exploration and exploitation and is realized in Lines 6–11. The probability threshold is $\epsilon \in (0, 1)$, which varies as

$$\epsilon := \begin{cases} \epsilon(1 - \epsilon_{decay}) & \text{if } \epsilon > \epsilon_{min} \\ \epsilon & \text{otherwise} \end{cases} \tag{9}$$

after each step, where ϵ_{decay} is the decay rate and ϵ_{min} is the minimal value of ϵ.

The action a_l determines an event subset $\Psi(\mathbf{x}_l, a_l)$, which is described in Section 4.2. If the event subset shares no event with the enable event set $\mathbf{Enb}(\mathbf{x}_l)$, i.e., $\Psi(\mathbf{x}_l, a_l) = \emptyset$, then we set the current state $\mathbf{x}_l$ as the next state $\mathbf{x}_{l+1}$, and the target value of $(\mathbf{x}_l, a_l)$ is updated by

$$target_l = t(\mathbf{x}_l, \sigma, \mathbf{x}_{l+1}). \tag{10}$$

Otherwise we predict the target value by (8). Then we store $(\mathbf{x}_l, a_l, target_l)$ as an experience in the buffer B. Note that the proposed algorithm does not store the next state $\mathbf{x}_{l+1}$ as a component of an experience since it is nondeterministic under an action, and is randomly selected from the reachable state set to act as the next state if $\Psi(\mathbf{x}_l, a_l) \neq \emptyset$. Each episode terminates with two conditions, when a) $\mathbf{Enb}(\mathbf{x}_{l+1}) = \emptyset$, that is, $\mathbf{x}_{l+1}$ is a deadlock, and b) the threshold of the maximal number of steps *MaxStep* of an episode is reached.

We use *Step* and i to denote the total steps and the index of the learning iterations of the proposed algorithm, respectively. If an experience is added to the buffer, then *Step* is updated as *Step* = *Step* + 1. The neural network Q is trained in Lines 22–33 if the number of experiences in B is more than the mini-batch N and *Step* is an integer multiple

Algorithm 1: A model based deep reinforcement learning procedure for computing a nonblocking coordinator

Data: $\mathbf{x}, \Delta, \Psi, \mathcal{A}^K$, and a buffer B of length l_{B}
Result: An optimal policy f^*

1 Initialize global state-event subset value function Q with random weights $\boldsymbol{\theta}_0$ and the target value function $\hat{Q}$ with weights $\boldsymbol{\phi}_0 = \boldsymbol{\theta}_0$;
2 $i \leftarrow 1$;
3 $Step \leftarrow 0$;
4 **for** *episode=1 to MaxEpi* **do**
5 **for** *l = 0 to MaxStep* **do**
6 Randomly generate a number $\rho \in (0, 1)$;
7 **if** $\rho < \epsilon$ **then**
8 Randomly select an action $a_l \in \mathcal{A}^K$;
9 **else**
10 $a_l \leftarrow \arg\max_{a \in \mathcal{A}^K} Q(\mathbf{x}_l, a; \boldsymbol{\theta}_{i-1})$;
11 **end**
12 Update ϵ by (9);
13 **if** $\Psi(\mathbf{x}_l, a_l) = \emptyset$ **then**
14 $\mathbf{x}_{l+1} \leftarrow \mathbf{x}_l$;
15 Compute $target_l$ by (10);
16 **else**
17 Compute $target_l$ by (8);
18 Randomly choose a state $\mathbf{x}_{l+1} \in \cup_{\sigma \in \Psi(\mathbf{x}_l, a_l)}\{\Delta(\mathbf{x}_l, \sigma)\}$;
19 **end**
20 Store $(\mathbf{x}_l, a_l, target_l)$ in the buffer B;
21 $Step \leftarrow Step + 1$;
22 **if** $Step \geq N$ *and* $mod(Step, \tau_1) = 0$ **then**
23 Randomly sample N experiences from B;
24 $L(\boldsymbol{\theta}_i) \leftarrow \frac{1}{N}\sum_{j=1}^{N}(target_j - Q(\mathbf{x}_j, a_j; \boldsymbol{\theta}_i))^2$;
25 $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \alpha\frac{\partial L(\boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_i}$;
26 Update α by (11);
27 **if** $mod(i, \tau_2) = 0$ **then**
28 $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\theta}_i$;
29 **else**
30 $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\phi}_{i-1}$;
31 **end**
32 $i \leftarrow i + 1$;
33 **end**
34 **if** $\mathbf{Enb}(\mathbf{x}_{l+1}) = \emptyset$ **then** break;
35 **end**
36 **end**
37 $f^*(\cdot) \leftarrow \arg\max_{a \in \mathcal{A}^K} Q(\cdot, a; \boldsymbol{\theta})$;

of τ_1, which aims to improve the training efficiency. In addition, the target network is updated every τ_2 iterations to improve the training stability. The learning rate $\alpha \in (0, 1)$ is changed by

$$\alpha := \begin{cases} \alpha(1 - \alpha_{decay}) & \text{if } \alpha > \alpha_{min} \\ \alpha & \text{otherwise} \end{cases} \tag{11}$$

after each training, where α_{decay} is the decay rate and α_{min} is the minimal value of α. We stop the training if the threshold of episode *MaxEpi* is reached. We generate the nonblocking coordinator for the modular supervisors obtained in Section 4.1 after the convergence of the neural network Q. The optimal policy f^* is computed in Line 37, and the control pattern at a global state $\mathbf{x}$ is $\Psi(\mathbf{x}, f^*(\mathbf{x}))$.

5. Case studies

We apply the proposed method to two cases and compare our results with those of the standard DQN algorithm. All computations are implemented by PYTHON with a desktop with Intel(R) Core(TM) i7-10700K CPU @3.80 GHz, 8 cores, and 32.0 GB installed memory (RAM). Compared with the DQN algorithm, the proposed method shows a higher training efficiency in these cases.

