Dynamic scheduling for semiconductor manufacturing systems with uncertainties using convolutional neural networks and reinforcement learning

Juan Liu1 · Fei Qiao1 · Minjie Zou2 · Jonas Zinn2 · Yumin Ma1 · Birgit Vogel-Heuser2

Received: 8 July 2021 / Accepted: 12 October 2021 / Published online: 2 September 2022 © The Author(s) 2022

Abstract
The dynamic scheduling problem of semiconductor manufacturing systems (SMSs) is becoming more complicated and challenging due to internal uncertainties and external demand changes. To this end, this paper addresses integrated release control and production scheduling problems with uncertain processing times and urgent orders and proposes a convolutional neural network and asynchronous advanced actor critic-based method (CNN-A3C) that involves a training phase and a deployment phase. In the training phase, actor–critic networks are trained to predict the evaluation of scheduling decisions and to output the optimal scheduling decision. In the deployment phase, the most appropriate release control and scheduling decisions are periodically generated according to the current production status based on the networks. Furthermore, we improve the four key points in the deep reinforcement learning (DRL) algorithm, state space, action space, reward function, and network structure and design four mechanisms: a slide-window-based two-dimensional state perception mechanism, an adaptive reward function that considers multiple objectives and automatically adjusts to dynamic events, a continuous action space based on composite dispatching rules (CDR) and release strategies, and actor–critic networks based on convolutional neural networks (CNNs). To verify the feasibility and effectiveness of the proposed dynamic scheduling method, it is implemented on a simplified SMS. The simulation experimental results show that the proposed method outperforms the unimproved A3C-based method and the common dispatching rules under the new uncertain scenarios.

Keywords Dynamic production scheduling · Uncertainties · Deep reinforcement learning (DRL) · Convolutional neural networks (CNN) · Rule-based

Introduction
Production scheduling problem of semiconductor manufacturing systems (SMSs) is one of the most complicated scheduling optimization problems in the literature due to its inherent characteristics, such as re-entrant flow, product variety, a large number of processing steps and machines, and the capital- and technical-intensive industry [31]. There are three key decision-making phases in the manufacturing process of SMS, including planning, releasing, and production scheduling. First, enterprises make production plans according to market demands in the planning phase and assign the plans...
production system and acquire and store vast amounts of production data [22]. The use of machine learning algorithms has recently aroused increasing interest in solving the dynamic scheduling problems of SMSs by learning from data and selecting the most appropriate scheduling rule [21]. Supervised learning algorithms are used to train a scheduling model from a large number of optimal samples, that is, the mapping relationship between the production state and the optimal or approximate optimal scheduling rule. The scheduling model was then applied to the manufacturing system to realize the online selection of the optimal or approximate optimal scheduling rule. Although this kind of methodology can deal with large-scale problems and improve the real-time and adaptability of the production scheduling system, there are still several limitations as follows.

(1) The dynamic adjustment only for production scheduling cannot fully deal with the external demand changes. It is necessary to co-optimize the release control and production scheduling of SMSs, but the related research is relatively lacking.

(2) The essence of the machine learning-based dynamic scheduling method is how to select one appropriate scheduling rule according to the production status. However, a single rule focuses only on one performance criterion and no rule outperforms all others under any objective. Composite dispatching rule (CDR) adopted in this paper is the combination of multiple heuristic dispatching rules by linear weighting, which is able to consider multiple objectives.

(3) The validity of scheduling model depends on the quality of sample set to a great extent. In practice, it is difficult to obtain sufficient optimal samples, i.e., labeled samples. Most of the existing works use a traversal simulation-based method to obtain optimal samples. The method is to first traverse all scheduling rules under different states through the manufacturing system simulation model and obtain the original data, and then select the optimal data to form the final sample set. However, especially for the CDR with continuous weight parameters to be decided, the above traversal simulation-based method is not applicable. Reinforcement learning (RL) does not need labeled samples. And as deep learning has enhanced the large-scale problem-solving ability of reinforcement learning, scholars have renewed their interest in deep reinforcement learning (DRL) to solve production scheduling problems in the past 2 years [20].

This paper therefore introduces DRL to address the integrated dynamic release control and production scheduling problems of SMSs with external urgent orders and internal uncertain processing times, and proposes a convolutional neural network (CNN)- and asynchronous advanced actor critic (A3C)-based method called CNN-A3C. In this method,
the CNN network is the scheduling model with state inputs and scheduling decision outputs. The scheduling decision consists of parameters of the release strategy and weight parameters of CDRs. Therefore, the decision space is a continuous space. The A3C algorithm proposed by Mnih et al. is adept at solving problems with continuous decision spaces [17]. Thus, in the proposed method, the A3C algorithm is used to train the CNN network. The network is then applied to online scheduling to periodically output the most suitable parameters of release strategy and CDRs according to the real-time production status.

Compared to the existing studies, the main contributions of this paper can be summarized as follows.

(1) This paper studies the integrated dynamic release control and production scheduling problem of SMSs with external urgent orders and internal uncertain processing times, and first applies the combination of DRL and CNN to address this problem. We propose a CNN-A3C-based release control and production scheduling method. In this method, the A3C algorithm is used to train the CNN network as the scheduling model with state inputs and scheduling decision outputs.

(2) To make the A3C algorithm more efficient on scheduling problem, we further improve the four key points in the A3C algorithm, state space, action space, reward function, and network structure and design four mechanisms: a slide-window-based two-dimensional state perception mechanism, an adaptive reward function that considers multiple objectives and automatically adjusts to dynamic events, a continuous action space based on CDR and release strategies, and actor–critic networks based on CNN.

(3) The proposed dynamic scheduling method is achieved to automatically generate the most appropriate release control and scheduling decision according to the current production status, which are successfully applied in a simplified SMS.

The remainder of this paper is organized as follows. The section “Literature review” provides a literature survey of machine learning-based dynamic scheduling approaches, especially DRL-based approaches. The section “Problem definition and modeling” describes the dynamic scheduling problem formulation based on RL. The section “CNN-A3C dynamic release control and production scheduling framework” proposes the CNN-A3C-based dynamic scheduling framework. The section “CNN-A3C-based dynamic release and scheduling implementation” introduces the state space, action space, reward function, and actor–critic network design. The section “Experimental results and analysis” presents and discusses the experimental results. Finally, the section “Conclusion” concludes the paper and briefly explores directions of future work.

**Literature review**

The literature review outlined in the following text focuses on dynamic scheduling using machine learning and existing approaches that consider RL for applications in dynamic production scheduling. The drawbacks of existing research are further analyzed in several aspects and compared with the proposed methods.

**Dynamic production scheduling based on machine learning**

In machine learning-based dynamic scheduling approaches, the scheduling problem can be described as a 3-tuple \( (F, D, P) \), where \( F \) is the set of complete production attributes of a manufacturing system and represents the production state, \( D \) is a set of scheduling decisions, and \( P \) is the performance after one scheduling period using the decision \( D \) under the production state \( F \). This method aims to establish a map from the production state \( F \) to the optimal scheduling decision \( D^* \) that meets the optimal performance evaluation criteria \( P^* \). This mapping is also called the scheduling knowledge or scheduling model. Based on this, an approximate optimal scheduling decision can be found to meet a better system performance evaluation criterion under a given production state. Figure 1 shows the general scheduling knowledge training process. As shown in Fig. 1, the general data-driven scheduling approach takes three steps [21]. In the first step, the state feature selection, the key production state features (SF) that are related to scheduling are selected to improve the scheduling efficiency. Since the historical data are not all optimal, the second step aims to select the optimal samples as the labeled samples. Afterward, the third step obtains the optimal samples and mines the scheduling knowledge using a machine learning algorithm.

