machines

MDPI

Article

Nonblocking Modular Supervisory Control of Discrete Event Systems via Reinforcement Learning and K-Means Clustering

Junjun Yang [1], Kaige Tan [2] and Lei Feng [2,*]

[1] School of Automation and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China; junjuny@lzjtu.edu.cn
[2] Department of Engineering Design, KTH Royal Institute of Technology, 10044 Stockholm, Sweden; kaiget@kth.se
* Correspondence: lfeng@kth.se

Abstract

Traditional supervisory control methods for the nonblocking control of discrete event systems often suffer from exponential computational complexity. Reinforcement learning-based approaches mitigate state explosion by sampling many random sequences instead of computing the synchronous product of multiple modular supervisors, but they struggle with limited reachable state spaces. A primary novelty of this study is to use the K-means clustering method for online inference with the learned state-action values. The clustering method divides all events at a state into the good group and the bad group. The events in the good group are allowed by the supervisor. The obtained supervisor policy can ensure both system constraints and larger control freedom compared to conventional RL-based supervisors. The proposed framework is validated by two case studies: an industrial transfer line (TL) system and an automated guided vehicle (AGV) system. In the TL case study, nonblocking reachable states increase from 56 to 72, while in the AGV case study, a substantial expansion from 481 to 3558 states is observed. Our new method achieves a balance between computational efficiency and nonblocking supervisory control.

Keywords: discrete event system; supervisory control theory; reinforcement learning; K-means clustering

check for updates

Academic Editors: Wilfried Yves Hamilton Adoni, Moez Krichen and Jamal Berrich

Received: 10 May 2025
Revised: 24 June 2025
Accepted: 25 June 2025
Published: 27 June 2025

Citation: Yang, J.; Tan, K.; Feng, L. Nonblocking Modular Supervisory Control of Discrete Event Systems via Reinforcement Learning and K-Means Clustering. *Machines* **2025**, *13*, 559. https://doi.org/10.3390/machines13070559

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Industrial automation requires multiple autonomous machines to work collaboratively to produce products with high productivity and safety. These machines include, for instance, assembly stations, mobile logistic vehicles, and storage buffers. The movement and operations of these machines must be coordinated by logic-level supervisory controllers. The supervisory control theory (SCT) [1,2] provides a theoretical framework for designing nonblocking and maximally permissive supervisors for discrete event systems (DESs). The supervisors may be designed by traditional exhaustive search methods or emerging sample-based methods using reinforcement learning (RL).

The traditional search-based methods suffer from exponential computational complexity. To reduce complexity, modular supervisory control [3] decomposes global specifications into multiple smaller local specifications, which are enforced by local modular supervisors [4]. Only the plant components that share events with the local specification are considered when designing the local modular supervisor. These supervisors collectively ensure compliance with the global specification. However, blocking states may appear in

the controlled system due to the lack of coordination between independently designed local modular supervisors. A straightforward way to ensure nonblocking control is to introduce a coordinator to resolve the conflicting control decisions among modular supervisors. Nevertheless, the complexity of computing such a nonblocking coordinator can still grow exponentially with the number of supervisors and plant components. Developing coordination mechanisms that ensure nonblocking behavior with tractable computational complexity remains an open research question.

1.1. Traditional Methods for Coordinator Computing

Traditional SCT employs two primary strategies to efficiently achieve nonblocking modular supervisory control. The first strategy leverages the structural properties of the system to ensure that local modular supervisors maintain the nonblocking property. For instance, Feng and Wonham [5] propose the control-flow-network method, which analyzes the interactions among local supervisors and plant components. This method identifies specific network configurations that inherently guarantee nonblocking behavior for certain modular supervisors.These favorable network structures are common in manufacturing lines. In addition, Goorden et al. [6] present three structural properties that can be checked to verify nonblocking behavior in modular supervisory control applications.

The second strategy aims to alleviate computational complexity by employing system decomposition and hierarchical abstraction techniques in the coordinator synthesis process. This approach partitions the system into smaller, more manageable subsystems and abstracts their behavior into higher-level models, enabling efficient synthesis of a global supervisor. For instance, Schmidt et al. [7,8] propose a hierarchical abstraction technique that uses state aggregation to reduce the state space of individual modules while preserving their essential control properties. Their method significantly reduces the computational burden of synthesizing a coordinator, particularly in large-scale systems. Similarly, Leduc et al. [9] introduce a compositional synthesis framework that combines abstraction and decomposition to achieve scalable nonblocking control. Their approach leverages the concept of "partial models," where each module is abstracted independently, and the global supervisor is synthesized incrementally. This method not only reduces computational complexity but also enhances the flexibility of the control method.

Recent advancements have further extended these strategies by integrating learning-based methods and formal verification tools. For instance, learning-based techniques are exploited for synthesizing modular supervisors in uncertain environments [10,11]. Additionally, tools like Supremica [12] and MIDES [13] have been developed to automate the synthesis and verification of modular supervisors, enabling the application of these strategies to industrial-scale systems. These developments highlight the ongoing evolution of SCT, where traditional strategies are being enhanced with modern computational techniques to address the challenges of increasingly complex systems.

1.2. RL-Based Methods

RL [14] has emerged as a promising alternative to traditional supervisory control methods in DESs. Traditional methods exhaustively search the state-transition model of the controlled system and prune unwanted states, but RL learns control policies through sample-based simulations of the target system. This sample-based approach effectively circumvents the state explosion problem commonly encountered in conventional coordinator design techniques. Kajiwara and Yamasaki [15] propose a RL-based approach for implementing supervisory control in decentralized DESs, where the central supervisor and local subsystems are characterized by distinct preference functions. The reward function is designed to account for both event occurrence and disabling during the learning process.

Furthermore, Zielinski et al. [16] integrate SCT and RL to assign an optimal schedule to an industrial system, where SCT is utilized to build a safety search environment for RL algorithms. Hu et al. [17] combine SCT and RL to synthesize an optimal directed controller for a DES. Deep reinforcement learning (DRL) algorithms are incorporated into the above framework to effectively address control problems in large-scale DESs. Konishi et al. [18] integrate DRL and SCT to solve a safety control problem in DESs, where DRL algorithms are used to compute a suboptimal solution efficiently and SCT is employed to restrict the sub-optimal solution to a safety solution. Liu [10] combines SCT and RL to design nonblocking modular supervisors for DESs with unknown environments. Our previous research [19] employs SCT and DRL to compute a nonblocking coordinator for a set of local modular supervisors. Furthermore, SCT is employed to reduce the learning time of DRL for supervisor designing for DESs with linear temporal logic specifications [20].

1.3. Motivation and Contributions

Classical SCT strives for the supremal supervisor with the greatest reachable state set, but RL-based supervisors do not explicitly maximize the reachable state set. Hence, the RL-based supervisors are often limited in the reachable state space.

At a reachable state, a supervisor allows all uncontrollable events and a subset of controllable events. The cardinality of the controllable event subset is flexible, ranging from 0 to all controllable events. This flexibility helps the supervisor satisfy all control specifications and achieve the largest reachable state set. In contrast, RL algorithms choose only one action at a reachable state. This limitation significantly confines the reachable state set of the controlled systems. Our previous studies [19,20] attempt to enlarge the reachable state set of the RL-learned supervisor by selecting multiple actions, but these methods choose a fixed number of controllable events and incur longer training time. The flexibility of choosing an arbitrary number of controllable events at a reachable state is desirable for the RL-learned supervisor.

This paper adds this missed flexibility to RL-learned supervisors. Suppose that an RL algorithm maximizes the reward. After the learning process, the learned supervisor has evaluations for all pairs of reachable state and controllable events. If the supremal nonblocking supervisor allows a controllable event at a state, the evaluation number of the pair of state and event may be high. Otherwise, the evaluation of the pair may be low. A research hypothesis is that all events with higher evaluations learned by RL should be allowed by the supremal nonblocking supervisor. Since there is not a fixed threshold value for dichotomizing the controllable events, we apply the clustering method to identify the pattern of the events and choose the flexible group of allowed events. Among clustering algorithms like hierarchical clustering [21], support vector machine [22], a density-based method [23], and fuzzy C-Means [24], K-means [25] emerges as the most suitable choice for this study because its simplicity and computational efficiency align with the binary classification requirement of evaluation numbers.