5.1. An automatically guided vehicles system

We use a system of automatically guided vehicles (AGVs) [49, 11, 4] that supports a manufacturing workcell as our first example. There are 15 parts in total in the system: **IPS1**, **IPS2**, **WS1**, **WS2**, **WS3**, **AGV1**, **AGV2**, **AGV3**, **AGV4**, **AGV5**, **Zone1**, **Zone2**, **Zone3**, **Zone4** and **CPS**, where **IPS1** and **IPS2** are two input stations for different types of parts, **WS1**, **WS2**, and **WS3** are three workstations, **AGV1** to **AGV5** are five AGVs, **Zone1** to **Zone4** are four zones with specific capacity, and **CPS** is the station for completed parts. These AGVs travel on fixed circular routes, as shown in Fig. 2, to transport the parts to the appropriate places.

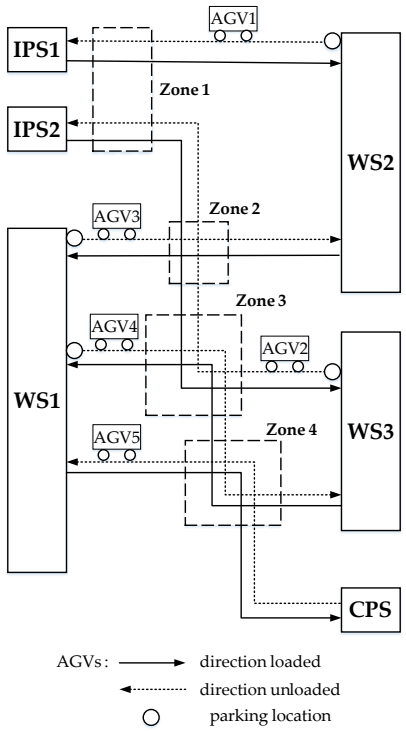


Figure 2: A manufacturing workcell with AGVs.

Figs. 3 and 4 respectively show the models of the five AGVs and eight individual specifications. The events in this case are denoted by σ_i ($i \in \mathbb{N}^+$), where the controllable events are labeled with odd number indexes and the uncontrollable events with even ones. Table 1 displays the meanings of the 26 events related to the system. The controllable and uncontrollable event sets of the system are respectively denoted as

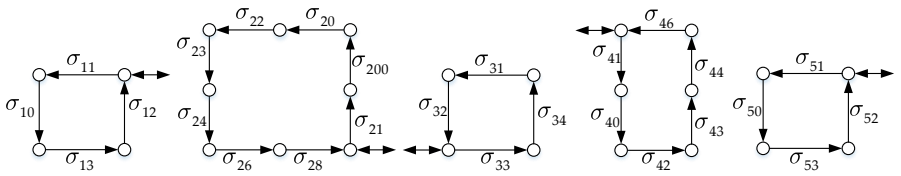
$$\Sigma_c = \{\sigma_{11}, \sigma_{13}, \sigma_{21}, \sigma_{23}, \sigma_{31}, \sigma_{33}, \sigma_{41}, \sigma_{43}, \sigma_{51}, \sigma_{53}\}$$

and

$$\Sigma_u = \{\sigma_{10}, \sigma_{12}, \sigma_{20}, \sigma_{22}, \sigma_{24}, \sigma_{26}, \sigma_{28}, \sigma_{32}, \sigma_{34}, \sigma_{40}, \sigma_{42}, \sigma_{44}, \sigma_{46}, \sigma_{50}, \sigma_{52}\}.$$

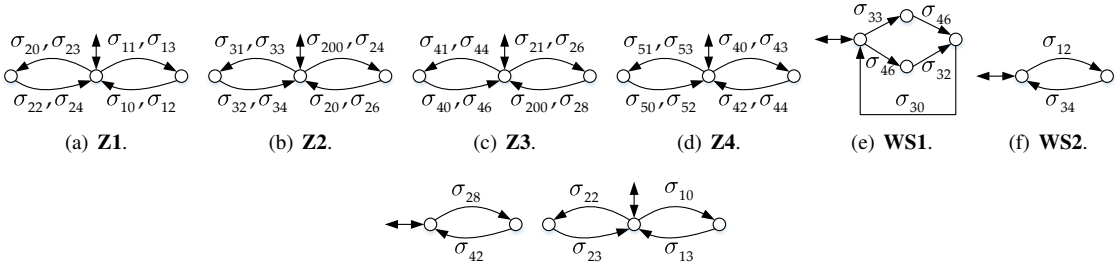
The proposed method first synthesizes eight reduced modular supervisors from the DFA models of the AGVs and the specifications. The details of these supervisors are displayed in Table 2. For instance, the reduced supervisor corresponding to the specification **Z1** is a DFA **Z1R** with 2 states and 42 transitions. A global state **x** in this case is constructed from the states of the five AGVs and the obtained eight reduced modular supervisors.

The objective of the AGV system is to ensure that the AGVs satisfy all control specifications and the entire manufacturing system is nonblocking. The satisfaction of the eight control specifications listed in Table 2 is realized by the eight



(a) **AGV1**. (b) **AGV2**. (c) **AGV3**. (d) **AGV4**. (e) **AGV5**.

Figure 3: DFA models of AGVs.



(a) **Z1**. (b) **Z2**. (c) **Z3**. (d) **Z4**. (e) **WS1**. (f) **WS2**.

(g) **WS3**. (h) **IPS**.

Figure 4: Specifications on AGVs.