Although this kind of method can improve the real time and adaptability of the production scheduling system, the following deficiencies remain. It is difficult to obtain sufficient labeled samples from real historical production data [21]. In addition, to some extent, the training efficiency and scheduling knowledge effectiveness depend excessively on the accuracy of feature selection. When debugging the scheduling knowledge mining in step 3, it is often unavoidable to go back to redoing or adjusting the first two steps.

**Dynamic production scheduling based on DRL**

RL has also been actively applied to solve dynamic scheduling problems to overcome the above defects. RL is a kind of machine learning concerned with how agents ought to take actions in an environment to maximize the cumulative reward. First, different from supervised learning, RL does not need the optimal sample selection procedure. Second, RL is
characterized by a sequential decision-making ability. That is, at each scheduling point, the status at the next scheduling point and the final performances are considered, while the supervised learning-based scheduling method can only optimize the performance after one scheduling period.

RL can be classified as policy-based RL (e.g., Policy Gradient) and value-based RL (e.g., q-learning). The policy-based RL trains a probability distribution by sampling strategies and enhances the probability of the action with high return value being selected. The value-based RL is to obtain a value table and select the action with high value according to it. The value of action refers to the expected return reward of action. However, when the problem is very complex and has an infinite number of states and actions, we cannot store the value in a table. Thus, scholars use neural network to fit value tables and probability distributions. It is the essential idea of DRL [13]. The introduction of deep learning (DL), represented by a deep neural network (DNN) with perception makes the state feature selection no longer a necessary step and can solve large-scale scheduling problems.

One of the earliest application studies of RL on production scheduling was from Zhang [33], who proposed a policy gradient-based method to learn domain-specific heuristics for job shop scheduling. In recent years, DRL has received much attention and has recently been employed to solve the dynamic scheduling problem [23]. Table 1 reviews the existing methods for DRL-based dynamic scheduling in chronological order, and summarizes the differences between the aforementioned works and our work.

**State-space design**

As shown in Table 1, there are two main state forms determined by the state perception method. One is the detailed state of each job and machine; for example, Waschnec et al. used the position of each job and the state of each machine as the state [28]. The second is the production attributes, such as the number of workpieces in processing (WIP) and the buffer queue length. In the work from Shiue et al., 30 system attributes were used to describe the state [23]. The first form was a more detailed and comprehensive modeling of the system state. However, it has the obvious drawback of being unable to handle uncertain problems with varying numbers of jobs. The second form could overcome the above drawback, but detailed information was lost [16]. Therefore, it is necessary to study a state perception method that can not only obtain enough detailed information but also address changes in the job count.

**Action space design**

As shown in Table 1, there are two main action forms. The job to be operated indicates which job the machine chooses to operate. The SDR (single dispatching rule), MDR (multiple dispatching rule), and CDR (composite dispatching rule) are all dispatching rule-based methods. The SDR indicates that the machines in the system all adopt the same dispatching rule. The MDR indicates that the machines in the system adopt different dispatching rules [23]. The CDR is the combination of multiple heuristic dispatching rules by the linear weighting method and is able to consider multiple objectives. Dispatching rules are broadly applied to solve real-world optimization problems when the computation time is limited and the problem size is large. Some common heuristic dispatching rules are as follows [17]: first in first out (FIFO), shortest remaining processing time (SRPT), earliest due date (EDD), and critical ratio (CR). The sequence of operations to be processed is determined by the priorities calculated by the dispatching rule. It is well known that a dispatching rule focuses on only one performance criterion, and no rule outperforms all others under any objective.

**Reward function design**

As shown in Table 1, most works adopted the fixed reward function that indicates an unchangeable reward calculation procedure. Several works have been conducted on designing the adaptive reward function to improve the adaptability of the reward function. For example, Waschnec et al. designed a two-phase reward function and proposed a multiagent DQN-based global scheduling approach. They divided the training process into two phases. Only one DQN agent was trained in the first phase, and the other machines were controlled by
Table 1: Existing methods on DRL-based dynamic scheduling

| Reference | Application                                      | Uncertainty                     | Objectives | Algorithm | Detailed design state design | Action design | Reward function | Network structure |
|-----------|--------------------------------------------------|---------------------------------|------------|-----------|------------------------------|---------------|-----------------|-------------------|
| [6]       | Job shop scheduling                              | None                            | Single     | Policy gradient | Position of each job         | SDR           | Fixed           | ANN               |
| [10,25]   | Order dispatching problem                        | None                            | Single     | Q-learning   | Production attributes        | Select a job to be operated | Fixed           | ANN               |
| [28]      | Global scheduling                                | Stochastic processing time      | Single     | Multiagent DQN | State of each job and each machine | MDR           | Adaptive        | ANN               |
| [23]      | Real-time production scheduling                  | Stochastic arrival of jobs      | Multiple   | Q-learning   | Production attributes        | MDR           | Fixed           | SOM (self-organizing map network) |
| [14]      | Smart manufacturing scheduling                   | None                            | Single     | DQN         | Production attributes        | MDR           | Fixed           | ANN               |
| [20]      | Multichip products scheduling                    | Stochastic processing time      | Single     | Multiagent Q-learning | Production attributes | Job to be operated | Fixed           | ANN               |
| [16]      | Flexible production scheduling                   | New job insertions              | Single     | DQN         | Production attributes        | SDR           | Fixed           | ANN               |
| [8]       | Flexible production scheduling                   | Stochastic arrival of jobs      | Single     | DQN         | Petri net model-based        | SDR           | Fixed           | GCN (graph convolutional network) |
| [1]       | Semiconductor production scheduling              | Random arrival of orders        | Single     | DQN         | Detailed state of orders and machines | Select an order to be operated | Fixed           | ANN               |
|           | Our proposed method                              | Dynamic release controlling and production scheduling | Urgent order and stochastic processing time | Multiple | A3C | Slide-window based state | CDR | Adaptive | CNN |

[Note:] DQN represents the Deep Q-learning algorithm. A3C represents the asynchronous advanced actor critic algorithm. They are both deep reinforcement learning. SDR represents the single dispatching rule method, MDR represents the multiple dispatching rule method, and CDR represents the composite dispatching rule method. ANN represents the artificial neural network, SOM represents the self-organizing map network, GCN represents the graph convolutional network, and CNN represents the convolutional neural network.
heuristics. In the second phase, all machines were controlled by DQN agents that learned separately. To satisfy the different demands of these two training phases, they designed a two-phase reward function [28]. The reward function needs to be adjustable according to the production state to adapt to the demand changes.

Network structure

As shown in Table 1, the artificial neural network (ANN) is adopted in most related works, and generally to fully connected neural network. In these works, the state space is generally one-dimensional. Several works have begun to design complex state space to improve situational awareness. The network architecture is also improved. For example, Hu et al. [8] model the flexible manufacturing system using Petri nets and then employ a graph convolutional network (GCN). GCN is a kind of CNN. Compared with traditional CNN, it can deal with unstructured input. CNN architecture is very important for its performance [30]. The typical structures include Lenet-5 [12], AlexNet [9], GoogLeNet [26], VGG-Nets [24], ResNets [7], etc. In addition to the above hand-crafted CNN structure, many scholars focus on the neural architecture search method. For example, Xue et al. [30] propose a self-adaptive mutation neural architecture search algorithm. Xue et al. [29] study a multi-objective evolutionary approach for neural architecture search to design accurate CNNs with a small structure. O’neill et al. [18] design a genetic algorithm to discover skip-connection structures on DenseNet networks. Recently, CNNs have achieved great success in the field of speech recognition, image recognition, and natural language processing [11]. However, the application research in scheduling field is still relatively lacking.