We evaluate the proposed method with a transfer line system and an automated guided vehicle (AGV) system. The results show that the proposed method greatly enlarges the state size of the controlled system, compared with the method only using RL. The structure of the paper is as follows. Section 2 reviews the necessary preliminaries. The research problem and its solution are presented in Section 4. Section 5 demonstrates the implementation of the proposed method and Section 6 concludes the paper.

2. Preliminaries

2.1. Deterministic Finite Automata (DFA)

A DFA is formally represented as a five-tuple $\mathbf{G} = \langle Q, \Sigma, \eta, q_0, Q_m \rangle$, where:

- Q denotes a finite, nonempty set of states;
- Σ is a finite, nonempty alphabet representing the set of event labels;
- $\eta : Q \times \Sigma \to Q$ is a (partially defined) transition function;
- $q_0 \in Q$ is the designated initial state;
- $Q_m \subseteq Q$ represents the set of marker (or accepting) states.

For a state $q \in Q$ and an event $\sigma \in \Sigma$, denote $\eta(q,\sigma)!$ if $\eta(q,\sigma)$ is defined. The set of events that are defined at q is $Enb(q) = \{\sigma \in \Sigma \mid \eta(q,\sigma)!\}$; q is a deadlock state if $Enb(q) = \emptyset$.

An event reward function $R_e : \Sigma \to \mathbb{R}$ and a state reward function $R_s : Q \to \mathbb{R}$ are defined for DESs as follows. For an event $\sigma \in \Sigma$,

$$R_e(\sigma) = \begin{cases} w_1, & \text{if } \sigma \in \Sigma_s \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $\Sigma_s \subseteq \Sigma$ defines a specific event set, and w_1 is a positive real number. For a state $q \in Q$,

$$R_s(q) = \begin{cases} w_1, & \text{if } q \in Q_m \\ w_2, & \text{if } q \notin Q_m \& Enb(q) = \emptyset \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where w_2 is a negative real number.

2.2. SCT and Local Modular Supervisors

A plant in SCT is modeled as a DFA $\mathbf{G}$, whose event set Σ is partitioned into controllable event set Σ_c and uncontrollable event set Σ_u. The specification of the plant behavior is described by another DFA $\mathbf{E}$, whose event set is also Σ. A supervisory controller is designed to ensure the plant satisfies the desired requirements by disabling some controllable events. The supervisory control function is $V : Q \to \mathcal{C}$, where $\mathcal{C} = \{\Pi \subseteq \Sigma | \Sigma_u \subseteq \Pi\}$ denotes the set of all event subsets that may be enabled at a given system state. Note that uncontrollable events are always permitted to occur. At a system state $q \in Q$, the supervisory control function determines a permissible event set $V(q)$, and all events not in $V(q)$ are disabled. Consequently, the closed-loop behavior of the plant $\mathbf{G}$ is constrained to a subset of the intersection of the behaviors of the plant $\mathbf{G}$ and the specification $\mathbf{E}$, and the closed-loop behavior is nonblocking. SCT synthesizes a maximally permissive supervisor $\mathbf{S}^*$ that permits the greatest reachable state set of the closed-loop behavior.

Many DES control problems involve multiple concurrent plant components and modular control specifications. Let $\mathbf{G}_i$ $(i \in 1,\ldots,b)$ and $\mathbf{E}_j$ $(j \in 1,\ldots,d)$ denote the DFAs modeling the i-th plant component and the j-th specification, respectively, each defined over event sets Σ_i and Σ'_j. Here, b and d are integers greater than one. The set of plant components that interact with specification $\mathbf{E}_j$ is given by $Share(\mathbf{E}_j) = \{i \in 1,\ldots,b \mid \Sigma_i \cap \Sigma'_j \neq \emptyset\}$. The synchronous product of two DFAs is defined as follows. Given two DFAs $\mathbf{G}_1 = \langle Q_1, \Sigma_1, \eta_1, q_{10}, Q_{1m}\rangle$ and $\mathbf{G}_2 = \langle Q_2, \Sigma_2, \eta_2, q_{20}, Q_{2m}\rangle$, their synchronization is $\mathbf{G}_1 \| \mathbf{G}_2 = \langle Q, \Sigma, \eta, q, Q_m\rangle$, where

- $Q = Q_1 \times Q_2$ is the state set;
- $\Sigma = \Sigma_1 \cup \Sigma_2$ is the set of events;

- $\eta : Q \times \Sigma \to Q$ is the partial transition function. For $q = (q_1, q_2) \in Q$ and $\sigma \in \Sigma$,

$$\eta(q,\sigma) = \begin{cases} (q_1', q_2), & \text{if } \sigma \in \Sigma_1 - \Sigma_2, q_1' = \eta_1(q_1,\sigma) \\ (q_1, q_2'), & \text{if } \sigma \in \Sigma_2 - \Sigma_1, q_2' = \eta_2(q_2,\sigma) \\ (q_1', q_2'), & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2, q_1' = \eta_1(q_1,\sigma), q_2' = \eta_2(q_2,\sigma) \\ \text{undefined}, & \text{otherwise;} \end{cases}$$

- $q_0 = (q_{10}, q_{20})$ is the initial state;
- $Q_m = Q_{1m} \times Q_{2m}$ is the set of marker states.

The operation can be extended to more than two DFAs. For each specification $\mathbf{E}_j$, the associated local plant is constructed as $\mathbf{H}_j = \|_{i \in Share(\mathbf{E}_j)} \mathbf{G}_i$. The corresponding modular supervisor $\mathbf{S}_j$, which enforces the specification $\mathbf{E}_j$, is computed from the composition of $\mathbf{H}_j$ and $\mathbf{E}_j$ using SCT [26,27]. Consequently, the concurrent behavior of all plant components $\mathbf{G}_i = \langle Q_i, \Sigma_i, \eta_i, q_{i0}, Q_{im} \rangle$ satisfies all specifications under the supervisory control of $\mathbf{S}_j = \langle Y_j, \Sigma, \delta_j, y_{j0}, Y_{jm} \rangle$. The overall behavior of the controlled DES is captured by the synchronous composition of all local modular supervisors along with their respective plant models. As noted in Section 1, the controlled DES may exhibit blocking behavior, which necessitates a coordinator to guarantee nonblocking execution [1,28].

2.3. Optimal Controller for the Markov Decision Process

A Markov decision process (MDP) is formally represented as a five-tuple $\mathcal{M} = \langle S, \mathcal{A}, \mathcal{T}, s_0, \mathcal{R} \rangle$:

- S denotes a finite state space containing at least one state;
- $\mathcal{A}$ represents the finite set of available actions;
- $\mathcal{T} : S \times \mathcal{A} \times S \to [0,1]$ specifies state transition probabilities;
- $s_0 \in S$ identifies the initial system state;
- $\mathcal{R} : S \times \mathcal{A} \times S \to \mathbb{R}$ defines the step reward function.

Within this framework, a deterministic control policy maps states to actions through a function $\pi : S \to \mathcal{A}$. Implementing policy π on $\mathcal{M}$ generates a Markov chain $\mathcal{M}_\pi$ with transition probabilities $\mathcal{T}(s, \pi(s), s')$ between states s and s'. The system's evolution under this policy produces infinite trajectories $\tau = (s_0, a_0, s_1), (s_1, a_1, s_2), \ldots$ where each transition satisfies $\mathcal{T}(s_i, a_i, s_{i+1}) > 0$ with $a_i = \pi(s_i)$.