Table 1: Events of AGVs

Event	AGV	Interpretation
σ_{11}	1	Unparks and enters **Zone1**
σ_{10}		Exits **Zone1** and loads from **IPS1**
σ_{13}		Re-enters **Zone1**
σ_{12}		Exits **Zone1**, unloads to **WS2**, and parks
σ_{21}	2	Unparks and enters **Zone3**
σ_{200}		Exits **Zone3** and enters **Zone2**
σ_{20}		Exits **Zone2** and enters **Zone1**
σ_{22}		Exits **Zone1** and loads from **IPS2**
σ_{23}		Re-enters **Zone1**
σ_{24}		Exits **Zone1** and re-enters **Zone2**
σ_{26}		Exits **Zone2** and re-enters **Zone3**
σ_{28}		Exits **Zone3**, unloads to **WS3**, and parks
σ_{33}	3	Unparks and enters **Zone2**
σ_{34}		Exits **Zone2** and loads from **WS2**
σ_{31}		Re-enters **Zone2**
σ_{32}		Exits **Zone2**, unloads to **WS1**, and parks
σ_{41}	4	Unparks and enters **Zone3**
σ_{40}		Exits **Zone3** and enters **Zone4**
σ_{42}		Exits **Zone4** and loads from **WS3**
σ_{43}		Re-enters **Zone4**
σ_{44}		Exits **Zone4** and re-enters **Zone3**
σ_{46}		Exits **Zone3**, unloads to **WS1**, and parks
σ_{51}	5	Unparks and enters **Zone4**
σ_{50}		Exits **Zone4** and loads from **WS1**
σ_{53}		Re-enters **Zone4**
σ_{52}		Exits **Zone4**, unloads to **CPS**, and parks

Table 2: Reduced modular supervisors

Specification	**Z1**	**Z2**	**Z3**	**Z4**	**WS1**	**WS2**	**WS3**	**IPS**
Reduced supervisor	**Z1R**	**Z2R**	**Z3R**	**Z4R**	**WS1R**	**WS2R**	**WS3R**	**IPSR**
States	2	2	2	2	4	2	2	2
Transitions	42	42	41	43	88	48	45	44

local modular supervisors presented in Table 2. The previous studies [11, 4] show that the conjunction of these modular supervisors causes deadlocks in the manufacturing system.

This section obtains the nonblocking coordinator by the proposed model-based DRL algorithm. Since the product is finalized by **AGV5**, we give rewards on the last three events (i.e., σ_{50}, σ_{53}, and σ_{52}) of **AGV5** to encourage the completion of products, with $r_e(\sigma_{50}) = 1, r_e(\sigma_{53}) = 5$ and $r_e(\sigma_{52}) = 10$. The special event σ_{52} means a task is finished, which indicates the reach of a global marker state, and thus we do not give reward to the marker state any more. Meanwhile, in order to avoid deadlocks during the transportation, we give the reward $r_s(\mathbf{x}') = -30$ if no possible events remains (i.e., $\mathbf{Enb}(\mathbf{x}') = \emptyset$). If the intersection is the empty set (i.e., $\Psi(\mathbf{x}, a) = \emptyset$), we give a reward -30. The rewards of other events and states are set to 0.

Table 3 shows the parameters used in Algorithm 1. The system has 10 controllable events, and the total number of the event subset is 2^{10}. To reduce the training difficulty, we restrict the maximal number of controllable events to 2 in an event subset, and the bounded set event subsets is $\mathcal{A}^2$, where $|\mathcal{A}^2| = C_{10}^0 + C_{10}^1 + C_{10}^2 = 56$. The Q value approximation is realized by a neural network, where we regard the global state $\mathbf{x}$ as the input and process it with two fully-connected layers of 128 units, while using ReLU (Rectified Linear Unit) as the activation function. The output is the estimated Q value with the dimension of $|\mathcal{A}^2|$.

Table 3: Parameters

parameter	l_B	*MaxEpi*	*MaxStep*	ϵ	ϵ_{decay}	N	α	α_{decay}	τ_1	τ_2
value	2^{13}	$1.3e5$	200	0.9	$5e{-}5$	2^{11}	$1e{-}3$	$2e{-}8$	100	100

Then we employ Algorithm 1 to obtain a nonblocking coordinator f^* for these modular supervisors. Fig. 5 displays the average rewards of the past 500 episodes by the traditional DQN and Algorithm 1 proposed in this study. We run 1.3×10^5 episodes for the two algorithms. Our algorithm needs about 9×10^4 episodes while the traditional DQN requires 1.1×10^5 episodes to get converged, and Algorithm 1 is 18.2% faster than DQN on the converging speed.

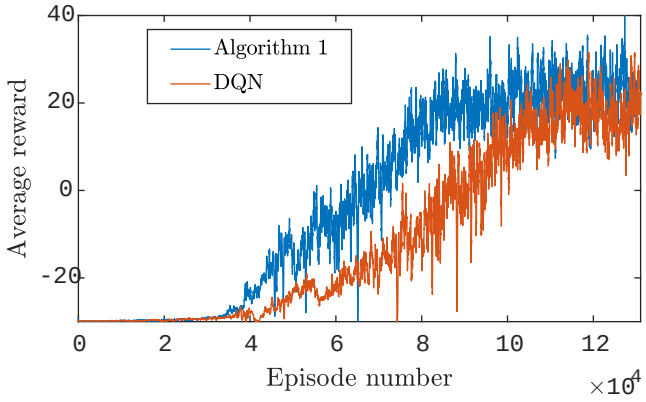
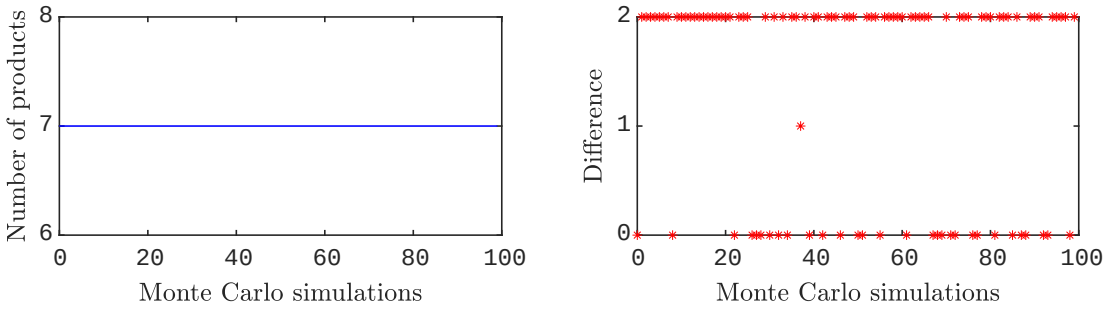


Figure 5: Plots of average rewards.