Most related works adopted ANNs that indicate the fully connected neural networks. Several works have been conducted on designing the adaptive reward function to improve the adaptability of the reward function. For example, Waschneck et al. designed a two-phase reward function and proposed a multiagent DQN-based global scheduling approach. They divided the training process into two phases. Only one DQN agent was trained in the first phase, and the other machines were controlled by heuristics. In the second phase, all machines were controlled by DQN agents that learned separately. To satisfy the different demands of these two training phases, they designed a two-phase reward function [28]. The reward function needs to be adjustable according to the production state to adapt to the demand changes.

Research on the application of DRL to production scheduling problems is still in the exploratory stage [19]. The task-specific features and object-related parameters limited the universal method framework. However, some common points can also be found. As seen from the first five columns, most recent works employed deep Q-learning (DQN) due to the rule-based discrete action space. As mentioned above, no single rule outperforms all others under any objective. This paper uses a composite dispatching rule (CDR) with weights, so that the action space becomes a continuous space. On the other hand, the scheduling problem with continuous action space is incompatible with DQN. The asynchronous advanced actor–critic (A3C) proposed by Mnih et al. can operate over continuous action spaces, constituting an actor–critic, model-free algorithm based on the deterministic policy gradient [17]. Therefore, this paper employs A3C for the dynamic production scheduling. Unlike the existing DQN-based scheduling methods, whose decision is the single dispatching rule, this paper improves the A3C to generate the composite dispatching rule to solve the dynamic scheduling problem of SMS.

Motivated by the above-mentioned remarks, we present a CNN-A3C-based dynamic release control and production scheduling framework. Based on this, a slide-window-based state perception mechanism is first designed, so that the state contains enough detailed information and is able to handle varying numbers of jobs. A state with a two-dimensional spatial structure is observed, which can be successfully handled by deep convolutional neural networks (CNNs). Second, the action space is designed as a combination of composite dispatching rule (CDR)-based continuous scheduling actions and release strategy-based discrete releasing actions. To deal with this combined action space, we improve A3C to training networks and design an action decoding system that converts the output actions into decisions that can be executed by the SMS. Third, we also proposed an adaptive reward function that considers multiple objectives and automatically adjusts to dynamic events. Finally, the proposed CNN-A3C method is verified on a semiconductor manufacturing line benchmark in terms of various performance criteria, on-time delivery date rate (ODR), and mean cycle time (MCT). It is proven that the proposed method outperforms the dispatching rule-based method and the other DRL methods through comparative experiments under uncertain processing times and urgent orders.

Problem definition and modeling

Dynamic release control and production scheduling problem definition

Let \( I_0 \) denote the scheduling problem with a certain environment. There are \( o \) orders of production in a semiconductor system, denoted as \( O_1, O_2, \ldots, O_o \). Each order has one type of product, and the type and the number of products may vary between orders. \( N_o \) and \( D_o \) represent the product count and the due date of order \( O_o \), respectively. There
are \( m \) machines and \( M \) workstations in the SMS. Production activities are performed according to the release strategy of the system and the CDR of each machine in this paper. And the CDRs of machines in the same workstation are the same. Thus, the aim of \( I_0 \) is to find a set of optimal parameters of release strategy and CDR, described as \( a^* = \{a_1^0, a_2^0, a_1^1, a_2^1, \ldots, a_k^1, \ldots, a_1^M, a_2^M, \ldots, a_k^M \} \). The first two bits \( \{a_1^0, a_2^0\} \) are parameters of release strategy, and the other bits represent the production scheduling decision. The detailed explanation is described in section “Action space design”.

However, uncertainties always occur during the production process. The urgent order and uncertain processing times of jobs are considered in this paper. Let \( I \) denotes the dynamic scheduling problem. \( u \) represents the urgent order, and \( G_u, N_u, \) and \( D_u \) represent the generation time, product count, and due date, respectively. The goal of dynamic scheduling is to adjust parameters of the release strategy and CDRs in real time according to the current state to protect production performances from the occurrence of disturbances. The dynamic scheduling problem \( I \) add the concept of time. The production state at time \( t \) is denoted as \( s_t \). The goal of \( I \) is to decide the most suitable \( a_t \) according to the current state \( s_t \) in real time.

Assumptions are made of this SMS as follows. And the used notations are defined in Appendix Table 8.

- Each order has one type of product, and the type of product may vary between orders.
- Different types of products have the same processing flow but different processing times.
- Not all jobs are available at the initial time. The arrival of jobs is determined by the release control strategy.
- All machines are available at the initial time and never break down.
- The operations of a job are independent of each other. The operations of one job are independent of those of other jobs.
- No semifinished product is scrapped.
- No job needs to be reworked.
- Urgent orders may be inserted during the production process.
- The processing times of operations are uncertain, and their expected values are known.

**Dynamic release control and production scheduling problem modeling based on RL**

In the RL-based scheduling approach, the problem is described as a Markov decision process with a 5-tuple representation \(<S, A, P, \gamma, R>\) [16]. \( S \) is the state space, \( A \) is the action space, \( P \) is the state transition probability, \( \gamma \) is the discount factor, and \( R \) is the reward. In the Markov decision process, the next state \( s_{t+1} \) of the system is only related to the current state \( s_t \) and state transition probability \( P \), and has nothing to do with the past state \( s_{t-1} \). The RL agent interacts with the SMS following a policy \( \pi \), which is a mapping from \( S \) to \( A \), \((S \rightarrow A)\), as shown in Fig. 2. In this paper, the policy is a neural network.

As shown in Fig. 2, \( t \) represents the scheduling point. There are \( T \) scheduling points in this production process. In general, the value of \( T \) is large; the value of the scheduling period is so small that the scheduling decision-making is real time. At each decision point \( t \), the RL agent chooses an action \( a_t \) according to the current state \( s_t \), after which the state changes into a new state \( s_{t+1} \) and receives an immediate reward \( r_t \in R \). The objective of the dynamic scheduling RL agent is to find a policy \( \pi^* \) that maximizes the expected sum of long-term rewards, as shown in Eq. (1), wherein \( \gamma \) is the discount factor

\[
\pi^* := \max_{\pi} E \left( \sum_{t=1}^{T} r_t \cdot \gamma^t \right).
\] (1)

In addition, at each scheduling point, the decision \( a_t \) is required to respond to the current changes in the environment and to take into account the long-term overall performance criterion. Therefore, the dynamic scheduling problem can be described as Eq. (2)

\[
I = \left\{ \begin{array}{ll}
\operatorname{opt} p (\pi (a_t | s_t)) & \\
\text{s.t. } a_t \in A, s_t \in S, t \in T,
\end{array} \right.
\] (2)

where there are \( T \) scheduling points in the product process, \( a_t \) represents the scheduling decision (the composite dispatching rule) at the \( t \)th scheduling point, \( a_t \in A \), and \( \pi (a_t | s_t) \) is a neural network and determines \( a_t \) according to the current state \( s_t \) at each scheduling point, \( s_t \in S \). Thus, the purpose of dynamic scheduling is to obtain the optimal policy \( \pi (a_t | s_t) \) that meets the optimal performance \( p \). \( p \) represents the accumulated performance criteria over the whole production process \( T \).

As can be seen from Eqs. (1) and (2), the immediate reward \( r_t \) needs to reflect production performance, so the design of the reward function is particularly important. In addition, the design of the state space and action space are also two key tasks in this RL-based dynamic scheduling problem.

**CNN-A3C dynamic release control and production scheduling framework**

This section proposes a CNN-A3C dynamic scheduling framework that consists of the actual manufacturing system in the physical space and the manufacturing system simula-
tion module, action decoding module, training module, and deployment module in the cycle space, as shown in Fig. 3.

(1) The manufacturing system simulation module has $N$ simulation models that provide the environment interacting with subnetworks. In this paper, these simulation models are established based on the discrete event simulation method. In addition to the basic function of reflecting the production logic, they also need to be able to transmit the production state and performance index data to the subnetwork and execute the output scheduling strategy of the subnetwork in real time.