The cumulative discounted return for trajectory τ is calculated as:

$$G(\tau) = \sum_{k=0}^{\infty} \gamma^k \mathcal{R}(s_k, a_k, s_{k+1}) \tag{3}$$

where $\gamma \in (0,1]$ serves as the discount factor, progressively diminishing the impact of future rewards. The state-value function $V_\pi : S \to \mathbb{R}$ quantifies the expected cumulative discounted return from state s:

$$V_\pi(s) = \mathbb{E}_\pi[G(\tau) \mid s_0 = s] \tag{4}$$

Following Bellman's principle of optimality [14], this value function satisfies the recursive relationship:

$$V_\pi(s) = \sum_{s' \in S} \mathcal{T}(s, \pi(s), s') \left[\mathcal{R}(s, \pi(s), s') + \gamma V_\pi(s') \right] \tag{5}$$

The optimal control policy π^* maximizes expected returns across all states:

$$\pi^* = \arg\max_{\pi} \sum_{s' \in S} \mathcal{T}(s, \pi(s), s') \left[\mathcal{R}(s, \pi(s), s') + \gamma V_\pi(s') \right] \tag{6}$$

There are two main approaches to determining π^*: model-based methods using dynamic programming when full knowledge of $\mathcal{T}$ and $\mathcal{R}$ is available, and model-free reinforcement learning methods when transition dynamics are unknown.

2.4. K-Means Clustering

K-means clustering [25] is a popular unsupervised learning algorithm designed to partition a dataset into multiple disjoint clusters based on similarity. The method iteratively assigns data points to clusters and refines the cluster centers to minimize the variance within each cluster. Hence, similar data points are effectively grouped together. Mathematically, given a dataset $U = \{u_i \in \mathbb{R}^m \mid i = 1, \ldots, n\}$, K-means aims to partition it into k clusters $\mathbf{C} = \{C_1, \ldots, C_k\}$ to minimize the sum of within-cluster variances:

$$\min_{\mathbf{C}} \sum_{C_l \in \mathbf{C}} \sum_{u_i \in C_l} || \, u_i - \mu_l \, ||^2, \tag{7}$$

subject to

$$\mu_l = \frac{1}{| \, C_l \, |} \sum_{u_i \in C_l} u_i, \tag{8}$$

$$C_l \cap C_p = \emptyset, l \neq p, l, p \in \{1, \cdots, k\}$$

$$\cup_{l=1}^{k} C_l = U$$

where μ_l is the centroid of cluster C_l and $|| \cdot ||$ represents the Euclidean norm.

The K-means algorithm first randomly selects k centroids from the dataset. It then calculates the distance of each data point to all chosen centroids and assigns each point to the cluster of the nearest centroid. When a data point is added to a cluster, the cluster's centroid is recalculated. This process repeats iteratively until the cluster assignments converge.

3. Schema of the Overall Approach

The proposed methodology for nonblocking modular supervisory control follows a structured four-step process, as illustrated in Figure 1. *Step 1* uses the plant models $\{\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_b\}$ and specification models $\{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_d\}$ to synthesize a group of local modular supervisors $\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_d\}$. *Step 2* augments the obtained DFAs to MDPs, according to the method elaborated in Section 4. *Step 3* applies the Q learning algorithm to learn the optimal state-action value function $Q^*(\mathbf{x}, a)(a \in A)$, which evaluates the reward value for every state and action pair $(\mathbf{x}, a)$. *Step 4* uses K-means clustering to infer the allowed event subset at the current state.

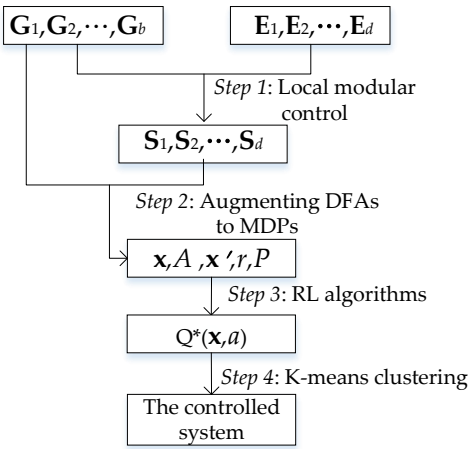


Figure 1. Schema for the proposed approach.

4. Combination of SCT, RL, and K-Means Clustering

We propose a procedure that combines SCT, RL, and K-means clustering to compute supervisory controllers that meet control requirements, ensure nonblocking property, and allow close-to-optimal freedom. Let $\mathbf{G}_i$ $(i = 1, \ldots, b)$ and $\mathbf{E}_j$ $(j = 1, \ldots, d)$ represent the DFA models of the i-th plant component and the j-th requirement, respectively. For the specification $\mathbf{E}_j$, the associated local plant is computed as $\mathbf{H}_j = \|_{i \in Share(\mathbf{E}_j)} \mathbf{G}_i$. The nonblocking local modular supervisor $\mathbf{S}_j = \langle Y_j, \Sigma_j, \delta_j, q_{j0}, Y_{jm} \rangle$ is synthesized by SCT from $\mathbf{H}_j$ and $\mathbf{E}_j$. As a result, d local modular supervisors are derived. Consequently, the controlled system under the local modular supervisors is described by a compositional DFA $\mathbf{G} = \langle \mathbf{X}, \Sigma, \Delta, \mathbf{x}_0, \mathbf{X}_m \rangle$, where $\mathbf{X}$ and $\mathbf{X}_m$ are collections of global states and global marker states, respectively.

As introduced in [19], a global state $\mathbf{x} = (q_1, q_2, \ldots, q_b, y_1, y_2, \ldots, y_d) \in \mathbf{X}$ of the system contains states of all plant components and local modular supervisors; $\mathbf{x} \in \mathbf{X}_m$ is called a global marker state if $(\forall i \in \{1, \ldots, b\}, \forall j \in \{1, \ldots, d\}) q_i \in Q_{im} \wedge y_j \in Y_{jm}$. The initial global state is $\mathbf{x}_0 = (q_{10}, \ldots, q_{b0}, y_{10}, \ldots, y_{d0})$. Define the index function $I : \Sigma \to 2^{\{1,\ldots,b\}}$ to determine the indexes of the plant components that contain a given event. For an event $\sigma \in \Sigma$, $I(\sigma) = \{i \in \{1, \ldots, b\} | \sigma \in \Sigma_i\}$. An event $\sigma \in \Sigma$ is allowed to occur at a global state $\mathbf{x}$, namely, $\Delta(\mathbf{x}, \sigma)!$, if $(\forall i \in I(\sigma)) \eta_i(q_i, \sigma)! \wedge (\forall j \in \{1, \ldots, d\}) \delta_j(y_j, \sigma)!$. The set of all allowed events at state $\mathbf{x}$ is $Enb(\mathbf{x}) = \{\sigma \in \Sigma \mid \Delta(\mathbf{x}, \sigma)!\}$. If $Enb(\mathbf{x}) = \emptyset$, then $\mathbf{x}$ is called a deadlock state. After an event $\sigma \in Enb(\mathbf{x})$ occurs at state $\mathbf{x}$, the DES reaches the next global state $\mathbf{x}' = \Delta(\mathbf{x}, \sigma) = (q_1', q_2', \ldots, q_b', y_1', y_2', \ldots, y_d')$, where $(\forall i \in I(\sigma)) q_i' = \eta_i(q_i, \sigma) \wedge (\forall i \notin I(\sigma)) q_i' = q_i \wedge (\forall j \in \{1, \ldots, d\}) y_j' = \delta_j(y_j, \sigma)$.

4.1. Augmenting DFAs to MDPs

This part constructs an MDP from the obtained DFA $\mathbf{G} = \langle \mathbf{X}, \Sigma, \Delta, \mathbf{x}_0, \mathbf{X}_m \rangle$ and computes the evaluations for all pairs of reachable state and controllable events for the following K-means clustering step. Since only controllable events can be disabled in SCT, the action set is defined as

$$A = \Sigma_c \tag{9}$$

to suit the RL framework. As introduced in Section 2.2, a supervisor of $\mathbf{G}$ decides a subset of allowed events $\Pi \in \mathcal{C}$ at every reachable state $\mathbf{x}$. To apply RL methods to obtain the event subset, we define a mapping $\pi : A \to \mathcal{C}$ that maps an action to an event subset. For an action $a \in A$, define

$$\pi(a) = \{a\} \cup \Sigma_u \tag{10}$$

If the agent chooses action $a \in A$ at a global state $\mathbf{x}$, there must be one controllable event that is permitted. As a result, each controllable event at a reachable state has a evaluation value after training. In addition, a none controllable event is allowed at this state if the evaluations of all controllable events at $\mathbf{x}$ do not satisfy the section condition. Define the function $\Psi : \mathbf{X} \times A \to 2^{\Sigma}$ as

$$\Psi(\mathbf{x}, a) = \pi(a) \cap Enb(\mathbf{x}) \tag{11}$$

to get all executable events at state $\mathbf{x}$ for action a.