We test the coordinator by 100 Monte Carlo simulations with 200 steps in each simulation. Fig. 6(a) shows that there are 7 complete products in each Monte Carlo simulation, and there is no system blocking in these simulations. Moreover, the previous studies [49, 11] have revealed that the nonblocking coordinator should not allow the absolute difference between the two types of parts in the system larger than three. Our RL coordinator is confirmed to satisfy this constraint. The absolute difference in each simulation is displayed in Fig. 6(b), where the maximal and minimal values

are 2 and 0, respectively. The average and standard division values of the differences are respectively 1.4 and 0.9.



(a) The number of products.

(b) The absolute difference of part 1 and part 2 in the system.

Figure 6: Plots of Monte Carlo simulations.

5.2. A railway network with shared tracks

The second example is a railway network with shared tracks. Multiple trains may enter the railway network from two directions and travel through the network. Each train exclusively occupies a section of the track and may enter the next section if it is free. All trains must eventually leave the railway network. The supervisory control of the railway network has been studied in many papers [13, 6, 14]. As illustrated in Fig. 7, the railway network includes four stations and four tracks connecting the four stations. A station is represented by a rectangle and a track by a diamond. A station has two or three tracks as illustrated in Fig. 7. Trains enter and leave the stations from both sides, where switches of the stations can lead a train to enter or leave a given track. The solid arrows stand for the events moving a train from left to right, and the dotted arrows for the events moving a train from right to left. The labels of the events are on arrows, where the controllable events are labeled by σ with odd indexes and the uncontrollable events with even indexes. Table 4 displays the physical meanings of events related to Station i ($i \in [1, 4]$).

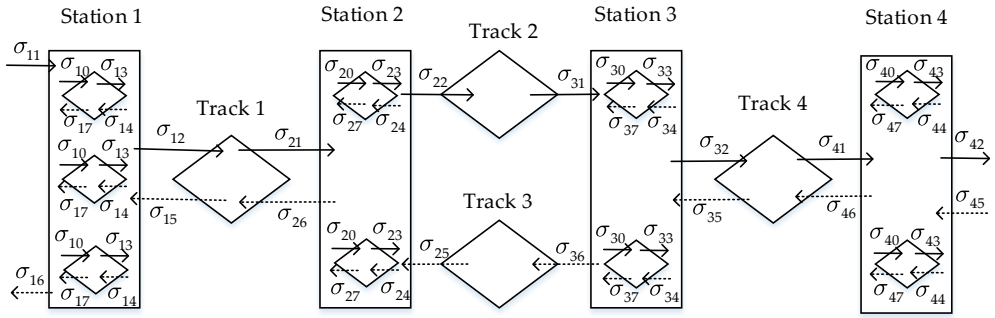
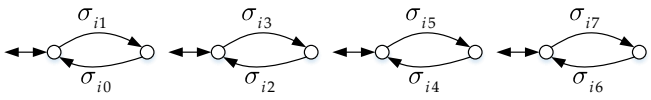


Figure 7: The structure of the railway network.

Table 4: Events of the railway networks

Event	Interpretation
σ_{i1}	A train enters station i from the left
σ_{i0}	The train enters a track segment in station i from the left
σ_{i3}	The train leaves the track segment to the right
σ_{i2}	The train leaves station i to the right
σ_{i5}	A train enters station i from the right
σ_{i4}	The train enters a track segment in station i from the right
σ_{i7}	The train leaves the track segment to the left
σ_{i6}	The train leaves station i the left

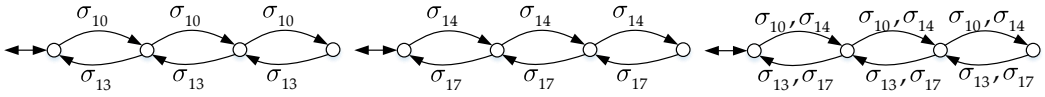
Fig. 8 shows the plant models of the four stations. For Station i, the plant models are described by four DFAs: $\mathbf{EL}_i$, $\mathbf{LR}_i$, $\mathbf{ER}_i$ and $\mathbf{LL}_i$. The synchronization of $\mathbf{EL}_i$, $\mathbf{LR}_i$, $\mathbf{ER}_i$ and $\mathbf{LL}_i$ is denoted by $\mathbf{Station}_i$. For example, Station 1



(a) $\mathbf{EL}_i$: Entering from the left. (b) $\mathbf{LR}_i$: Leaving to the right. (c) $\mathbf{ER}_i$: Entering from the right. (d) $\mathbf{LL}_i$: Leaving to the left.

Figure 8: Plant models of Station i.

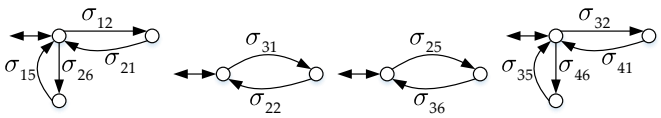
consists of three tracks, and the specifications related to it are shown in Fig. 9. The specifications act as a buffer with capacity three, and the synchronization of the four DFAs is denoted by $\mathbf{ST}_1$. Other stations have two tracks but not three tracks; thus each DFA that illustrates the specification has three states, and their models have the same structures with them as Station 1. Moreover, the names of plant models from Station 2 to 4 are $\mathbf{ST}_2, \mathbf{ST}_3$, and $\mathbf{ST}_4$. The details are in [6].