(2) The training module is the crucial part of this method and involves a global network and $N$ subnetworks. Each subnetwork has the same network structure as the global network. Each subnetwork interacts with the environ-
The deployment module is the online application of the trained network. The interaction of each subnet is implemented as a thread. These N threads do not interfere with each other and run independently.

(3) The action decoding system translates the action from the network into a schedule that can be executed by the production system directly.

(4) The deployment module is the online application of the trained network. The function, input, and output of these modules are shown in Table 2.

### CNN-A3C-based dynamic release and scheduling implementation

Based on the above dynamic scheduling framework, this paper further focuses on the training module and proposes a CNN-A3C-based dynamic scheduling method. The three crucial elements in the application of A3C, state space, action space, and reward function are designed in this section.

#### State-space design

This paper designs a two-dimensional state representation that consists of three parts: machine-related, order-related, and job-related, as shown in Fig. 4. The number of columns in the state is constant and equal to seven. Each row has seven elements which are production attributes, as shown in Table 9. Among them, there are six production attributes in the machine-related part. The last column of this part is filled with zeros. As shown in Fig. 4, the machine-related part has \(m\) rows, and \(m\) is the machine count. The order-related part has \(o + 1\) rows, \(o\) is the number of existing orders, and the other part refers to a new order that may come. The job-related part comprises detailed job-related information, and the row count should be equal to the job count. The total number of jobs is variable due to the occurrence of the urgent order. Thus, the number of rows in the state varies with the number of jobs.

To ensure a constant state dimension, we present a production state perception mechanism based on the sliding window that mainly focuses on the state of jobs in the processing stage. The row count of the job-related part is equal to the maximum capacity of the production line. The production state perception mechanism based on the sliding window is shown in Fig. 4.

In a smart manufacturing system, the state of each job can be tracked and recorded through sensor and communication technologies. In the case of this paper, radio-frequency identification (RFID) technology is used to track and manage jobs. Each job has an RFID tag that is scanned before and after every operation. We can therefore obtain the state data of the jobs. Then, the finished jobs are sorted by finishing time, and the jobs in the processing are sorted by entering time. As shown in Fig. 4, the job to right has an earlier finishing time or a late entering time. At each sampling point, the perception window with a width of \(C\) slides to the far left of the job flow. \(C\) is the maximum capacity of the manufacturing system, which is equal to the sum of the capacity of all buffers and equipment.

#### Action space design

The action consists of two parts, \(a_t = \{a_0^0, a_0^2, a_1^1, a_2^1, \cdots, a_k^1, \cdots, a_1^M, a_2^M, \cdots, a_k^M\}\). As shown in Fig. 5, the first two bits \(\{a_0^0, a_0^2\}\) are parameters of the release strategy, and the other bits represent the production scheduling decision. In this paper, the rule CONWIP is used as the release strategy [2], where \(a_0^0\) indicates the maximum value of jobs in processing (WIP) and determines the release speed, and \(a_0^2\) represents the release proportion of orders. The second part \(\{a_1^1, a_2^1, \cdots, a_k^1, \cdots, a_1^M, a_2^M, \cdots, a_k^M\}\) is divided into \(M\) sections. Taking \(\{a_1^1, a_2^1, \cdots, a_k^1\}\) as an example, the scheduling decision of workstation 1 can be represented; each element is the weight of the single dispatching rule, and \(k\) is the number of selected dispatching rules in the CDR.

The procedure of action decoding and real-time scheduling execution is as follows, where \(Q_i\) is the priority of job
Fig. 4 Production state perception mechanism based on the sliding window

$i$ and $q_{1,i}, q_{2,i}, \ldots, q_{k,i}$ represent the priority calculated by the single dispatching rule.

**Reward function design**

According to the analysis in section Reward “function design”, the reward function needs to be adaptable to changes in demand at different phases of production, i.e., it can be adjusted according to the production state. In this paper, urgent orders and uncertain processing times are considered in the release control and production scheduling problems. During the arrival of urgent orders, optimization goals are skewed towards ODR of urgent orders. Thus, we divide the production process into two types of phase, urgent order arrival phase and no urgent order phase. As shown in Algorithm 2, in the proposed adaptive reward mechanism, rewards are calculated differently at different phases.

$r_i$ is calculated by comparing the simulation output performance measures with their standard values marked with *. In this study case, the system’s objectives are to maximize the ODR of urgent orders, minimize the MCT, and ensure that the productivity (PROD), average movement (AvgMOV) and flow time do not degrade. Thus, the simulation output performance measures include $uTP, TP$, and $WT$. Wherein, $uTP$ is the number of finished jobs of the urgent order in this scheduling period, $TP$ is the total number of finished jobs in this scheduling period, and $WT$ is the maximum value of the waiting time of WIP.

If the urgent order has been placed and not finished, the current production is in the urgent order arrival phase. The reward $r_i$ is then calculated according to $uTP$ and $WT$ as shown in Algorithm 2. If an urgent order is not placed or completed, the current production is in the no urgent order
Algorithm 1: Action decoding and real-time scheduling

**Input**: The action \( a \) outputted by the RL network

**Output**: Scheduling result

Compute the maximum WIP value using

\[
W = a^0 \cdot C
\]

while this scheduling period is not over

\[
\text{do}
\]

Compute the current \( w_{\text{ip}} \)

if \( w_{\text{ip}} \leq W - (o + 1) \) then

\[
\text{Compute the priorities of waiting jobs using } Q_i = a^1 q_{1,i} + a^2 q_{2,i} + \ldots + a^k q_{k,i}
\]

else

Release \( o + 1 \) jobs for the urgent order

end if

end do

phase. The reward \( r_t \) is then calculated according to \( TP \) and \( WT \) as shown in Algorithm 2.

**Algorithm 2**: Adaptive reward function with ODR, PROD and CT criteria

**Input**: Performance measures of the one-step scheduling period

**Output**: The immediate reward \( r_t \)

\[
r_t = 0 \\
\text{if the urgent order has been placed and not finished then}
\]

if \( uTP > uTP^* \) then

\[
r_t = 1 \\
\text{else if } uTP < uTP^*_L \text{ then}
\]

\[
r_t = -1 \\
\text{else}
\]

\[
r_t = 0 \\
\text{end if}
\]

else

if \( TP > TP^*_L \) then

\[
r_t = 1 \\
\text{else if } TP < TP^*_L \text{ then}
\]

\[
r_t = -1 \\
\text{else}
\]

\[
r_t = 0 \\
\text{end if}
\]

end if

Compute the maximum value of waiting times of WIP, WT

if \( WT > WT^*_H \) then

\[
r_t = 1 \\
\text{end if}
\]

Acting–critic network architecture

The neural network of A3C algorithm agent is called acting-critic network, which essentially contains an Actor network and a Critic network. The input of Actor network is state, and the output is action. The input of Actor network is state, and the output is the evaluation value of the state. To reduce the difficulty of training, Actor network and Critic network are generally adopted to share part of network structure and parameters. In this section, acting–critic network based on convolutional neural network is designed. This paper designs the Actor–Critic network according to the Lenet-5 [12] network structure. As shown in Fig. 6, the Actor–Critic network contains four convolution layers (C1-C4), two fully con-
connected layers (FC1 and FC2), and the output layer that is divided into action output and value output.