Since an arbitrary event in $\Psi(\mathbf{x}, a)$ may occur, the next state is uncertain. This implies that, after the introduction of the action set A, the global DFA obtained becomes stochastic for the RL agent and can hence be considered as an MDP $\mathbf{M} = \langle \mathbf{X}, A, P, \mathbf{x}_0, r \rangle$ for the RL agent, where the state transition probability function is $P : \mathbf{X} \times A \times \mathbf{X} \to [0, 1]$. For a global state $\mathbf{x}$ and an action $a \in A$, if $\Psi(\mathbf{x}, a) = \emptyset$, then for any next global state $\mathbf{x}'$, the state transition probability is $P(\mathbf{x}, a, \mathbf{x}') = 0$. Otherwise, we derive the probability value $P(\mathbf{x}, a, \mathbf{x}')$ from the assumption that any event in the allowed event subset $\Psi(\mathbf{x}, a)$ has an equal chance to occur, i.e., $(\forall \sigma \in \Psi(\mathbf{x}, a))\ Pr(\sigma|\mathbf{x}, a) = \frac{1}{|\Psi(\mathbf{x}, a)|}$. The uniform event occurrence probability assumption is verified in [19] by a case study. The results show that this assumption does not affect the nonblocking property of the controlled system and the majority of the episode rewards based on the uniform distribution are bigger than those based on the non-uniform probability distribution.

Furthermore, a function is defined as $E : \mathbf{X} \times A \times \mathbf{X} \to 2^{\Sigma}$, such that

$$E(\mathbf{x}, a, \mathbf{x}') = \{\sigma \in \Psi(\mathbf{x}, a) | \Delta(\mathbf{x}, \sigma) = \mathbf{x}'\} \tag{12}$$

to collect the events that can reach $\mathbf{x}'$ from $\mathbf{x}$ by action a. The probability value $P(\mathbf{x}, a, \mathbf{x}')$ is the cardinality of $E(\mathbf{x}, a, \mathbf{x}')$ divided by the number of all events in $\Psi(\mathbf{x}, a)$, i.e.,

$$P(\mathbf{x}, a, \mathbf{x}') = \frac{|E(\mathbf{x}, a, \mathbf{x}')|}{|\Psi(\mathbf{x}, a)|} \tag{13}$$

Evidently, if $\mathbf{x}' \notin \Delta(\mathbf{x}, \pi(a))$, then $P(\mathbf{x}, a, \mathbf{x}') = 0$, where $\Delta(\mathbf{x}, \pi(a)) = \{\mathbf{x}' \mid \mathbf{x}' = \Delta(\mathbf{x}, \sigma), \sigma \in \pi(a)\}$.

The step reward function $r : \mathbf{X} \times A \times \mathbf{X} \to \mathbb{R}$ for the MDP is defined based on Equations (1) and (2). For a step from $\mathbf{x}$ to $\mathbf{x}'$ by action $a \in A$, $r(\mathbf{x}, a, \mathbf{x}') = 0$ if $E(\mathbf{x}, a, \mathbf{x}') = \emptyset$. Otherwise,

$$r(\mathbf{x}, a, \mathbf{x}') = \frac{1}{|E(\mathbf{x}, a, \mathbf{x}')|} \sum_{\sigma \in E(\mathbf{x}, a, \mathbf{x}')} R_e(\sigma) + R_s(\mathbf{x}') \tag{14}$$

According to (6), the optimal controller f^* for MDP $\mathbf{M} = \langle \mathbf{X}, A_o, P, \mathbf{x}_0, r \rangle$ is computed by

$$f^* = \arg\max_{f} \sum_{\mathbf{x}' \in \Delta(\mathbf{x}, \Psi(\mathbf{x}, f(\mathbf{x})))} P(\mathbf{x}, f(\mathbf{x}), \mathbf{x}')[r(\mathbf{x}, f(\mathbf{x}), \mathbf{x}') + \lambda U_f(\mathbf{x}')] \tag{15}$$

where $\Delta(\mathbf{x}, \Psi(\mathbf{x}, f(\mathbf{x}))) = \{\mathbf{x}' \mid \mathbf{x}' = \Delta(\mathbf{x}, \sigma), \sigma \in \Psi(\mathbf{x}, f(\mathbf{x}))\}$. To employ Q learning, a state-action value function $Q_f : \mathbf{X} \times A \to \mathbb{R}$ for the controller f is defined to replace U_f in the learning process. Executing an action $a \in A$ at a global state $\mathbf{x}$, the expected return is

$$\begin{aligned} Q_f(\mathbf{x}, a) &= \sum_{\mathbf{x}' \in \Delta(\mathbf{x}, \Psi(\mathbf{x}, a))} P(\mathbf{x}, a, \mathbf{x}')[r(\mathbf{x}, a, \mathbf{x}') + \lambda U_f(\mathbf{x}')] \\ &= \frac{1}{|\ \Psi(\mathbf{x}, a)\ |} \sum_{\sigma \in \Psi(\mathbf{x}, a)} R_e(\sigma) + R_s(\Delta(\mathbf{x}, \sigma) + \lambda U_f(\Delta(\mathbf{x}, \sigma)) \end{aligned} \tag{16}$$

The optimal Q function is

$$Q^*(\mathbf{x}, a) = \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x}, a)} R_e(\sigma) + R_s(\Delta(\mathbf{x}, \sigma)) + \lambda \max_{a'} Q^*(\Delta(\mathbf{x}, \sigma), a') \tag{17}$$

under the optimal control function f^*.

To evaluate and improve the learning performance, the concept of temporal difference (TD) [14] is introduced. Temporal difference methods enable the agent to learn optimal value functions incrementally by comparing estimates across successive states. Specifically, the TD error measures the discrepancy between the predicted value of the current state-action pair and the updated estimate based on subsequent states. For employing the Q learning algorithm, the temporal difference error is defined as

$$TD_{error} = \frac{1}{\mid \Psi(\mathbf{x}, a) \mid} \sum_{\sigma \in \Psi(\mathbf{x}, a)} [R_e(\sigma) + R_s(\Delta(\mathbf{x}, \sigma)) + \lambda \max_{a'} Q(\Delta(\mathbf{x}, \sigma), a')] - Q(\mathbf{x}, a) \tag{18}$$

Consequently,

$$Q(\mathbf{x}, a) \leftarrow Q(\mathbf{x}, a) + \alpha \times TD_{error} \tag{19}$$

where α denotes the learning rate and regulates the learning speed. By incorporating the temporal difference, the RL agent learns to approximate the optimal Q-function efficiently.

Algorithm 1 outlines the RL algorithm employed in this paper. Each episode begins with the initial global state $\mathbf{x}_0$ and terminates if the maximal step is arrived or no event is available at the reached state, i.e., $Enb(\mathbf{x}) = \emptyset$. Note that the second termination condition includes two situations. First, the terminal state is a marker state that qualifies a positive state reward w_1. Second, the terminal state is indeed a deadlock state and deserves a negative number w_2. The exploration–exploitation trade-off is managed using the method named $\epsilon-$greedy strategy, where $\epsilon \in (0, 1)$. At the end of each episode at Line 25, ϵ is reduced but the minimal value is 0.01. At a global state $\mathbf{x}$, the algorithm randomly selects an action $a \in A$ with probability ϵ or chooses the action $a = \arg\max_a Q^*(\mathbf{x}, a)$.