(a) Train from left to right. (b) Train from right to left. (c) Three tracks.

Figure 9: Specifications of station 1.

Fig. 10 shows the models of the four tracks connecting these stations. A train can enter Tracks 1 and 4 from both sides, while Tracks 2 and 3 only support one direction. In addition, Track 2 accepts a train from the left and releases it to right, and Track 3 accepts a train from the right and releases it to left.



(a) TR_1: Track 1. (b) TR_2: Track 2. (c) TR_3: Track 3. (d) TR_4: Track 4.

Figure 10: Tracks connecting stations.

In this case, the set of individual specifications is $\{\mathbf{ST}_1, \mathbf{ST}_2, \mathbf{ST}_3, \mathbf{ST}_4, \mathbf{TR}_1, \mathbf{TR}_2, \mathbf{TR}_3, \mathbf{TR}_4\}$, and the set of plant components is $\{\mathbf{EL}_i, \mathbf{LR}_i, \mathbf{ER}_i, \mathbf{LL}_i\}$ ($i \in [1,4]$). We first synthesize a set of local modular supervisors by the modular supervisory control theory. Table 5 lists the DFAs of local plants, the individual specifications, the controlled behaviors, and the reduced supervisors. The numbers in the parentheses denote the state size.

Table 5: Reduced supervisors

Specification	Local plant	Supervisor	Reduced supervisor
$\mathbf{ST}_1$	$\mathbf{EL}_1 \| \mathbf{LR}_1 \| \mathbf{ER}_1 \| \mathbf{LL}_1$	$\mathbf{CS}_1(100)$	$\mathbf{RS}_1(16)$
$\mathbf{ST}_2$	$\mathbf{EL}_2 \| \mathbf{LR}_2 \| \mathbf{ER}_2 \| \mathbf{LL}_2$	$\mathbf{CS}_2(52)$	$\mathbf{RS}_2(9)$
$\mathbf{ST}_3$	$\mathbf{EL}_3 \| \mathbf{LR}_3 \| \mathbf{ER}_3 \| \mathbf{LL}_3$	$\mathbf{CS}_3(52)$	$\mathbf{RS}_3(9)$
$\mathbf{ST}_4$	$\mathbf{EL}_4 \| \mathbf{LR}_4 \| \mathbf{ER}_4 \| \mathbf{LL}_4$	$\mathbf{CS}_4(52)$	$\mathbf{RS}_4(9)$
$\mathbf{TR}_1$	$\mathbf{LR}_1 \| \mathbf{ER}_1 \| \mathbf{EL}_2 \| \mathbf{LL}_2$	$\mathbf{CS}_5(20)$	$\mathbf{RS}_5(4)$
$\mathbf{TR}_2$	$\mathbf{LR}_2 \| \mathbf{EL}_3$	$\mathbf{CS}_6(6)$	$\mathbf{RS}_6(2)$
$\mathbf{TR}_3$	$\mathbf{ER}_2 \| \mathbf{LL}_3$	$\mathbf{CS}_7(6)$	$\mathbf{RS}_7(2)$
$\mathbf{TR}_4$	$\mathbf{LR}_3 \| \mathbf{ER}_3 \| \mathbf{EL}_4 \| \mathbf{LL}_4$	$\mathbf{CS}_8(20)$	$\mathbf{RS}_8(4)$

A global state is $\mathbf{x} = (y_1, \ldots, y_4, q_1, \ldots, q_8)$, where y_i belongs to $\mathbf{Station}_i$ ($i \in [1,4]$), and q_j belongs to $\mathbf{RS}_j$ ($j \in [1,8]$). The previous studies have shown that deadlocks arise under the eight local modular supervisors. The objective of

this section is to coordinate more trains from both directions to travel through the network without blocking. Due to the high complexity and large state space of the railway network system, the rare valuable transition samples are prone to be hidden in a mass of highly redundant failure cases during the training process. As a result, the majority of the memory space in the experience buffer is occupied by data with less learning value. To solve this issue, compared with the AGV case, we give a more comprehensive definition of the reward to overcome this *imbalanced training problem* [53], which normally arises from the coarse reward definition.

From Fig. 7, we define the rewards of events as

$$\begin{cases} r_e(\sigma_{42}) = 20, r_e(\sigma_{16}) = 20 \\ r_e(\sigma_{43}) = 10, r_e(\sigma_{17}) = 10 \\ r_e(\sigma_{41}) = 5, r_e(\sigma_{15}) = 5 \\ r_e(\sigma_{33}) = 2.5, r_e(\sigma_{27}) = 2.5 \\ r_e(\sigma_{31}) = 1, r_e(\sigma_{25}) = 1 \\ r_e(\sigma_{23}) = 0.5, r_e(\sigma_{37}) = 0.5 \\ r_e(\sigma_{21}) = 0.25, r_e(\sigma_{35}) = 0.25 \\ r_e(\sigma_{13}) = 0.1, r_e(\sigma_{47}) = 0.1 \\ r_e(\sigma_{11}) = 0.05, r_e(\sigma_{45}) = 0.05. \end{cases}$$

In addition, we assign state reward $r_s(\mathbf{x}') = -30$ if the next state $\mathbf{x}'$ is a deadlock but is not a marker state. The event σ_{16} or σ_{42} means a train leaves the stations, implying that a global marker state is reached. Thus we do not give rewards to the marker states any more. We promote the continuation of the forward-driving by increasing the reward of a step closer to the destination. In addition, the rewards of other events and states are set to 0.