**CNN-A3C-based scheduling network training**

**Experimental results and analysis**

This section evaluates the performance of the proposed method, i.e., the CNN-A3C dynamic scheduling method, denoted as CNN-A3C here, in terms of the ODR, MCT, PROD, AvgMOV, and Flow time. To validate the effects of the proposed state, action, and reward design methods, the first experiment focuses on the training phase to analyze the performances of our proposed CNN-A3C method and the other three A3C-based methods denoted as A3C, R-A3C, and SN-A3C. The second experiment focuses on the deployment phase to compare the environmental adaptability of our proposed CNN-A3C method with the A3C and rule-based methods.

**Experimental scenarios and parameter setting**

The proposed method is evaluated on the semiconductor smart manufacturing demonstration unit system that our research group established according to the benchmark Minifab proposed in [27]. As shown in Fig. 7, the top part is the physical photograph, and the bottom part is the top view. This system contains five machines, three workstations, two buffers, and one robot. The diffusion workstation consists of two parallel and batching diffusion machines, Ma and Mb. The ion implantation workstation has two parallel ion implantation machines, Mc and Md. The lithography workstation has one lithography machine Me. There are three types of products: product_a, product_b, and product_c, running in this system. Each product type includes six processing steps. In addition, this demonstration system is equipped with a reliable sensor network based on RFID, OLE for Process Control (OPC), and industrial Internet technologies to enable real-time data transactions. Its reliable execution unit ensures the precise execution of the schedule.

Based on this Minifab system, we extended six experimental scenarios to verify the proposed method, as shown in Table 3. Scenario 2 is prepared to train the networks and verify the proposed state, action, and reward design effects in the CNN-A3C dynamic scheduling approach. The other scenarios are designed to test the adaptability of our approach in terms of the variabilities in the urgent order and processing times.

**Effectiveness of the CNN-A3C on the training phase**

This section analyzes the effectiveness of the proposed state design, action design, and adaptive reward function on the CNN-A3C method by comparing the training performances with the other three A3C-based methods. The contrastive methods are described in Table 4. A3C represents the conventional A3C-based dynamic scheduling method that uses the existing production attribute-based state space, fixed reward function, and ANN network architecture. SN-A3C represents the A3C-based method using the proposed adaptive reward function. R-A3C represents the A3C-based method using the proposed slide-window-based state space and CNN networks.

In this study case, the parameter settings of these four methods are detailed in Tables 11, 12, 13, and 14. A detailed description of A3C can be found in [17].

The networks are trained for 30,000 episodes on a deep learning workstation equipped with an Intel 12-core 3.5-GHz CPU, a GTX1080TI GPU, and 16-GB memory. The performance curves of the training phase are shown in Fig. 8,
where the x and y coordinates of each point represent the cumulative number of episodes and the performance criteria, respectively. For quantitative comparison, we calculate the average performance criteria of the last 300 episodes, as shown in Table 5.

ODR_\text{a}, ODR_\text{b}, and ODR_\text{u} are the on-time delivery rates of order \text{a}, order \text{b} and urgent order, respectively. ODR_\text{X} of an order \text{X} is the ratio of the number of finished products before the due date of the order \text{X} to the total product count of the order \text{X}. MCT represents the mean cycle time of all the jobs. PROD represents the average daily productivity. AvgMOV represents the average daily movement. Flow time represents the total processing time. For MCT and flow time, the lower value is the better. For the others, the higher value is the better.

First, the A3C and the R-A3C methods are compared. The R-A3C adopts the proposed state space and CNN network structure on the basis of the A3C method. It can be seen from Table 5 that the convergence values of the two methods are almost equal in each performance index. As can be seen from Fig. 8, the performances of the R-A3C converge after approximately 8,000 episodes, while the A3C method requires 20,000 episodes to converge. The proposed state perception mechanism is conducive to the overall perception of production state. The two additions prove to be quite beneficial in training phase.

Secondly, the A3C and the SN-A3C methods are compared. The SN-A3C adopts the proposed adaptive reward function on the basis of the A3C method. As can be seen from Table 5, the value of ODR_\text{u} of the A3C method is 0.69, while

### Table 3 Description of experimental scenarios (/min)

| No. | $N_a$ | $D_a$ | $N_b$ | $D_b$ | Dynamic events | $G_a$ | $G_b$ | $\varepsilon$ |
|-----|-------|-------|-------|-------|----------------|-------|-------|-------------|
| 1   | 100   | 21000 | 100   | 22000 | 25 [5000,6440] | 8000,8400 | 2.5% |
| 2   | 100   | 21000 | 100   | 22000 | 50 [5000,6440] | 13000,14440 | 5%  |
| 3   | 100   | 21000 | 100   | 22000 | 75 [5000,6440] | 18000,19440 | 7.5% |
| 4   | 200   | 40000 | 200   | 40000 | 25 [5000,6440] | 8000,8440  | 2.5% |
| 5   | 200   | 40000 | 200   | 40000 | 50 [5000,6440] | 13000,14440 | 5%  |
| 6   | 200   | 38000 | 200   | 40000 | 75 [5000,6440] | 18000,19440 | 7.5% |

[Note]: $N_a$ and $N_b$ represent the product counts of order \text{a} and order \text{b}. $D_a$ and $D_b$ represent the due dates of order \text{a} and order \text{b}. Similarly, $N_u$, $G_u$, and $G_b$ represent the number, generation time, and due date of the urgent order of product \text{c}. In addition, $\varepsilon$ is the parameter of uncertain processing time of step 3. The actual processing time is uniformly distributed $[PT - PT \cdot \varepsilon, PT + PT \cdot \varepsilon]$, wherein $PT$ is the expected processing time of step 3

### Table 4 Descriptions of contrastive methods

| Method     | State space      | Action space | Reward function | Network structure |
|------------|------------------|--------------|-----------------|------------------|
| CNN-A3C    | Slide- window-based | Continuous   | Adaptive        | CNN              |
| R-A3C      | Slide- window-based | Continuous   | Fixed           | CNN              |
| SN-A3C     | Production attributes | Continuous | Adaptive        | ANN              |
| A3C        | Production attributes | Continuous | Fixed           | ANN              |
Fig. 8 Training results of the subnetwork 2(best agent). a ODR_u, b ODR_a, c ODR_b, and d MCT

Table 5 Training results of the last 300 episodes’ method

| Method   | ODR_u | ODR_a | ODR_b | MCT (min) | PROD (/day) | AvgMOV (/day) | Flow time (min) |
|----------|-------|-------|-------|-----------|-------------|---------------|----------------|
| CNN-A3C  | 1.00  | 0.96  | 1.00  | 1164.06   | 16.307      | 97.85         | 22075.51       |
| SN-A3C   | 0.99  | 0.99  | 1.00  | 2208.36   | 16.312      | 97.87         | 22074.32       |
| R-A3C    | 0.70  | 1.00  | 1.00  | 2290.67   | 16.311      | 97.87         | 22071.96       |
| A3C      | 0.69  | 0.99  | 1.00  | 2284.68   | 16.311      | 97.87         | 22071.19       |

the value of it of the SN-A3C method is 0.99. The SN-A3C significantly improves the ODR_u. In terms of the other performance criteria, the differences are so small that they can be negligible. As can be seen from Fig. 8, the performances of the A3C converge after approximately 20,000 episodes, while the SN-A3C method requires 27,000 episodes to converge. Therefore, the complexity of reward function will slow down the convergence rate, and the proposed adaptive reward function can optimize the ODR of the urgent order ensure the other performances.

Finally, we compare the CNN-A3C method with the other three methods to verify the effectiveness of the combination of all improvements. Through the comparative analysis of the above two groups, we have discussed the effectiveness and disadvantages of each improvement. As shown in Fig. 8, the performances of the CNN-A3C method converge after approximately 8,000 episodes. Compared with the other three methods, it has faster convergence rate. As can be seen from Table 5, in terms of MCT, the value of the CNN-A3C method is 1164.06, while the values of the others are all over 2200. method is 0.99. The SN-A3C significantly improves the ODR_u. In terms of ODR_u, In terms of the other performance criteria, the differences are so small that they can be negligible. Therefore, the CNN-A3C method can optimize the ODR of the urgent order and MCT and ensure the other performances when uncertainties of urgent orders and stochastic processing times occur.
Environmental adaptability on the deployment phase

To further examine the adaptability of the CNN-A3C method, this section applies the network trained in scenario 2 to the other new scenarios. The proposed method is compared with the conventional A3C method and the rule-based method FIFO.