The event subset decided at $\mathbf{x}$ by the chosen action a is $\Psi(\mathbf{x}, a)$. If $\Psi(\mathbf{x}, a) = \emptyset$, meaning that the controllable event selected is not defined at $\mathbf{x}$, the state-action value is updated in Line 13, where a negative real number w_2 is assigned as a penalty for this selection. The current state is then retained as the next state to facilitate sufficient exploration during the learning process in this situation. Conversely, if $\Psi(\mathbf{x}, a) \neq \emptyset$, we consider the transitions caused by all events in $\Psi(\mathbf{x}, a)$ and update the Q function as Equation (19). In this scenario, the next state is arbitrarily chosen from the set of reachable states $\Delta(\mathbf{x}, \Psi(\mathbf{x}, a))$.

Algorithm 1 is developed for finite state MDPs, and its convergence is guaranteed according to [29]. Moreover, the reward generated in one episode, the episode number, and the average reward over the previous *Len* episodes are denoted by *EpReward*, *EpR*, and *AvR*, respectively. The iteration terminates when either *AvR* exceeds the threshold *e* or *EpNum* surpasses the maximum allowed limit *MaxEp*. At this point, the converged state-action table $Q(\mathbf{x}, a)$ is returned. To track the mean reward over the most recent *Len* episodes, an empty circular buffer *Que* of size *Len* is initialized. After each episode completes, the obtained reward is inserted into *Que*.

4.2. Online Inference by K-Means Clustering

Algorithm 1 provides the learned Q function values, where each reachable state and controllable event pair is associated with a value, $Q^*(\mathbf{x}, a)$. This section focuses on computing the final supervisor by employing the K-means clustering algorithm for online inference. The goal is to map each reachable state to a permissible event subset based on

the Q function values. At a reachable state $\mathbf{x}$, the set of Q function values for controllable events is expressed as

$$U(\mathbf{x}) = \{Q^*(\mathbf{x}, a) \mid a \in A\} \tag{20}$$

Algorithm 1: Q-learning for computing state-action values

Data: $\mathbf{M} = \langle \mathbf{X}, A_o, P, \mathbf{x}_0, r \rangle$, λ, α, ϵ, dec, e, $MaxSt$, $MaxEp$ and Len
Result: A converged state-action table $Q(\mathbf{x}, a)$

Initialize $Q(\mathbf{x}, a)$ as zeros;
$EpNum \leftarrow 0$;
$AvR \leftarrow 0$;
Declare Que as a circular queue with length $Length$;
Initialize Que as empty;
while $AvR < e$ *and* $EpNum < MaxEp$ **do**
  $EpR \leftarrow 0$;
  $\mathbf{x} \leftarrow \mathbf{x}_0$;
  $StNum \leftarrow 0$;
  while $Enb(\mathbf{x}) \neq \emptyset$ *and* $StNum \leq MaxSt$ **do**
    Choose an action $a \in A$ by ϵ-greedy policy;
    if $\Psi(\mathbf{x}, a) = \emptyset$ **then** `// Punish deadlock`
      $Q(\mathbf{x}, a) \leftarrow Q(\mathbf{x}, a) + w_2$;
      $\mathbf{x}' \leftarrow \mathbf{x}$;
      $EpR \leftarrow EpR + w_2$;
    else
      Update Q function by (19);
      Randomly select a state $\mathbf{x}'$ from $\Delta(\mathbf{x}, \Psi(\mathbf{x}, a))$;
      $EpR \leftarrow EpR + \lambda \times r(\mathbf{x}, a, \mathbf{x}')$;
    end
    $StNum \leftarrow StNum + 1$;
    $\mathbf{x} \leftarrow \mathbf{x}'$;
  end
  if $\epsilon > 0.01$ **then**
    $\epsilon \leftarrow \epsilon \times (1 - dec)$;
  end
  $EpNum \leftarrow EpNum + 1$;
  Add EpR to Que;
  $SumR \leftarrow \sum_{n \in Que} n$;
  $AvR \leftarrow SumR/len(Que)$;
end

The K-means clustering algorithm, as discussed in Section 2.4, is applied to partition $U(\mathbf{x})$ into two clusters, C_1 and C_2, such that $min(C_1) > max(C_2)$. The cluster with larger values, C_1, corresponds to the set of allowed actions at $\mathbf{x}$. This resulting action set is

$$A_1 = \{a \mid Q^*(\mathbf{x}, a) \in C_1\} \tag{21}$$

Since each action corresponds to a controllable event, A_1 is the permitted controllable event subset at $\mathbf{x}$. Finally, the allowed event subset at $\mathbf{x}$ is $(A_1 \cup \Sigma_u) \cap Enb(\mathbf{x})$. This method ensures that the permissible event set incorporates actions with high evaluation values while also considering all defined uncontrollable events.

Algorithm 2 computes the permissible event subset γ for a given state $\mathbf{x}$ based on the learned Q function. The process leverages variance analysis and K-means clustering to determine whether or not clustering is necessary and to enlarge the set of controllable events if appropriate. The algorithm requires three inputs: the current state $\mathbf{x}$, the learned Q function values $Q^*(\mathbf{x}, a)$ for all actions $a \in A$, and a predefined threshold $\delta \in [0, 1]$. The value set $U(\mathbf{x})$ is computed as Equation (20) at $\mathbf{x}$. If all values are less than zero, it indicates that all controllable events lead to deadlock states. In this case, no controllable events are permitted, and the event subset γ includes only the defined uncontrollable events. Otherwise, δ is used to determine the partition of controllable events. To eliminate manual parameter recalibration for threshold δ across datasets with differing magnitudes, values at $\mathbf{x}$ are normalized as

$$U_{norm}(\mathbf{x}) = \{\frac{Q^*(\mathbf{x}, a) - U_{min}}{U_{max} - U_{min}} \mid a \in A\} \tag{22}$$

where U_{max} and U_{min} represent the maximum and minimum of $U(\mathbf{x})$, respectively. Let u_i be the $i-$th element of $U_{norm}(\mathbf{x})$. The standard deviation σ_{norm} is calculated as

$$\sigma_{norm} = \sqrt{\frac{1}{\mid A \mid} \sum_{i=1}^{|A|} (u_i - \overline{u})^2} \tag{23}$$

where

$$\overline{u} = \frac{1}{\mid A \mid} \sum_{i=1}^{|A|} u_i \tag{24}$$

If $\sigma_{norm} < \delta$, the variance is deemed insignificant. All controllable events are permitted, and the permissible event subset includes the defined controllable and uncontrollable events at the current state. If $\sigma_{norm} \geq \delta$, K-means clustering is applied to divide $U(\mathbf{x})$ into two clusters. The actions corresponding to the cluster C_1 with larger estimation values are used to construct the permissible event subset, and the event subset γ is computed in Line 11.

Algorithm 2: Computing the permitted event subset at $\mathbf{x}$ based on $Q^*(\mathbf{x}, a)$ and K-means clustering

Data: The learned Q-function $Q^*(\mathbf{x}, a)$, a reachable state $\mathbf{x}$, and the threshold δ
Result: Permitted event subset γ

```
1  Generate U(x) = {Q*(x, a) | a ∈ A};
2  if max(U(x)) < 0 then
3  |  γ ← Enb(x) ∩ Σ_u;
4  else
5  |  Compute σ_norm by Equation (23).
6  |  if σ_norm < δ then
7  |  |  γ ← Enb(x);
8  |  else
9  |  |  Partition U(x) into C_1 and C_2 with min(C_1) > max(C_2);
10 |  |  A_1 = {a | Q*(x, a) ∈ C_1};
11 |  |  γ ← (A_1 ∪ Σ_u) ∩ Enb(x);
12 |  end
13 end
```

The threshold δ largely influences the clustering process. When δ is small, the Q-values of actions at state $\mathbf{x}$ are more likely to be divided into two distinct clusters—one

representing favorable controllable events and the other unfavorable ones. Consequently, controllable events in the unfavorable cluster are disabled. In contrast, a larger δ may lead to all controllable events being permitted, which increases the risk of violating the nonblocking property. The tuning of δ is to find the largest value that can still ensure the nonblocking property.

5. Case Studies for Quantitative Evaluations

The proposed approach is verified by two case studies related to industrial automation. To show the strengths of the new method on computational efficiency and control freedom, we compare it with two existing methods. Denoted as SCT + RL, method 1 combines SCT and RL and the RL controller chooses at most two controllable events at every reachable state [19]. The comparison shows that the new method requires similar learning time for RL but the K-means method significantly increases control freedom. Denoted as SCT, method 2 is the standard supervisory control synthesis method based on exhaustive search of finite state automata [30]. Method 2 ensures the supremal nonblocking supervisor. The comparison shows that the new approach can achieve a close-to-supremal supervisor.

5.1. Supervisory Control of a Production Transfer Line

The first case study considers a production transfer line composed of two machines, $\mathbf{M}_1$ and $\mathbf{M}_2$, a testing unit **TU**, and two intermediate product buffers, $\mathbf{B}_1$ and $\mathbf{B}_2$, as illustrated in Figure 2. Suppose that buffers $\mathbf{B}_1$ and $\mathbf{B}_2$ have storage capacities of 4 and 2, respectively.

In this system, machine $\mathbf{M}_1$ loads and processes a part, then transfers the completed part to buffer $\mathbf{B}_1$. Next, machine $\mathbf{M}_2$ retrieves a part from $\mathbf{B}_1$, processes it, and passes the finished item to buffer $\mathbf{B}_2$. The test unit **TU** picks up parts from $\mathbf{B}_2$ for inspection. If a part passes the quality test, it is discharged from the production line; otherwise, it is redirected back to $\mathbf{B}_1$ for reprocessing by $\mathbf{M}_2$. This operation introduces a feedback loop in the material flow. The system has two control requirements. (1) Buffers $\mathbf{B}_1$ and $\mathbf{B}_2$ must be protected against underflow and overflow. (2) The controlled system must not have any deadlock state.

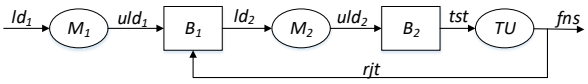
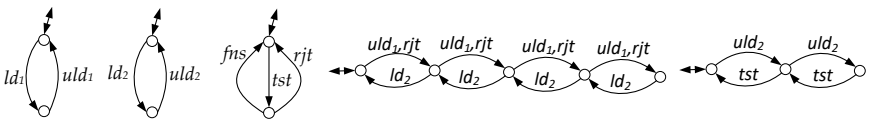


Figure 2. Illustration of the transfer line.

The models of $\mathbf{M}_1, \mathbf{M}_2, \mathbf{TU}, \mathbf{B}_1$, and $\mathbf{B}_2$ are displayed in Figure 3. The event sets of controllable and uncontrollable events are $\Sigma_c = \{ld_1, ld_2, tst\}$ and $\Sigma_u = \{uld_1, uld_2, fns, rjt\}$, respectively. The models of the two buffers are specifications, and others are plant components. The proposed method first computes two reduced local modular supervisors, $\mathbf{S}_1$ and $\mathbf{S}_2$, based on the plant components and specifications, where $\mathbf{S}_1$ corresponds to $\mathbf{B}_1$ with nine states and 51 transitions, and $\mathbf{S}_2$ corresponds to $\mathbf{B}_2$ with three states and 12 transitions. With the restriction of $\mathbf{S}_1$ and $\mathbf{S}_2$, the controlled system satisfies the specification but leads to deadlocks.



(**a**) $\mathbf{M}_1$. (**b**) $\mathbf{M}_2$. (**c**) **TU**. (**d**) $\mathbf{B}_1$. (**e**) $\mathbf{B}_2$.

Figure 3. Component DES.

For the new method, the initial global state is $\mathbf{x}_0 = (q_{1,0}, q_{2,0}, q_{3,0}, y_{1,0}, y_{2,0})$, where $q_{1,0}, q_{2,0}, q_{3,0}, y_{1,0}$ and $y_{2,0}$ are the initial states of $\mathbf{M}_1, \mathbf{M}_2, \mathbf{TU}, \mathbf{S}_1$, and $\mathbf{S}_2$, respectively. The corresponding MDP is constructed as introduced in Section 4.1, where the action space is $\{ld_1, ld_2, tst\}$.

5.1.1. Computation of the Q Function Values

Algorithm 1 is applied to compute the optimal Q function, with the hyperparameters summarized in the first line in Table 1. For instance, the used positive and negative rewards are $w_1 = 30$ and $w_2 = -50$, respectively. Figure 4 shows the plots of average rewards for the past 30 episodes of the learning processes through Algorithm 1 and the SCT + RL method in our previous work [19]. In the SCT + RL method, the RL algorithm allows at most two controllable events at every reachable state. It shows that Algorithm 1 requires 300 episodes to converge but the SCT + RL method needs more than 400 episodes. The reason for this is that the action space sizes are three and four, respectively. Consequently, Algorithm 1 has a higher training efficiency.

Table 1. Values of parameters.

Parameter	α	λ	ϵ	w_1	w_2	e	dec	$MaxSt$	$MaxEp$	Len
TL	0.9	0.96	0.98	30	−50	0.001	0.05	100	1000	30
AGV	0.95	0.99	0.95	30	−50	0.001	5×10^{-5}	100	5000	100

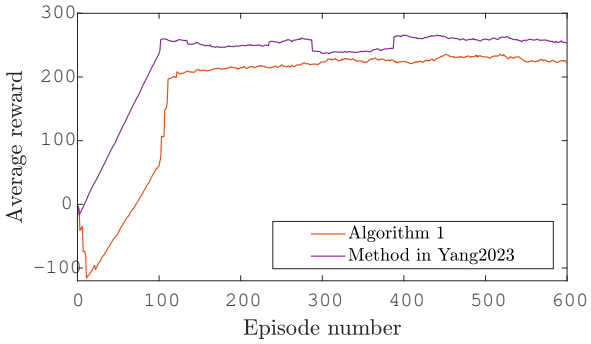


Figure 4. Plots of average rewards [19].

5.1.2. Online Inference for the Final Supervisor

After training, the Q table stores the learned values for the pairs of reachable states and controllable events are obtained. Table 2 displays the learned Q values at four reachable states. Each global state contains the state of five components: $\mathbf{M}_1, \mathbf{M}_2, \mathbf{TU}, \mathbf{S}_1$, and $\mathbf{S}_2$, respectively.

Table 2. Values of controllable events at four reachable states.

State	Controllable Events		
	ld_1	ld_2	tst
$\mathbf{x}_0$: (0,0,0,0,0)	68.77	−50	−50
$\mathbf{x}_1$: (0,0,1,5,0)	89.41	51.41	83.66
$\mathbf{x}_2$: (0,0,0,4,2)	82.89	−50	40.83
$\mathbf{x}_3$: (1,1,0,6,1)	−42.88	−43.08	−43.29

Algorithm 2 is employed to decide the allowed events based on the learned Q function. For instance, as shown in Table 2, all values of controllable events at $\mathbf{x}_3$ are negative; hence, no controllable event is allowed at this state. Since the defined uncontrolled events are uld_1 and uld_2, the final permitted event subset at $\mathbf{x}_3$ is $\{uld_1, uld_2\}$. Moreover, controllable events

are divided into $\{ld_1, tst\}$ and $\{ld_2\}$ at $\mathbf{x}_2$ according to the values, and the permitted event subset for $\mathbf{x}_2$ is $\{ld_1, tst\}$. Similarly, the determined event subset at $\mathbf{x}_0$ is $\{ld_1\}$. Since all values for controllable events at $\mathbf{x}_1$ are positive, Algorithm 2 computes $\sigma_{norm} = 0.77$. If $\delta = 0.5$, i.e., $\sigma_{norm} > \delta$, the controllable events are partitioned into $\{ld_1, tst\}$ and $\{ld_2\}$, and the decided controllable event subset is $\{ld_1, tst\}$. Since tst is not allowed by the modular supervisors, the final permitted controllable event subset is $\{ld_1\}$. If $\delta = 0.8$, all controllable events are selected, and the final permitted controllable event subset is $\{ld_1, ld_2\}$.

Figure 5 illustrates the relationship between the threshold parameter δ and the number of reachable states of the controlled system, where the number of nonblocking reachable states is 72 with $0.85 \leq \delta \leq 0.88$. The result shows that the controlled system is blocking if $\delta > 0.88$.

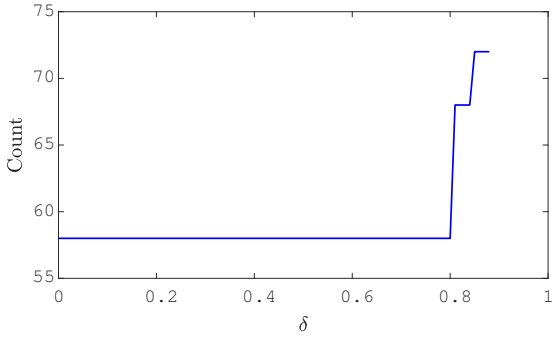


Figure 5. Variation of nonblocking reachable states with δ.

In this case, we employ $\delta = 0.88$. Table 3 presents the control results from three methods: integrating RL and SCT [19]; our new method combining RL, SCT, and K-means clustering; and directly employing SCT [31]. The results show that the SCT + RL method obtains a nonblocking supervisor of 56 reachable states. The novel SCT + RL + K-means method results in a nonblocking controlled system with 72 reachable states. For comparison, SCT is utilized to synthesize the supremal nonblocking supervisor, and the controlled system under this supervisor also achieves 72 reachable states. All three methods above obtain nonblocking supervisors. Our proposed method computes a nonblocking supervisor with equal number of reachable states as the supremal nonblocking supervisor does.

Table 3. Comparison of the control results by three methods.

Methods	SCT + RL	SCT + RL + K-Means	SCT
State space size	56	72	72
Nonblocking	Yes	Yes	Yes

5.2. Control of an Automatically Guided Vehicles System

The second application is the control of automatically guided vehicles (AGVs) [19] for a manufacturing workcell. The system consists of 15 components, including two input stations (IPS1, IPS2), three workstations (WS1, WS2, WS3), five AGVs (AGV1–AGV5), four zones (Zone1–Zone4), and the station (CPS) for completed parts.

As shown in Figure 6, five AGVs travel on fixed circular routes, transporting parts to their designated places. Figures 7 and 8 respectively display the DFA models of AGVs and specifications, where the controllable event set is

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

The models of the local modular supervisors corresponding to the specifications and the physical meanings of these events are illustrated in [19]. As a result, the controlled system under these modular supervisors has deadlocks.

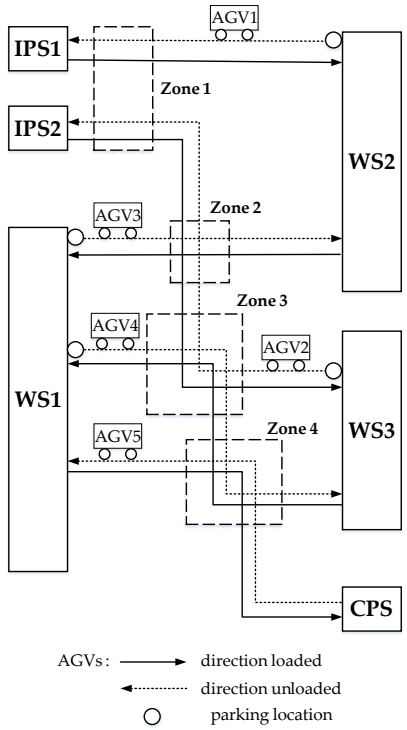
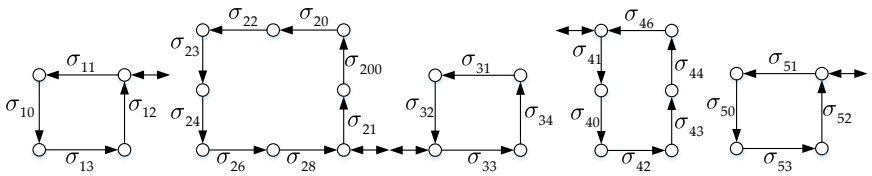
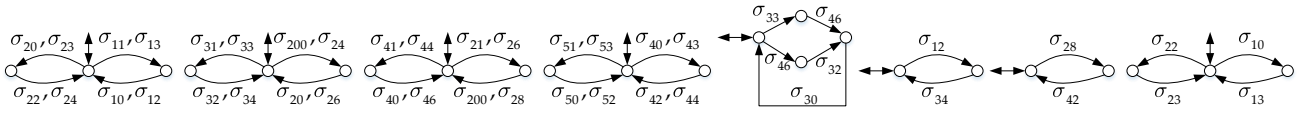


Figure 6. A manufacturing workcell with AGVs.



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

Figure 7. DFA models of AGVs.



(a) Z1. **(b) Z2.** **(c) Z3.** **(d) Z4.** **(e) WS1.** **(f) WS2.** **(g) WS3.** **(h) IPS.**

Figure 8. DFA models of specifications on AGVs.

A global state consists of the states of the five AGV models and the eight reduced modular supervisors by SCT. The constructed action set is $A = \Sigma_c$, with 10 actions. The proposed method then applies Algorithm 1 to learn the value function $Q^*(\mathbf{x}, a)$ for the MDP that corresponds to the global state transition system. Figure 9 shows the plots of average rewards for the past 100 episodes of the learning processes through Algorithm 1 and the proposed method in our previous work [19], where the parameters used are displayed in the "AGV case value" line in Table 1. Consequently, Algorithm 1 requires less than 3000 episodes, while the other needs more than 4000 episodes to get converged.

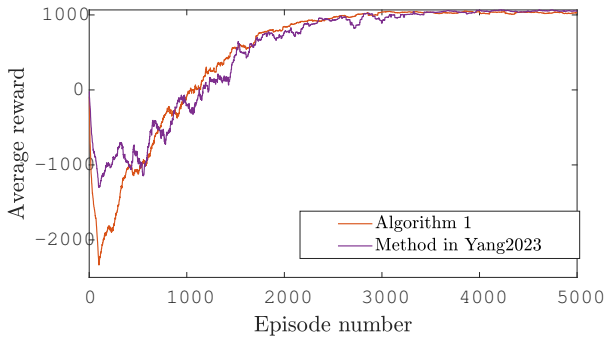


Figure 9. Plots of average rewards [19].

Algorithm 2 is then applied to infer the supervisor's policy online. Figure 10 displays the variation of nonblocking reachable states with δ. The largest number of states, 3558, is obtained when $\delta = 0.32$, and the controlled system reaches deadlock if $\delta > 0.32$. Therefore, we use $\delta = 0.32$ in this case.

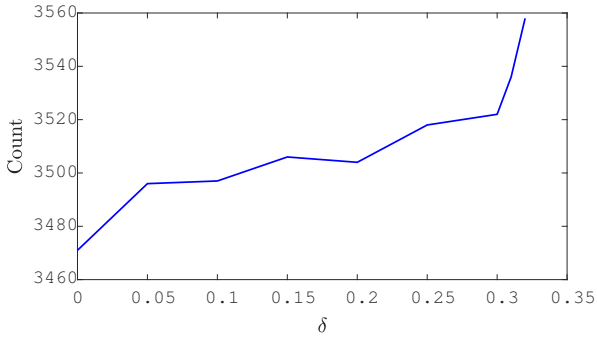


Figure 10. Variation of nonblocking reachable states with δ.

Table 4 presents the control results of the system under the method employed in the previous case. As a result, the controlled system with the SCT + RL method is nonblocking and includes 481 states, where the number of controllable events at a state is limited to, at most, two. Through the proposed method, the controlled system remains nonblocking and expands to 3558 states. Furthermore, With the supervisor by SCT, the controlled system achieves 4406 nonblocking reachable states. Compared with the method that only combines RL algorithms and SCT, the proposed method admits a much larger state size of the controlled system. Simultaneously, the satisfaction of the specifications is ensured.

Table 4. Comparison of the control results by three methods.

Methods	SCT + RL	SCT + RL + K-Means	SCT
State space size	481	3558	4406
Nonblocking	Yes	Yes	Yes

6. Conclusions

This paper presents an integrated approach for nonblocking modular supervisory control in DESs by combining SCT, RL, and K-means clustering. The proposed method enhances learning efficiency by reducing the action space of RL algorithms while improving system freedom in RL-based supervisors. Through RL, the system efficiently learns values for all state action pairs, and K-means clustering infers the allowed controllable events online to ensure a balance between permissiveness and system constraints. Experimental validation on two case studies confirms the effectiveness of the new approach, demon-

strating a significant increase in the number of nonblocking reachable states compared to prior methods.

In the future, we plan to enhance the scalability and adaptability of the approach by incorporating a Deep Q-Network (DQN). DQNs extends traditional Q-learning by using deep neural networks to approximate Q-values, enabling efficient learning in high-dimensional and large-scale discrete event systems. This will allow the system to handle more complex plant structures, larger event spaces, and dynamic environments without requiring exhaustive state enumeration. Furthermore, techniques such as experience replay and target networks [32] will be explored to stabilize training and improve convergence. Moreover, partial observability due to sensor noise, delays, or failures is one of our future works.

Author Contributions: J.Y., conceptualization, methodology, software, formal analysis, investigation, writing—original draft; K.T., software.; L.F., conceptualization, methodology, formal analysis, investigation, supervision, writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: The work is supported in part by the Youth Science and Technology Fund Project of Gansu Province, China, under Grant 25JRRA960 and the Youth Foundation of Lanzhoujiaotong University, China, under Grant 2024044. Lei Feng is financially supported by the research center of XPRES at KTH.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: 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.

References

1. Wonham, W.M.; Cai, K. *Supervisory Control of Discrete-Event Systems*; Springer: Berlin/Heidelberg, Germany, 2019.
2. Patrik, B.; Fabian, M. Calculating Restart States for Systems Modeled by Operations Using Supervisory Control Theory. *Machines* **2013**, *1*, 116–141. [CrossRef]
3. Wonham, W.M.; Ramadge, P.J. Modular supervisory control of discrete-event systems. *Math. Control. Signals Syst.* **1988**, *1*, 13–30. [CrossRef]
4. Queiroz, M.H.d.; Cury, J.E.R. Modular supervisory control of large scale discrete event systems. In Proceedings of the International Conference on Discrete Event Systems, Anchorage, AK, USA, 27 September 2000; pp. 103–110.
5. Feng, L.; Wonham, W.M. Computationally Efficient Supervisor Design: Control Flow Decomposition. In Proceedings of the IEEE 8th International Workshop on Discrete Event Systems, Ann Arbor, MI, USA, 10–12 July 2006; pp. 9–14.
6. Goorden, M.; van de Mortel-Fronczak, J.; Reniers, M.; Fabian, M.; Fokkink, W.; Rooda, J. Model properties for efficient synthesis of nonblocking modular supervisors. *Control Eng. Pract.* **2021**, *112*, 104830. [CrossRef]
7. Schmidt, K.; Marchand, H.; Gaudin, B. Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models. In Proceedings of the IEEE 8th International Workshop on Discrete Event Systems, Ann Arbor, MI, USA, 10–12 July 2006; pp. 149–154.
8. Schmidt, K.; Moor, T.; Perk, S. Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Trans. Autom. Control* **2008**, *53*, 2252–2265. [CrossRef]
9. Leduc, R.J.; Brandin, B.A.; Lawford, M.; Wonham, W.M. Hierarchical interface-based supervisory control-part I: Serial case. *IEEE Trans. Autom. Control* **2005**, *50*, 1322–1335.
10. Liu, Y.; Liu, F. Optimal control of discrete event systems under uncertain environment based on supervisory control theory and reinforcement learning. *Sci. Rep.* **2024**, *14*, 25077. [CrossRef]
11. Dai, J.; Lin, H. A learning-based synthesis approach to decentralized supervisory control of discrete event systems with unknown plants. *Control Theory Technol.* **2014**, *12*, 218–233. [CrossRef]
12. Malik, R.; Åkesson, K.; Flordal, H.; Fabian, M. Supremica–an efficient tool for large-scale discrete event systems. *IFAC-PapersOnLine* **2017**, *50*, 5794–5799. [CrossRef]
13. Farooqui, A.; Hagebring, F.; Fabian, M. MIDES: A tool for supervisor synthesis via active learning. In Proceedings of the IEEE 17th International Conference on Automation Science and Engineering, Lyon, France, 23–27 August 2021; pp. 792–797.
14. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.

15. Kajiwara, K.; Yamasaki, T. Adaptive supervisory control based on a preference of agents for decentralized discrete event systems. In Proceedings of the IEEE SICE Annual Conference, Taipei, Taiwan, 18–21 August 2010; pp. 1027–1032.
16. Zielinski, K.M.C.; Hendges, L.V.; Florindo, J.B.; Lopes, Y.K.; Ribeiro, R.; Teixeira, M.; Casanova, D. Flexible control of Discrete Event Systems using environment simulation and Reinforcement Learning. *Appl. Soft Comput.* **2021**, *111*, 107714. [CrossRef]
17. Hu, Y.; Wang, D.; Yang, M.; He, J. Integrating reinforcement learning and supervisory control theory for optimal directed control of discrete-event systems. *Neurocomputing* **2024**, *613*, 128720. [CrossRef]
18. Konishi, M.; Sasaki, T.; Cai, K. Efficient safe control via deep reinforcement learning and supervisory control-csae study on multi-robot warehouse automation. *IFAC-PapersOnLine* **2022**, *55*, 16–21. [CrossRef]
19. Yang, J.; Tan, K.; Feng, L.; Li, Z. A model-based deep reinforcement learning approach to the nonblocking coordination of modular supervisors of discrete event systems. *Inf. Sci.* **2023**, *630*, 305–321. [CrossRef]
20. Yang, J.; Tan, K.; Feng, L.; El-Sherbeeny, A.M.; Li, Z. Reducing the learning time of reinforcement learning for the supervisory control of discrete event systems. *IEEE Access* **2023**, *11*, 59840–59853. [CrossRef]
21. Ran, X.; Xi, Y.; Lu, Y.; Wang, X.; Lu, Z. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artif. Intell. Rev.* **2023**, *56*, 8219–8264. [CrossRef]
22. Kurani, A.; Doshi, P.; Vakharia, A.; Shah, M. A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting. *Ann. Data Sci.* **2023**, *10*, 183–208. [CrossRef]
23. Kriegel, H.P.; Kröger, P.; Sander, J.; Zimek, A. Density-based clustering. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2011**, *1*, 231–240. [CrossRef]
24. Bezdek, J.C.; Ehrlich, R.; Full, W. FCM: The fuzzy c-means clustering algorithm. *Comput. Geosci.* **1984**, *10*, 191–203. [CrossRef]
25. Ikotun, A.M.; Ezugwu, A.E.; Abualigah, L.; Abuhaija, B.; Heming, J. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Inf. Sci.* **2023**, *622*, 178–210. [CrossRef]
26. Feng, L.; Wonham, W.M. TCT: A computation tool for supervisory control synthesis. In Proceedings of the IEEE 8th International Workshop on Discrete Event Systems, Ann Arbor, MI, USA, 10–12 July 2006; pp. 3–8.
27. Su, R.; Wonham, W.M. What information really matters in supervisor reduction? *Automatica* **2018**, *95*, 368–377. [CrossRef]
28. Malik, R.; Teixeira, M. Optimal modular control of discrete event systems with distinguishers and approximations. *Discret. Event Dyn. Syst.* **2021**, *31*, 659–691. [CrossRef]
29. Lewis, F.L.; Vrabie, D.; Vamvoudakis, K.G. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controller. *IEEE Control Syst. Mag.* **2012**, *32*, 76–105.
30. Feng, L.; Cai, K.; Wonham, W. A structural approach to the non-blocking supervisory control of discrete-event systems. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 1152–1168. [CrossRef]
31. Wonham, W.M.; Ramadge, P.J. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim.* **1987**, *25*, 637–659. [CrossRef]
32. Saglam, B.; Mutlu, F.B.; Cicek, D.C.; Kozat, S.S. Actor prioritized experience replay. *J. Artif. Intell. Res.* **2023**, *78*, 639–672. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.