However, one limitation of the reward function is that it fails to balance the trains from both directions. A high reward can be obtained even if all trains from one direction can pass the railway network but the trains from the other direction are blocked. This definition of reward will lead the system to converge into a local minimum where the network only contains trains running in one direction. To overcome this issue, we append two counters C_1 and C_2 into the global state to record the total number of trains exiting the railway network from Station 4 and Station 1, respectively. If a train exits from Station 4, i.e., event σ_{42} occurs, then $C_1 = C_1 + 1$. Similarly, C_2 is updated as $C_2 = C_2 + 1$ if event σ_{16} is executed. Thus, the absolute difference of the leaving trains from both directions is written as $|C_1 - C_2|$, and a positive reward is only given when the difference is small. In addition, the global state state is changed to $\mathbf{x}_{ext} = (\mathbf{x}, C_1, C_2) \in \mathbb{R}^{14}$.

When the number of arrival trains from one side exceeds the other, a negative reward is assigned to the corresponding actions that send trains from the same direction. Thus, the balance of the number of arrival trains from both ends can be achieved. Accordingly, we develop a *sigmoid*-type reward function to realize this purpose, given by

$$t(\mathbf{x}_{\text{ext}}, \sigma, \mathbf{x}'_{\text{ext}}) = \frac{p_1 r_e(\sigma)}{1 + e^{p_2|C_1 - C_2|}} + p_3 + r_s(\mathbf{x}'_{\text{ext}}), \tag{12}$$

where p_1, p_2 and p_3 denote the shape coefficients in the reward function, and $\mathbf{x}'_{\text{ext}}$ is the next state of $\mathbf{x}_{\text{ext}}$ after executing the event σ. The *logistic curve* in the reward function (12) provides a saturation in the reward per step, ensuring that the value of a rare action will not be excessively overestimated. In addition, here we maintain the previous setting $r_s(\mathbf{x}'_{\text{ext}}) = 0$ for other states.

The system has 16 controllable events, and the total number of the event subsets that include all uncontrollable event is 2^{16}. We also restrict the maximal number of controllable events to 2 in an event subset, thus $|\mathcal{A}^2| = C_{16}^0 + C_{16}^1 + C_{16}^2 = 137$. Moreover, we initialize action a_1 such that the corresponding event subset is $\Omega^{-1}(a_1) = \{\sigma_{11}, \sigma_{45}\}$, which allows trains to enter the railway network from both sides at the first step. Owing to the complexity of the railway system model, compared with the neural network implemented in the AGV section, we append one hidden layer in the network and increase the number of neurons in each layer to 256. The activation function is still achieved by ReLU.

We apply Algorithm 1 and obtain the optimal control function f^*. Fig. 11 displays the average rewards of the past 500 episodes by the traditional DQN and Algorithm 1. We run 4×10^5 episodes for the two algorithms. The result shows that the proposed algorithm can handle 12 arrival trains but the traditional DQN algorithm leads to a system blocking within 200 steps starting from the initial global state. The training process is repeated several times with different random seeds for validity check. Table 6 shows the parameters used in this case.

The reult is tested by 100 Monte Carlo simulations with 200 steps. We set six trains at each side of the train system as the initialization of each test simulation. Fig. 12(a) shows the number of steps needed in each simulation for the 12 trains to exit the railway network. The number of steps needed in each simulation is smaller than the maximal step we

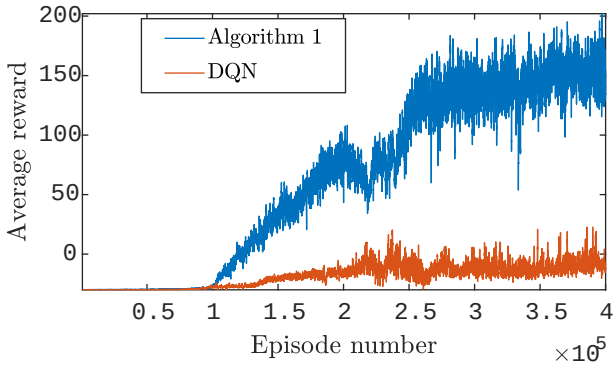
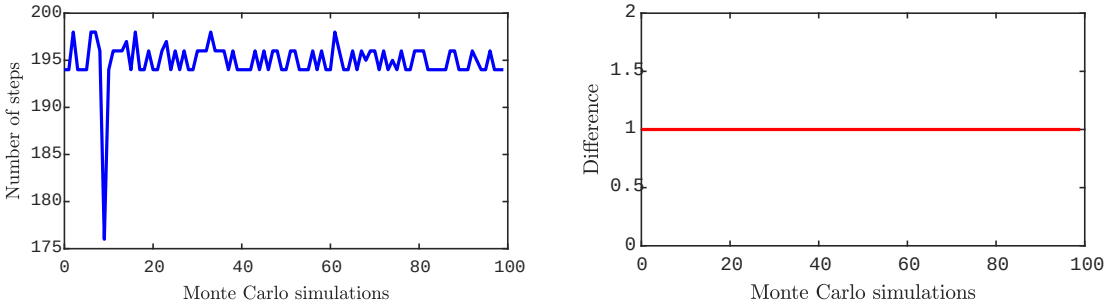


Figure 11: Plots of average rewards.

Table 6: Parameters

parameter	l_B	*MaxEpi*	*MaxStep*	ϵ	ϵ_{decay}	N	α	α_{decay}	τ_1	τ_2	p_1	p_2	p_3
value	2^{13}	$4e5$	200	0.9	$5e{-}5$	2^{11}	$1e{-}3$	$2e{-}8$	100	100	2	-2	-0.762

set, which indicates that these simulations are nonblocking. Fig. 12(b) displays the maximal absolute difference of the exiting trains between two sides, which indicates that the trains alternately exit the station from Stations 1 and 4. In summary, with the control of the obtained coordinator and the modular supervisors, there does not exist any conflict in the stations if the number of trains entering the stations from one side is less than 7.