To avoid randomization of the results, we conducted 30 repeat experiments. Table 6 shows the mean results of 30 repeat tests. The last column shows the values of comprehensive performance, which are calculated by normalizing each performance value and then summing them. In addition, larger values indicate better performances. For more clarity, the comprehensive performance data in the table are illustrated in Fig. 9 in the form of a histogram. Table 6 and Fig. 9 show that the CNN-A3C method is still effective in new scenarios and is superior to the other two methods in terms of multi-objective optimization.

Furthermore, the improvement rates of the CNN-A3C and A3C methods over the rule-based method are calculated and shown in Table 7. It can be seen from Table 7 that CNN-A3C could significantly improve the ODR of the urgent order and MCT. For instance, in scenario 4, CNN-A3C improves ODR_u and MCT by 14.81% and 52%, respectively, compared to the rule-based method. In addition, CNN-A3C has a small attenuation in other performance criteria. For example, in scenario 3, CNN-A3C reduces the ODR_b, PROD, AvgMOV, and flow time by 2.13%, 0.68%, 0.68%, and 0.68%, respectively, compared to the rule-based method. However, as shown in the last column of Table 7, CNN-A3C outperforms the A3C and rule-based methods in terms of the mean improvement.

Table 6 Results in the new scenarios of 30 repeat tests

| Scenario No. | Method | ODR_u | ODR_a | ODR_b | MCT (min) | PROD (/day) | AvgMOV (/day) | Flow time (min) | Comprehensive performance |
|--------------|--------|-------|-------|-------|-----------|-------------|---------------|----------------|--------------------------|
| 1            | CNN-A3C | 0.96  | 1.00  | 1.00  | 1160.51   | 16.55       | 99.32         | 19573.29        | 5.93                     |
|              | A3C    | 0.96  | 1.00  | 1.00  | 2344.20   | 16.55       | 99.32         | 19573.18        | 5.04                     |
|              | Rule   | 0.68  | 1.00  | 0.94  | 2009.34   | 100.31      | 99.32         | 19380.04        | 4.45                     |
| 2            | CNN-A3C | 0.84  | 0.96  | 0.97  | 2477.78   | 16.55       | 99.32         | 24565.17        | 5.99                     |
|              | A3C    | 0.97  | 0.54  | 0.94  | 24565.17  | 96.72       | 99.32         | 2400.48         | 5.58                     |
|              | Rule   | 0.79  | 0.96  | 0.79  | 2400.48   | 101.38      | 99.32         | 2374.73         | 5.58                     |
| 3            | CNN-A3C | 0.96  | 1.00  | 1.00  | 1140.04   | 16.92       | 101.51        | 24565.17        | 5.99                     |
|              | A3C    | 0.96  | 1.00  | 1.00  | 2353.22   | 16.92       | 101.51        | 24565.03        | 5.08                     |
|              | Rule   | 0.67  | 0.96  | 0.79  | 2374.73   | 17.02       | 102.12        | 2374.73         | 4.34                     |
| 4            | CNN-A3C | 0.93  | 1.00  | 0.96  | 1158.35   | 16.75       | 100.53        | 39675.95        | 5.28                     |
|              | A3C    | 0.93  | 1.00  | 0.96  | 1158.35   | 16.75       | 100.53        | 39675.95        | 5.28                     |

Fig. 9 Test results in other new scenarios
| Scenario No. | Method   | Improvement over rule−based method (%) | ODR_u | ODR_a | ODR_b | MCT | PROD | AvgMOV | Flow time | Mean |
|-------------|---------|--------------------------------------|-------|-------|-------|-----|------|--------|-----------|------|
| 1           | CNN−A3C | 41.18                                | 0     | 51.65 | −1.02 | −0.99 | −1.00 | 12.83  |
|             | A3C     | 41.18                                | 0     | 2.34  | −1.02 | −0.99 | −1.00 | 5.79   |
| 3           | CNN−A3C | 22.78                                | −2.13 | 51.61 | −0.68 | −0.68 | −0.68 | 10.67  |
|             | A3C     | 6.33                                 | −13.83| 2.29  | −0.68 | −0.68 | −0.67 | 0.55   |
| 4           | CNN−A3C | 14.81                                | 3.09  | 52.00 | −0.48 | −0.46 | −0.46 | 9.79   |
|             | A3C     | 13.58                                | 1.12  | 1.14  | −0.42 | −0.46 | −0.45 | 2.35   |
| 5           | CNN−A3C | 12.82                                | 3.37  | 51.99 | −0.36 | −0.38 | −0.38 | 9.74   |
|             | A3C     | 11.54                                | 7.87  | 1.36  | −0.36 | −0.36 | −0.36 | 2.33   |
| 6           | CNN−A3C | 12.82                                | 3.37  | 51.99 | −0.36 | −0.36 | −0.36 | 9.74   |
|             | A3C     | 11.54                                | 7.87  | 1.36  | −0.36 | −0.36 | −0.36 | 2.33   |

**Conclusion**

This paper studied the dynamic release control and production scheduling problem of SMSs while considering uncertainties from the internal and external environment and proposed a CNN-A3C-based approach. We provide the following conclusions based on the results of this work. The CNN-A3C method is able to optimize the ODR of the urgent order and the MCT and improves the overall performance of the manufacturing system under the disturbances of urgent orders and stochastic processing times. On the other hand, the trained network is still effective in the new scenarios.

In the future, because the proposed approach cannot respond to changes in the machine count, it would be of interest to improve the network structure and introduce multiagent technology. In addition, the complete realization of the demonstration unit indicates that the proposed method has the potential for further application in real industry.

**Declarations**

**Funding** This work was supported in part by the National Key R&D Program of China under Grant No. 2018AAA0101704, in part by the National Natural Science Foundation of China under Grant No. 71690230/71690234, 61973237, 61873191, and in part by the China Scholarship Council scholarship.

**Conflict of interest** All of the authors declare that they have no conflict of interest.

**Availability of data and materials** The data used to support the findings of this study are available from the corresponding author upon request.

**Code availability** The code is available from the corresponding author upon request.

**Ethics approval** This paper does not contain any studies with human participants or animals performed by any of the authors.