(a) The number of steps.

(b) Maximal value of $| C_1 - C_2 |$.

Figure 12: Plots of Monte Carlo simulations.

The Petri net approach in [13, 14] is only suitable when the number of trains in the network is less than 7. The automata approach in [6] is more permissive since it allows more than 7 trains concurrently running in the network. Our solution allows maximally four trains running concurrently in the network.

6. Conclusions and Future Works

This paper presents a method for synthesizing nonblocking modular supervisors for a large-scale DES by integrating the modular supervisory control theory and a deep reinforcement learning method. The estimation of the value in the learning process is based on event subsets that include all uncontrollable events, ensuring compliance with supervisory control theory. Consequently, we obtain a nonblocking coordinator for the modular supervisors obtained by SCT.

In the future, we plan to use linear temporal logic to describe the specifications in DESs because of its rich expressivity. A linear temporal logic formula can be converted to an ω-automaton with an acceptance condition. Then, a learning algorithm is used to generate a global nonblocking supervisor that meets the control requirements.

Disclosure statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

The work is supported by Chinese Scholarship Council (CSC), KTH 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, and the Science Technology Development Fund, MSAR, under Grant 0064/2021/A2.

Notes on contributor(s)

Junjun Yang : Conceptualization of this study, Methodology, Software, Formal analysis, Investigation, Writing Original Draft. **Kaige Tan** : Software. **Lei Feng**, **Zhiwu Li** : Supervision, Writing-Review & Editing.

References

[1] Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34, 26–38.

[2] van Beek, D.A., Reniers, M.A., Rooda, J.E., Schiffelers, R.R., 2008. Concrete syntax and semantics of the compositional interchange format for hybrid systems. IFAC Proceedings Volumes 41, 7979–7986.

[3] Cassandras, C.G., Lafortune, S., 2008. Introduction to discrete event systems. Springer, New York, USA.

[4] Cheng, L., Feng, L., Li, Z., 2021. Model abstraction for discrete-event systems by binary linear programming with applications to manufacturing systems. Science Progress 104, 1–32.

[5] Coogan, S., Arcak, M., Belta, C., 2017. Formal methods for control of traffic flow: Automated control synthesis from finite-state transition models. IEEE Control Systems Magaine 37, 109–128.

[6] Feng, L., 2007. Computationally efficient supervisor design for discrete-event systems. Ph.D. thesis. University of Toronto.

[7] Feng, L., Cai, K., Wonham, W.M., 2009. A structural approach to the non-blocking supervisory control of discrete-event systems. IEEE Transactions on Automatic Control 53, 1449–1461.

[8] Feng, L., Wonham, W.M., 2006a. Computationally efficient supervisor design: Control flow decomposition, in: Proceedings of IEEE 8th International Workshop on Discrete Event Systems, pp. 9–14.

[9] Feng, L., Wonham, W.M., 2006b. TCT: A computation tool for supervisory control synthesis, in: Proceedings of IEEE 8th International Workshop on Discrete Event Systems, pp. 3–8.

[10] Feng, L., Wonham, W.M., 2007. Nonblocking coordination of discrete-event systems by control-flow nets, in: Proceedings of IEEE 46th Conference on Decision and Control, pp. 3375–3380.

[11] Feng, L., Wonham, W.M., 2008. Supervisory control architecture for discrete-event systems. IEEE Transactions on Automatic Control 53, 1449–1461.

[12] Feng, L., Wonham, W.M., 2010. On the computation of natural observers in discrete-event systems. Discrete Event Dynamic Systems 20, 63–102.

[13] Giua, A., Seatzu, C., 2001. Supervisory control of railway networks with Petri nets, in: Proceedings of IEEE 40th Conference on Decision and Control, pp. 5004–5009.

[14] Giua, A., Seatzu, C., 2008. Modeling and supervisory control of railway networks using Petri nets. IEEE Transactions on Automation Science and Engineering 5, 431–445.

[15] Gohari, P., Wonham, W.M., 2000. On the complexity of supervisory control design in the RW framework. IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics, Special Issue on DES 30, 643–652.

[16] Goorden, M., van de Mortel-Fronczak, J., Reniers, M., Fabian, M., Fokkink, W., Rooda, J., 2021. Model properties for efficient synthesis of nonblocking modular supervisors. Control Engineering Practice 112, 104830.

[17] Hill, R.C., Tilbury, D.M., 2006. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction, in: Proceedings of IEEE 8th International Workshop on Discrete Event Systems, pp. 399–406.

[18] Kajiwara, K., Yamasaki, T., 2010. Adaptive supervisory control based on a preference of agents for decentralized discrete event systems, in: Proceedings of IEEE SICE Annual Conference, pp. 1027–1032.

[19] Kajiwara, K., Yamasaki, T., 2013. Reinforcement learning of optimal supervisor for discrete event systems with different preferences. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 96, 525–531.

[20] Leduc, R.J., Brandin, B.A., Lawford, M., Wonham, W.M., 2005a. Hierarchical interface-based supervisory control-part I: Serial case. IEEE Transactions on Automatic Control 50, 1322–1335.

[21] Leduc, R.J., Lawford, M., Wonham, W.M., 2005b. Hierarchical interface-based supervisory control-part II: Parallel case. IEEE Transactions on Automatic Control 50, 1336–1348.

[22] Lewis, F.L., Vrabie, D., Vamvoudakis, K.G., 2012. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controller. IEEE Control Systems Magazine 32, 76–105.

[23] Li, D., Meng, L., Li, J., Lu, K., Yang, Y., 2022a. Domain adaptive state representation alignment for reinforcement learning. Information Sciences 609, 1353–1368.