**Consent to participate** Not applicable.

---

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

**Appendix**

See Tables 8, 9, 10, 11, 12, 13 and 14.
### Table 8 Notations in this paper

| Parameters of the SMS |  
|-----------------------|---------------------------------------------------------------|
| \( o \)              | The number of orders                                          |
| \( O_o \)            | The \( o_b \) order                                          |
| \( N_o \)            | The product count of order \( O_o \)                         |
| \( D_o \)            | The due date of order \( O_o \)                              |
| \( m \)              | The number of machines                                        |
| \( M \)              | The number of workstations                                   |
| \( C \)              | The maximum capacity of the manufacturing system             |

| Uncertainty-related parameters of the CNN-A3C |  
|-----------------------------------------------|---------------------------------------------------------------|
| \( u \)                                       | The urgent order                                              |
| \( G_u \)                                     | The generation time of the urgent order \( u \)               |
| \( N_u \)                                     | The product count of the urgent order \( u \)                 |
| \( D_u \)                                     | The due date of the urgent order \( u \)                      |
| \( PT \)                                      | The expected processing time of operation \( i \)             |
| \( \varepsilon \)                              | The parameter of uncertain processing time                    |

| State-related parameters of the CNN-A3C       |  
|-----------------------------------------------|---------------------------------------------------------------|
| \( s_t \)                                     | The production state at time \( t, s \in S \)                |

| Action-related parameters of the CNN-A3C      |  
|-----------------------------------------------|---------------------------------------------------------------|
| \( a_t \)                                     | The action at time \( t, a_t = [a_1^t, a_2^t, a_1^t, a_2^t, \ldots, a_k^M, a_2^M, \ldots, a_k^M], a_t \in A \) |
| \( [a_1^t, a_2^t] \)                           | The parameters of the release strategy                         |
| \( [a_1^M, a_2^M, \ldots, a_k^M] \)           | The parameters of CDR on workstation \( M \)                  |
| \( Q_i \)                                     | The priority of job \( i \) calculated by CDR                 |
| \( q_{k,i} \)                                 | The priority of job \( i \) calculated by the \( k \) time dispatching rule |
| \( k \)                                       | The number of dispatching rules in the CDR                     |
| \( W \)                                       | The maximum WIP                                               |

| Reward-related parameters of the CNN-A3C      |  
|-----------------------------------------------|---------------------------------------------------------------|
| \( r_t \)                                     | The immediate reward at time \( t, r_t \in R \)               |
| \( WT \)                                      | The maximum value of the waiting times of WIP                 |
| \( WH^* \)                                    | The \( WT \)-related parameters of reward                     |
| \( uTP \)                                     | The number of finished jobs of the urgent order in one scheduling period |
| \( uTP^* \), \( uTP^*_L \)                    | The \( uTP \)-related parameters of reward                     |
| \( TP \)                                      | The total number of finished jobs in one scheduling period     |
| \( TP^*_H \), \( TP^*_L \)                    | The \( TP \)-related parameters of reward                      |

| Other parameters of the CNN-A3C               |  
|-----------------------------------------------|---------------------------------------------------------------|
| \( \gamma \)                                  | The discount factor in A3C algorithm                           |
| \( P \)                                       | The state transition probability                               |
| \( N \)                                       | The number of subnetworks in A3C algorithm                     |
Table 9  Parameter settings of the state space based on the slide-window method

| Category        | State attributes                                                                 |
|-----------------|-----------------------------------------------------------------------------------|
| Machine-related | The workstation ID Processing State (Idle/Occupied) Running State (Normal/Failure/Repair) The ID of jobs on the machine The number of jobs on the machine The queue length |
| Order-related   | The product ID The number of the order The due date The number of released jobs The proportion of released jobs The number of finished jobs The proportion of finished jobs |
| Job-related     | The product ID Online time The current step The due date The waiting time of the current step The remaining time The total waiting time |

Table 10  Parameter settings of the state space based on production attributes

| No. | State attributes                        |
|-----|----------------------------------------|
| 1   | The WIP on the machine Ma              |
| 2   | The WIP on the machine Mb              |
| 3   | The WIP on the machine Mc              |
| 4   | The WIP on the machine Md              |
| 5   | The WIP on the machine Me              |
| 6   | The queue length of the buffer Mab_B   |
| 7   | The queue length of the buffer Mcd_B   |
| 8   | The queue length of the buffer Me_B    |
| 9   | The WIP of the order a                 |
| 10  | The WIP of the order b                 |
| 11  | The WIP of the order u                 |
| 12  | The total WIP                          |
| 13  | The number of jobs finished Step 4 of order a |
| 14  | The number of jobs finished Step 4 of order b |
| 15  | The number of jobs finished Step 4 of order c |
| 16  | The number of the order a              |
| 17  | The number of the order b              |
| 18  | The number of the order u              |
| 19  | The proportion of finished jobs of the order a |
| 20  | The proportion of finished jobs of the order b |
| 21  | The proportion of finished jobs of the order u |

Table 11  Parameter setting of the A3C method

| Category | Parameters | Setting/value |
|----------|------------|---------------|
| State    | Size       | 21            |
|          | Detailed information | Table 10 |
| Action   | Size       | 17            |
|          | System capacity | 30            |
|          | CDR        | EDD, SRPT, CR, FIFO, SPT |
| Reward   | Just related with TP |               |
|          | $TP_{II}^*$ | 2             |
|          | $TP_{IL}^*$ | 4             |
| Category                  | Parameters | Setting/value |
|---------------------------|------------|---------------|
| Network structure         | Input      | 21            |
|                           | FC1        | 128           |
|                           | FC2        | 128           |
|                           | FC3        | 64            |
|                           | FC4        | 64            |
|                           | FC5        | 128           |
|                           | Output1    | 1             |
|                           | Output2    | 17            |
|                           | Output3    | 17            |
| Hyper-parameter           | Activation function | ReLU         |
|                           | Optimizer  | Adam          |
|                           | Learning rate | 0.0001    |
|                           | Batch size | 1000          |
|                           | Max episode | 50000        |

**Table 12** Parameter setting of the SN-A3C method

| Category                  | Parameters | Setting/value |
|---------------------------|------------|---------------|
| State                     | Size       | 21            |
|                           | Detailed information | Table 10 |
| Action                    | Size       | 17            |
|                           | System capacity | 30          |
|                           | CDR        | EDD, SRPT, CR, FIFO, SPT |
| Reward                    | $u T P_H^*$  | 3             |
|                           | $u T P_L^*$  | 2             |
|                           | $T P_H^*$   | 4             |
|                           | $T P_L^*$   | 2             |
|                           | $W T_H^*(\min)$ | 1440         |
| Network structure         | Input      | 21            |
|                           | FC1        | 128           |
|                           | FC2        | 128           |
|                           | FC3        | 64            |
|                           | FC4        | 64            |
|                           | FC5        | 128           |
|                           | Output1    | 1             |
|                           | Output2    | 17            |
|                           | Output3    | 17            |
| Hyper-parameter           | Activation function | ReLU         |
|                           | Optimizer  | Adam          |
|                           | Learning rate | 0.0001    |
|                           | Batch size | 1000          |
|                           | Max episode | 50000        |
Table 13 Parameter setting of
the R-A3C method

| Category       | Parameter | Setting/value |
|----------------|-----------|---------------|
| State          | Size      | 38 × 7        |
| Detailed info  |           |               |
| Action         | Size      | 17            |
|                | C         | 30            |
|                | CDR       | EDD, SRPT, CR, FIFO, SPT |
| Reward         |           |               |
|                | Just related with $TP$ |
|                | $TP^*_H$  | 4             |
|                | $TP^*_L$  | 2             |
| Network structure |         |               |
| Input          |           | 38 × 7        |
| Conv1          | $K=4$, $F=3$, $S=1$, $P=“same”$ |
| Conv2          | $K=4$, $F=3$, $S=1$, $P=“same”$ |
| Conv3          | $K=8$, $F=1$, $S=1$, $P=“same”$ |
| Conv4          | $K=4$, $F=1$, $S=1$, $P=“valid”$ |
| FC1            | 1024      |
| FC2            | 128       |
| Output1        | 1         |
| Output2        | 17        |
| Output3        | 17        |
| Hyper-parameter | $N$       | 4             |
|                | Activation function | ReLU |
|                | Optimizer  | Adam          |
|                | Learning rate | 0.0001       |
|                | Batch size  | 1000          |
|                | Max episode | 30000         |

[Note]: In the network structure, $K$ represents the number of filters. $F$ is the spatial extent. $S$ is the stride. $P$ represents the type of padding algorithm to use.

Table 14 Parameter setting of
the CNN-A3C method

| Category       | Parameter | Setting/value |
|----------------|-----------|---------------|
| State          | Size      | 38 × 7        |
| Detailed info  |           |               |
| Action         | Size      | 17            |
|                | C         | 30            |
|                | CDR       | EDD, SRPT, CR, FIFO, SPT |
| Reward         |           |               |
|                | $uTP^*_H$ | 3             |
|                | $uTP^*_L$ | 2             |
|                | $TP^*_H$  | 4             |
|                | $TP^*_L$  | 2             |
|                | $WT^*_H$  | 1440          |
| Network structure |         |               |
| Input          |           | 38 × 7        |
| Conv1          | $K=4$, $F=3$, $S=1$, $P=“same”$ |
| Conv2          | $K=4$, $F=3$, $S=1$, $P=“same”$ |
Table 14 continued

| Category | Parameter | Setting/value |
|----------|-----------|---------------|
| Conv3    |  \( K = 8, \; F = 1, \; S = 1, \; P = \text{"same"} \) |
| Conv4    |  \( K = 4, \; F = 1, \; S = 1, \; P = \text{"valid"} \) |
| FC1      | 1024 |
| FC2      | 128 |
| Output1  | 1 |
| Output2  | 17 |
| Output3  | 17 |
| Hyper-parameter |  \( N = 4 \) |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.0001 |
| Batch size | 1000 |
| Max episode | 30000 |

[Note]: In the network structure,  \( K \) represents the number of filters.  \( F \) is the spatial extent.  \( S \) is the stride.  \( P \) represents the type of padding algorithm to use

References

1. Altenmüller T, Stüker T, Waschneck B, Kuhnle A, Lanza G (2020) Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints. Prod Eng 14(3):319–328. https://doi.org/10.1007/s11740-020-00967-8
2. Bahaji N, Kuhl ME (2008) A simulation study of new multi-objective composite dispatching rules, CONWIP, and push lot release in semiconductor fabrication. Int J Prod Res 46(14):3801–3824. https://doi.org/10.1080/00207540460711879 http://www.tandfonline.com/doi/abs/10.1080/00207540460711879
3. Bhatt N, Chauhan, NR (2015) Genetic algorithm applications on Job Shop Scheduling Problem: A review. In: 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), pp. 7–14 https://doi.org/10.1109/ICSCTI.2015.7489556
4. Chen Y , Zhou H, Huang P, Chou F, Huang S (2019) A refined order release method for achieving robustness of non-repetitive dynamic manufacturing system performance. Ann Oper Res https://doi.org/10.1007/s10479-019-03484-9 http://link.springer.com/10.1007/s10479-019-03484-9
5. Chiang TC (2013) Enhancing rule-based scheduling in water fabrication facilities by evolutionary algorithms: Review and opportunity. Comput Ind Eng 64(1):524–535. https://doi.org/10.1016/j.cie.2012.08.009 https://linkinghub.elsevier.com/retrieve/pii/S0360835212002082
6. Gabel T, Riedmiller M (2012) Distributed policy search reinforcement learning for job-shop scheduling tasks. Int J Prod Res 50(1):41–61. https://doi.org/10.1080/00207543.2011.571443 http://www.tandfonline.com/doi/abs/10.1080/00207543.2011.571443
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. Las Vegas, USA (2016). https://doi.org/10.1109/CVPR.2016.90 https://ieeexplore.ieee.org/document/7780459
8. Hu L, Liu Z, Hu W, Wang Y, Tan J, Wu F (2020) Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. J Manuf Syst 55:1–14. https://doi.org/10.1016/j.jmsy.2020.02.004 https://linkinghub.elsevier.com/retrieve/pii/S0278612520300145
9. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun. ACM 60(6):84–90. https://doi.org/10.1145/3065386 https://doi.org/10.1145/3065386
10. Kuhnle A, Schäfer L, Stricker N, Lanza G (2019) Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems. Procedia CIRP 81:234–239. https://doi.org/10.1016/j.procir.2019.03.041 https://linkinghub.elsevier.com/retrieve/pii/S221282719303464
11. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539 https://www.nature.com/articles/nature14539
12. Lecun Y, Bottou L, Bengio Y, Haﬀner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791 https://ieeexplore.ieee.org/document/726791
13. Li, H., Kumar, N., Chen, R., Georgiou, P.: A deep reinforcement learning framework for identifying funny scenes in movies. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3116–3120 (2018). https://doi.org/10.1109/ICASSP.2018.8462686 https://ieeexplore.ieee.org/document/8462686
14. Lin CC, Deng DJ, Chih YL, Chiu HT (2019) Smart Manufacturing Scheduling With Edge Computing Using Multiclass Deep Q Network. IEEE Trans Ind Inform 15(7):4276–4284. https://doi.org/10.1109/TII.2019.2908210 https://ieeexplore.ieee.org/document/8676376/
15. Lowe JJ, Mason SJ (2016) Integrated Semiconductor Supply Chain Production Planning. IEEE Trans Semicond Manuf 29(2):116–126. https://doi.org/10.1109/TSM.2016.2544202 http://ieeexplore.ieee.org/document/7436801/
16. Luo S (2020) Dynamic scheduling for ﬂexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput 91:106208. https://doi.org/10.1016/j.asoc.2020.106208 https://linkinghub.elsevier.com/retrieve/pii/S1872595220301484
17. Mnih V, Badia A, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning pp. 1928–1937
18. O’Neill D, Xue B, Zhang M (2021) Evolutionary neural architecture search for high-dimensional skip-connection structures on
densenet style networks. In: IEEE Transactions on Evolutionary Computation. https://doi.org/10.1109/TEVC.2021.3083315

19. Palombarini JA, Martínez EC (2019) Closed-loop Rescheduling using Deep Reinforcement Learning. IFAC-PapersOnLine 52(1):231–236. https://doi.org/10.1016/j.ifacol.2019.06.067 https://linkinghub.elsevier.com/retrieve/pii/S2405896319301521

20. Park, I.B., Huh, J., Kim, J., Park, J.: A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities. IEEE Trans Autom Sci Eng pp. 1–12 (2020). https://doi.org/10.1109/TASE.2019.2956762. https://ieeexplore.ieee.org/document/8946870/

21. Priore P, Gómez A, Pino R, Rosillo R (2014) Dynamic scheduling of manufacturing systems using machine learning: An updated review. Artif Intell Eng Des Anal Manuf 28(1):83–97. https://doi.org/10.1017/S0890060413000516https://www.cambridge.org/core/product/identifier/S0890060413000516/type/journal_article

22. Rossit DA, Tohmé F, Frutos M (2019) A data-driven scheduling approach to smart manufacturing. J Ind Inf Integr 15:69–79. https://doi.org/10.1016/j.jii.2019.04.003 https://linkinghub.elsevier.com/retrieve/pii/S2452414X18300475

23. Shiu YR, Lee KC, Su CT (2018) Real-time scheduling for a smart factory using a reinforcement learning approach. Comput Ind Eng 125:604–614. https://doi.org/10.1016/j.cie.2018.03.039 https://linkinghub.elsevier.com/retrieve/pii/S036083521830130X

24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Comput Sci (2014). arXiv:1409.1556pdf

25. Stricker N, Kuhnle A, Sturm R, Friess S (2018) Reinforcement learning for adaptive order dispatching in the semiconductor industry. CIRP Ann 67(1):511–514. https://doi.org/10.1016/j.cirp.2018.04.041 https://linkinghub.elsevier.com/retrieve/pii/S0007850618300659

26. Szegedy C, Wei L, Jia Y, Sermanet P, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9. Boston, USA. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7298594

27. Vargas-Villamil F, Rivera D, Kempf K (2003) A hierarchical approach to production control of reentrant semiconductor manufacturing lines. IEEE Trans Control Syst Technol 11(4):578–587. https://doi.org/10.1109/TCST.2003.813368http://ieeexplore.ieee.org/document/1208336/