[24] Li, D., Yin, L., Wang, J., Wu, N., 2022b. Game current-state opacity formulation in probabilistic resource automata. Information Sciences 613, 96–113.

[25] Lin, F., Wonham, W.M., 1988. Decentralized supervisory control of discrete-event systems. Information Sciences 44, 199–224.

[26] Malik, R., Åkesson, K., Flordal, H., Fabian, M., 2017. Supremica–an efficient tool for large-scale discrete event systems. IFAC-PapersOnLine 50, 5794–5799.

[27] Malik, R., Teixeira, M., 2021. Optimal modular control of discrete event systems with distinguishers and approximations. Discrete Event Dynamic Systems 31, 659–691.

[28] Masopust, T., 2017. Complexity of verifying nonblockingness in modular supervisory control. IEEE Transactions on Automatic Control 63, 602–607.

[29] Queiroz, M.H.D., Cury, J.E., 2000. Modular supervisory control of large scale discrete event systems, in: Proceedings of International Conference on Discrete Event Systems, pp. 103–110.

[30] Ramadge, P.J., Wonham, W.M., 1989. The control of discrete event systems. Proceedings of the IEEE 77, 81–98.

[31] Rudie, K., 2006. The integrated discrete-event systems tool, in: Proceedings of IEEE 8th International Workshop on Discrete Event Systems, pp. 394–395.

[32] Saravanakumar, T., Anthoni, S.M., Zhu, Q., 2020. Resilient extended dissipative control for markovian jump systems with partially known transition probabilities under actuator saturation. Journal of the Franklin Institute 357, 6197–6227.

[33] Schmidt, K., Marchand, H., Gaudin, B., 2006. Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models, in: Proceedings of IEEE 8th International Workshop on Discrete Event Systems, pp. 149–154.

[34] Schmidt, K., Moor, T., Perk, S., 2008. Nonblocking hierarchical control of decentralized discrete event systems. IEEE Transactions on Automatic Control 53, 2252–2265.

[35] Seow, K.T., 2013. Organizational control of discrete-event systems: A hierarchical multiworld supervisor design. IEEE Transactions on Control systems technology 22, 23–33.

[36] Su, R., Wonham, W.M., 2004. Supervisor reduction for discrete-event systems. Discrete Event Dynamic Systems 14, 31–53.

[37] Su, R., Wonham, W.M., 2018a. A generalized theory on supervisor reduction, in: Proceedings of IEEE 57th Conference on Decision and Control, pp. 3950–3955.

[38] Su, R., Wonham, W.M., 2018b. What information really matters in supervisor reduction? Automatica 95, 368–377.

[39] Sutton, R.S., Barto, A.G., 2015. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press.

[40] Taş, M.K., Kaya, K., Yenigün, H., 2021. Synchronizing billion-scale automata. Information Sciences 574, 162–175.

[41] Ushio, T., Yamasaki, T., 2003. Supervisory control of partially observed discrete event systems based on a reinforcement learning, in: Proceedings of 2003 IEEE International Conference on Systems, Man and Cybernetics, pp. 2956–2961.

[42] Wang, D., Wang, X., Li, Z., 2019a. Nonblocking supervisory control of state-tree structures with conditional-preemption matrices. IEEE Transactions on Industrial Informatics 16, 3744–3756.

[43] Wang, R., Guan, Y., Song, H., Li, X., Li, X., Shi, Z., Song, X., 2019b. A formal model-based design method for robotic systems. IEEE Systems Journal 13, 1096–1107.

[44] Wang, Y., Li, Y., Yu, Z., Wu, N., Li, Z., 2021. Supervisory control of discrete-event systems under external attacks. Information Sciences 562, 398–413.

[45] Watkins, C.J., Dayan, P., 1992. Q-learning. Machine Learning 8, 279–292.

[46] Wong, K.C., Thistle, J.G., Malhamé, R.P., Hoang, H.H., 2000. Supervisory control of distributed systems: Conflict resolution. Discrete Event Dynamic Systems 10, 131–186.

[47] Wong, K.C., Wonham, W.M., 1998. Modular control and coordination of discrete-event systems. Discrete Event Dynamic Systems 8, 247–297.

[48] Wong, K.C., Wonham, W.M., 2004. On the computation of observers in discrete-event systems. Discrete Event Dynamic Systems 14, 55–107.

[49] Wonham, W.M., Cai, K., 2019. Supervisory control of discrete-event systems. Springer.

[50] Wonham, W.M., Cai, K., Rudie, K., 2018. Supervisory control of discrete-event systems: A brief history. Annual Reviews in Control 45, 250–256.

[51] Wonham, W.M., Ramadge, P.J., 1988. Modular supervisory control of discrete-event systems. Mathematics of Control, Signals and Systems 1, 13–30.

[52] Yamasaki, T., Ushio, T., 2004. Decentralized supervisory control of discrete event systems based on reinforcement learning. IFAC Proceedings Volumes 37, 367–372.

[53] Yuan, W., Li, Y., Zhuang, H., Wang, C., Yang, M., 2021. Prioritized experience replay-based deep Q learning: Multiple-reward architecture for highway driving decision making. IEEE Robotics & Automation Magazine 28, 21–31.

[54] Yuan, Y., Yu, Z.L., Gu, Z., Deng, X., Li, Y., 2019. A novel multi-step reinforcement learning method for solving reward hacking. Applied Intelligence 49, 2874–2888.

[55] Zeigler, B.P., 2018. Discrete event system specification framework for self-improving healthcare service systems. IEEE Systems Journal 12, 196–207.

[56] Zielinski, K.M., Hendges, L.V., Florindo, J.B., Lopes, Y.K., Ribeiro, R., Teixeira, M., Casanova, D., 2021. Flexible control of discrete event systems using environment simulation and reinforcement learning. Applied Soft Computing 111, 107714.

Declaration of interests

☒The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: