SCIENCE CHINA
Technological Sciences

Special Topic: Robotics and Intelligent Manufacturing
•Article•

July 2023 Vol.66 No.7: 1937–1951
https://doi.org/10.1007/s11431-022-2413-5

Digital twin-enabled adaptive scheduling strategy based on deep reinforcement learning

GAN XueMei[1], ZUO Ying[2*], ZHANG AnSi[3], LI ShaoBo[3] & TAO Fei[2]

[1] *School of Mechanical Engineering, Guizhou University, Guiyang 550025, China;*
[2] *School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China;*
[3] *State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China*

Received November 26, 2022; accepted April 26, 2023; published online June 14, 2023

The modern complicated manufacturing industry and smart manufacturing tendency have imposed new requirements on the scheduling method, such as self-regulation and self-learning capabilities. While traditional scheduling methods cannot meet these needs due to their rigidity. Self-learning is an inherent ability of reinforcement learning (RL) algorithm inhered from its continuous learning and trial-and-error characteristics. Self-regulation of scheduling could be enabled by the emerging digital twin (DT) technology because of its virtual-real mapping and mutual control characteristics. This paper proposed a DT-enabled adaptive scheduling based on the improved proximal policy optimization RL algorithm, which was called explicit exploration and asynchronous update proximal policy optimization algorithm (E2APPO). Firstly, the DT-enabled scheduling system framework was designed to enhance the interaction between the virtual and the physical job shops, strengthening the self-regulation of the scheduling model. Secondly, an innovative action selection strategy and an asynchronous update mechanism were proposed to improve the optimization algorithm to strengthen the self-learning ability of the scheduling model. Lastly, the proposed scheduling model was extensively tested in comparison with heuristic and meta-heuristic algorithms, such as well-known scheduling rules and genetic algorithms, as well as other existing scheduling methods based on reinforcement learning. The comparisons have proved both the effectiveness and advancement of the proposed DT-enabled adaptive scheduling strategy.

agent, reinforcement learning, digital twin, adaptive scheduling, generalization

Citation: Gan X M, Zuo Y, Zhang A S, et al. Digital twin-enabled adaptive scheduling strategy based on deep reinforcement learning. Sci China Tech Sci, 2023, 66: 1937–1951, https://doi.org/10.1007/s11431-022-2413-5

1 Introduction

As production environments have become more complex and variable, new requirements are put forward for production management, especially for allocating limited resources. An optimized allocation of limited resources demands an adaptive scheduling strategy, which is capable of self-learning and self-regulation. The job shop scheduling problem (JSSP) has attracted extensive research in engineering and academic fields. JSSP is a combinatorial optimization problem, which is usually solved by precise and approximate algorithms.

Precise algorithms are mainly based on operational research methods, including mathematical programming [1], lagrangian relaxation technique [2], branch and bound [3], etc. These methods can obtain theoretically optimal solutions, and are suitable for small or simple scheduling problems. But they do not apply to practical production due to their over-accurate modeling, rigidity, and massive calculation for complex problems.

Approximate algorithms are also widely employed in scheduling, such as priority rules and meta-heuristic algo-

*Corresponding author (email: yingzuo@buaa.edu.cn)

rithms. Priority rules with their rapid response are more applicable to specific scheduling scenarios. They tend to be ineffective when the resources in the task pool change. As for the meta-heuristic algorithms applied to scheduling, there are many swarm intelligence algorithms based on characteristics of biological population performance, e.g., genetic algorithm [4,5], particle swarm optimization [6], ant colony algorithm [7], artificial bee colony algorithms [8,9], the novel heuristic electromagnetism-like mechanism algorithm [10], and gray wolf algorithm [11]. Meta-heuristic algorithms solve specific problems and perform well through knowledge of relevant behaviors, functions, rules, and mechanisms in biological, physical, chemical, and artistic. However, they involve a long time to recalculate when conditions change.

These above optimization algorithms are unable to adapt to new environments through self-updating. When the production state changes, especially the scheduling scale, the scheduling strategy needs to be recalculated. The self-learning of scheduling becomes feasible with the emergence of reinforcement learning (RL), an optimization algorithm that simulates continuous interaction and learning between the agent and the environment [12]. Through continuous trial-and-error and experience accumulation, the optimal scheduling strategy will be generated with self-learning ability and strong generalization characteristics. The self-regulation ability depends on the awareness of the discrepancy between the real-time processing state and the predicted one. The key to self-regulation is how to obtain real-time resource information in the physical job shop and feed it back to the virtual scheduling, which is difficult to realize in traditional manufacturing. But the emerging digital twin (DT) technology makes the self-adjustment of scheduling possible through the virtual-real mutual interaction nature.

With the motivations above, this paper developed a digital twin-enabled adaptive scheduling strategy based on RL to cope with the complicated industry environment and changeable order. The primary contributions of this paper are mainly listed as follows. (1) DT-enabled adaptive scheduling system framework is built to facilitate the interaction between the virtual and physical job shops to adapt the complex production. (2) The explicit exploration and asynchronous update proximal policy optimization algorithm (E2APPO) is designed with an innovative search strategy and asynchronous update mechanism, bringing higher exploration efficiency and more stable network updates. (3) Numerical experiments demonstrate the effectiveness of the proposed DT-enabled adaptive scheduling strategy.

The remainder of this paper is as follows. A brief review of scheduling based on RL is presented in Section 2, as well as the related research of DT technology. Section 3 describes the job shop scheduling and the mathematical model. Section 4 designs the DT-enabled adaptive scheduling methodology. Section 5 proposes the design details of the improved E2APPO algorithm, which plays a vital role in the proposed DT-enabled scheduling. Section 6 presents the numerical comparison experiments, which prove the advancement of the proposed E2APPO algorithm within the DT-enabled scheduling system framework. Finally, the conclusions and future work are drawn in Section 7.

2 Literature review

Traditional scheduling strategies, which are only applicable to specific scenarios because of their rigidity, show deficiencies in complicated manufacturing environments. The development of RL and DT technology makes it possible to design an adaptive scheduling strategy.

2.1 RL-based scheduling

To cope with the insufficient self-learning ability of the traditional scheduling methods, many scholars have used RL to train scheduling models for the increasingly complicated industry environment and the changeable order. RL is an unsupervised learning algorithm without requiring the preparation of label data in advance. It has unique advantages in scenarios where it is difficult to collect and obtain label data. The job shop can be seen as a similar scenario, where the agent selects an operation to be machined according to the current state.

To the best of our knowledge, the application of RL on scheduling can be mainly classified into four different action space types of RL, as represented in Table 1. Firstly, RL is combined with a meta-heuristic algorithm, the parameter pool is designed as the action space, and the algorithm performance is improved by selecting the algorithm's optimal parameters [13–15]. For example, Emary et al. [13] developed an improved gray wolf optimization method combined with a neural network named experienced gray wolf optimization (EGWO). They designed the exploration rate as action space to learn the optimal parameter, and demonstrated that EGWO outperforms the previous algorithm in all performance metrics. Secondly, RL is applied in combination with priority rules, and the rules pool is designed as the action space. The agent finds the best rule at each decision-making point [16,17]. For example, Lin et al. [17] provided the DQN method and edge computing to solve the JSSP, and the single priority rule or compound rule is taken as an action set so that the agent learns how to assign rules to different machines. Thirdly, the operations of the workpiece are designed as the action space, and the agent directly selects the operation at each decision-making point [18–21]. For example, Xia et al. [20] applied a Q-network for semiconductor manufacturing and defined an action set including all fea-

Table 1 Scheduling based on RL for four different action space types

Action space type	Work	Type of problem	Algorithm	Objective
Choose parameter	Emary et al. [13], Shahrabi et al. [14], Zhao and Zhang [15]	DJSS/JSSP	RL and heuristics	Performance /makespan/ tardiness
Choose scheduling rule	Stricker et al. [16], Lin et al. [17]	JSSP	Q-learning/ D3QN	Makespan/tardiness
Choose operation	Shi et al. [18], Palombarini and Martínez [19], Xia et al. [20], Park et al. [21]	JSSP	Q-learning /deep-Q/DQN /Q-network	Makespan/tardiness
Choose machine	Zhou et al. [22]	DJSS	Q-learning	Mean flow time

sible operations. Experiments showed the superiority of the proposed method on makespan. Lastly, machine ID or transferring material is defined as action space for an agent to choose [22].

The above literature suggests that continuous learning and trial-and-error of RL provide a new idea for the adaptive scheduling scheme. However, there are still two issues that need to be concerned to alleviate time-consuming in network training, i.e., the stability of network model updates and the efficient exploration mode of action space. The improvement of the optimization algorithm for the above two problems will be introduced in detail in Section 5.

2.2 Digital twin and its application on scheduling

The self-regulation of scheduling strategy can be promoted with the emergence of DT, which benefits from the progress of the Internet of Things (IoT), big data technology, visualization, communication technology, sensing technology, simulation technology, etc. The five-dimensional model is a popular model for DT, which includes DT data, an evolving virtual entity, a physical entity, and the on-demand services and their connectivity [23]. The five-dimensional model emphasizes bidirectional interaction between real and virtual to realize mutual control. The first white paper on DT was released in 2014, showing that DT could be extended to many fields [24].

DT technology has been widely used in the manufacturing industry in recent years, such as in job shop scheduling [25], diagnosis and maintenance [26], asset simulation [27], etc. In the studies of DT application on scheduling, Li et al. [28] proposed an anomaly detection and dynamic scheduling framework based on DT, and used the improved grey wolf optimization algorithm to solve the job shop problem. Yan et al. [29] proposed a machine maintenance strategy enabled by DT, integrating with Q–learning algorithm for effective scheduling. Zhang et al. [30] proposed a five-dimension DT of the machine for the availability prediction in the job shop scheduling, triggering rescheduling when necessary. Negri et al. [31] built a framework including machine health predictions, and embedded the machine health indicator in the virtual scheduling model, providing various scheduling alternatives by the genetic algorithm.

These studies indicate the positive function of DT in job shop scheduling. Most of them diagnose equipment faults by DT to adjust the scheduling strategy, which inspires future research. Unlike previous works, this paper intends to obtain the deviation between the real-time and predicted state through the interaction between physical and virtual job shops to realize the self-regulation of scheduling policy.

3 Problem description and optimization function

3.1 Problem description

The JSSP can be regarded as a sequential decision-making problem. The n jobs and m machines are used as examples in this paper, and each job includes m different operations. Due to the process constraints, each operation has to be processed on a specific machine. A job is finished after the completion of its last operation. The proper processing sequencing of operations is critical to the completion time of the jobs. All the scheduling objectives are related to the completion time of all the jobs, and the target function of minimizing the makespan corresponds to the length of the schedule. Notations used in formulating the problem are listed in Table 2.

To facilitate modeling, several predefined constraints as follows are agreed upon for this problem [32]. (1) The sequential relationship and processing time of different operations of the same workpiece are known in advance. (2) Each machine can process at most one operation at a time. (3) Each operation can only be processed on one machine at a time. (4) Any operation should be processed continuously without interruption until completion. (5) No sequential constraints between operations of different workpieces. (6)

Table 2 Notations used in formulating the problem

Notations	Explanations
N	Number of jobs
m	Number of machines
J_b	The b_{th} job, $1\leq b\leq n$
M_k	The k_{th} machine, $1\leq k\leq m$
$O_{b,k}$	The k_{th}operation of job J_b
$t_{b,k}$	The processing time of job J_b on the machine M_k
$t_{a,k}$	The processing time of job J_a on the machine M_k
$t_{b,h}$	The processing time of job J_b on the machine M_h
$C_{b,k}$	The completion time of job J_b on the machine M_k
$C_{b,h}$	The completion time of job J_b on the machine M_h
$C_{a,k}$	The completion time of job J_a on the machine M_k

All jobs are available at time 0.

3.2 Optimization objective

The optimization function of the adaptive scheduling in this paper is defined as follows [33]:

$$C_{\max} = min\max\left\{C_{b,k}\right\}. \tag{1}$$

where b=1,2,…,n; k=1,2,…,m. max represents the completed time of the last operation of all jobs, and *min* represents the optimization goal, the minimum of all solutions.

$$C_{bk} - t_{bk} + E(1 - y_{bhk}) \geqslant C_{bh}, \tag{2}$$

where E is infinitely large values, b=1,2,…,n; k, h=1,2,…,m.

$$C_{ak} - C_{bk} + E(1 - x_{bak}) \geqslant t_{ak}, \tag{3}$$

where E is infinitely large values, a, b=1,2,…,n; k=1,2,…,m.

$$y_{bhk} = \begin{cases} 1, & \text{if job } b \text{ is processed by machine } h \text{ before } k, \\ 0, & \text{other situation}, \end{cases} \tag{4}$$

$$x_{bak} = \begin{cases} 1, & \text{if machine } k \text{ processes job } b \text{ before job } a, \\ 0, & \text{other situation}, \end{cases} \tag{5}$$

$$C_{bk} - C_{ak} \geqslant t_{bk} \ or \ C_{ak} - C_{bk} \geqslant t_{ak}, \tag{6}$$

where C_{bk} and C_{ak} are the actual completion times of job J_b and J_a on the machine M_k, b, a=1,2,…,n; k=1,2,…,m.

$$C_{bk} - C_{bh} \geqslant t_{bk} \ or \ C_{bh} - C_{bk} \geqslant t_{bh}, \tag{7}$$

where C_{bk} and C_{bh} are the actual completion times of job J_b on the machine M_k and M_h, b=1,2,…,n; k, h=1,2,…,m.

Eq. (1) is the total objective function, which minimizes the completion time of all jobs. All remaining formulas are limitations for the scheduling process. Eqs. (2) and (4) indicate that job J_b is processed on machine M_h before machine M_k. Eqs. (3) and (5) suggest that job J_b is processed on machine M_k before job J_a. Eq. (6) indicates that each machine can process only one operation at a time. Eq. (7) suggests that an operation could only be processed on one machine at a time [34]. For such an instance, this paper is to explore the adaptive scheduling strategy to address the scheduling problem.

4 DT-enabled job shop adaptive scheduling methodology

The modern complicated manufacturing environment and smart manufacturing tendency put forward a new adaptive requirement for scheduling. Self-adaptability is the significant difference between the scheduling enabled by DT and the scheduling based on traditional heuristic scheduling algorithms. The adaptive scheduling system framework enabled by DT is built in this section, which includes five modules: the physical job shop module (*PJSM*), virtual job shop module (*VJSM*), data module (*DM*), service module (*SM*), and the connection module (*CM*). To explain the operation mechanism more clearly, this section first introduces the general process of DT-enabled scheduling shown in Figure 1, then presents the interaction between modules of the adaptive scheduling system framework.

4.1 Operation process of the DT-enabled adaptive scheduling

The preparation before generating the scheduling strategy is to transfer and store the production state in the form of data in the database, such as order size, equipment capability, processing time, etc. The database contains not only the above information, but also historical and experience data, all of which will be used for the scheduling policy generation. The specific processes of the proposed DT-enabled adaptive scheduling are as follows.

Step 1: The DT-enhanced adaptive system framework is explored with the components described previously, which includes *SM*, *PJSM*, *VJSM*, *DM*, *CM*. All the components share information and data through direct or indirect interaction. Different components play different roles. In the service module, the optimization algorithm uses the saved data to train the scheduling model. The optimization algorithm continuously iterates according to its specific rules, converging to generate the scheduling model for the corresponding shop scheduling.

Step 2: The models generated in the previous step, either limited by the myopic nature of some algorithms, or limited by some field factors in the job shop, have different performances. In this step, the virtual job shop uses these scheduling models for virtual scheduling to validate their performance, screen out the models with superior performance, and eliminate the poor ones.

Step 3: The purpose of this step is to build the scheduling model base and save the perfect models validated in the

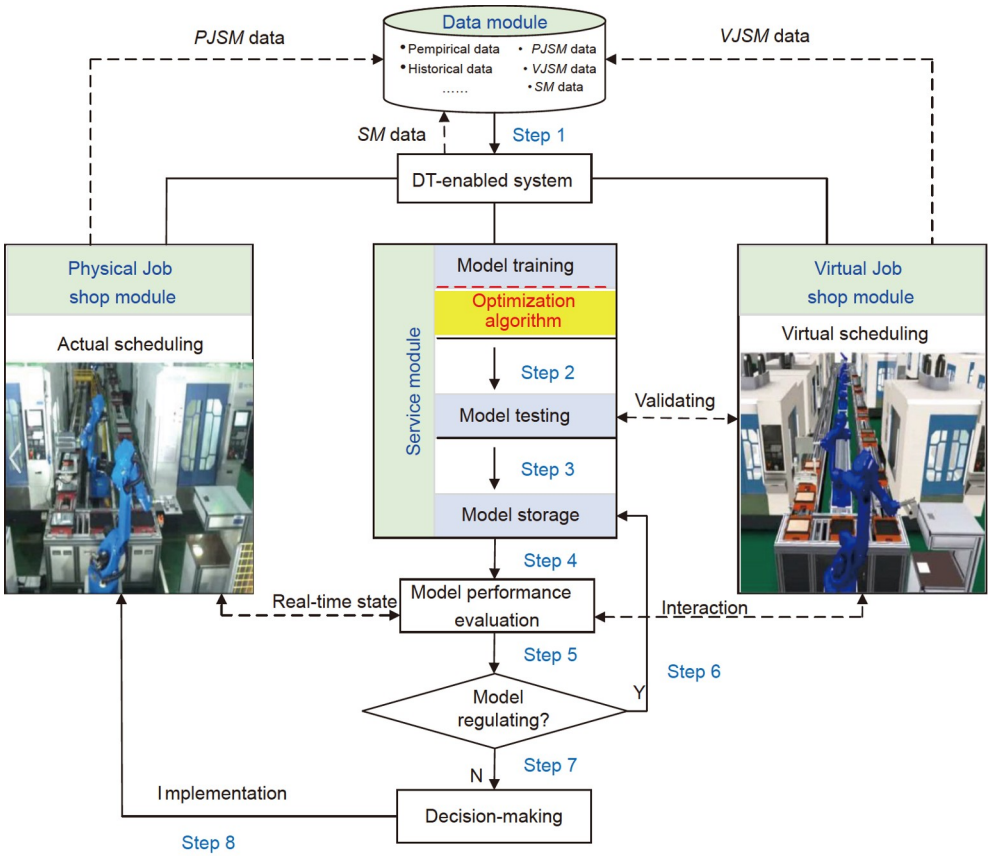

Figure 1 (Color online) Operation mechanism of the DT-enabled adaptive scheduling.

previous step for actual production calls. The model storage rules are as follows. (a) The order size is graded according to size, with a rank interval of 5. The scale difference of 5 is planned to store a single model. (b) Not all orders of any size need to store the exact model, but the stored models can be applied to orders of similar size. The purpose of these rules is to make lightweight service modules. The model load is reduced to balance the scheduling quality and computing resource consumption. The above work about model training, testing, and storage is completed in the service module.

Step 4: At this step, through the interaction between the physical and the virtual job shop, the real-time job shop data (real-time state) is transferred to the virtual job shop for virtual scheduling, and the scheduling performance of the model for the current order is evaluated.

Step 5: Based on the evaluation results of the model in the previous step, the scheduling model makes a judgment and gives instructions.

Step 6: There will be two evaluation results of the previous step. When the model is insufficient and needs to be regulated, go back to the model base in step 3, and re-match the scheduling model more consistent with the current order, then re-evaluate.

Step 7: When the evaluation result is: the current model does not need to be adjusted, it can be directly used to save the recalculation time and promote the efficient production.

Step 8: After the model evaluation and adjustment, the model determined is applied to the actual physical workshop. The data from physical, virtual, and service modules are stored in the database for future use.

4.2 Interactions between DT-enabled adaptive scheduling system framework modules

From the process in the previous section, it can be seen that the service module plays a key role in the proposed adaptive scheduling system framework. The proposed scheduling system framework is shown in Figure 2. The service module acquires information through interaction with other modules and employs an optimization algorithm to train, validate, and save the optimal scheduling models. These interactions of service modules with other modules, which like the blood vessels of the adaptive scheduling system framework and contribute to the adaptive scheduling model, are described as follows.

4.2.1 Interaction between the service module and physical job shop module

The physical shop is a job shop filled with machine tools and sensing hardware. It is responsible for receiving production

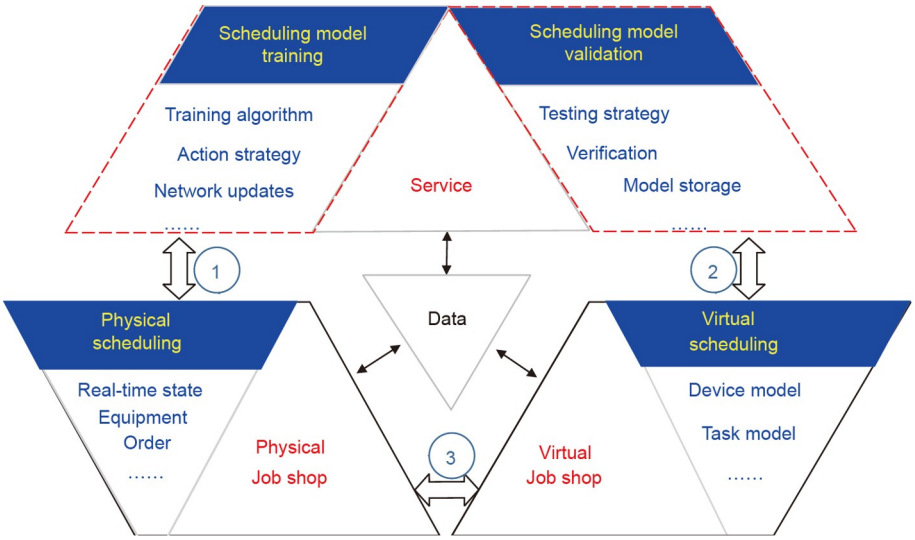

Figure 2 (Color online) DT-enabled adaptive scheduling system framework.

tasks, and executing production activities in strict accordance with the scheduling strategy optimized by the virtual shop scheduling. The physical shop is also capable of sensing and transferring data from multiple heterogeneous sources, presenting the production state (S_t). The production state of the physical job shop can be described as follows:

$$S_t = S_{equip} + S_{job} + S_{order} + S_{enviro}, \tag{8}$$

where S_{equip} indicates device capabilities and equipment health status M_k; S_{job} indicates workpiece tasks O_{bi}; S_{order} indicates the order information J_b and S_{enviro} indicates the production environment A_{bk}, t_{bk}, and C_{bk}.

These factors of production are integrated into the production state (S_t) at each scheduling decision-making point. The state (S_t) is served as the input to the optimization algorithm in the service module for agent learning. The state of the next decision point (S_{t+1}) changes according to the timely reward (r_t) that the agent receives after executing the action (a_t). Finally, the scheduling model is generated through continuous learning, trial-and-error, and experience accumulation.

The interaction between the service module and the physical job shop module contributes to the generation of a scheduling model, which has a self-learning ability and can adapt to complex production environments.

4.2.2 Interaction between the service module and virtual job shop module

The virtual job shop module is essentially a collection of models and mapping of the physical job shop, such as machines, materials, workpieces, and environments. The virtual job shop module not only focuses on the digital geometric modeling of production resources, but also portrays the physical attributes and behavior rules of production resources.

The service module interacts with the virtual module, and drives the latter to carry out the generated model to validate its performance. The scheduling models well-trained will be saved in the service module to compose the model base with the storage rules. These saved scheduling models have excellent performance, meaning that they can obtain the maximum cumulative reward value in the scheduling process to achieve the optimization objective of makespan $C_{\min}$ in this paper.

On the other hand, the virtual module interacts with the service module and perceives the history state data of the job shop, simulating the actual production. So virtual scheduling is performed to verify the effectiveness of the saved scheduling model, and the verification results are fed back to the service module to optimize the saved scheduling model.

4.2.3 Virtual-physical interaction facilitated by the service module

Through the interaction with the service module, the real-time production data (S_t) from the physical module is passed to the virtual module to simulate actual production. The saved scheduling model is used for virtual scheduling to verify its superiority, and it will be adjusted according to the deviation between the real-time production state (S_t) and the predetermined state. The saved scheduling model in the service module will be updated.

On the other hand, the virtual scheduling reflects the actual scheduling effect of the physical job shop. It guides the actual production activities by evaluating and regulating the saved scheduling model in the service module on time.

It can be seen from the above that the service module plays a critical position in the adaptive scheduling system framework. All interactions between the service module and the others are supported by the data module, which stores all the data for the entire process and lifecycle. In the service

module, the optimization algorithm is a powerful tool for generating and testing the scheduling model. The improved optimization algorithm is described in detail in Section 5 below.

5 Optimization approach of the DT-enabled adaptive scheduling

In the service module of the DT-enabled adaptive scheduling system framework, optimization algorithms play an essential role in generating scheduling models. An improved proximal policy optimization (PPO) with an innovation search strategy and asynchronous update mechanism is proposed to train the scheduling model in this paper. This section begins with a review of the basic PPO algorithm, and then presents the design detail of the proposed method, such as the pseudo-code and Markov decision process of the shop floor scheduling environment.

5.1 Review of the proximal policy optimization algorithm

PPO is an RL algorithm developed based on policy gradient (PG). It is based on the typical actor-critic structure, where the actor-network is used for action selection, and the critic-network is used for evaluating the decisions made by the actor, figuring out the state value function $V(S_t)$. On this basis, it restricts the updating range of new and old policies to guarantee stability, making the PG algorithm less sensitive to a more significant learning rate.

It implements the clip loss function, constraining the objective function value between 1–μ and 1+μ as shown in eq. (9), where the μ is a hyper-parameter [35].

$$J^{\theta'}_{PPo_2}(\theta) = \sum_{(S_t,a_t)} min\left[\begin{array}{l}\frac{p_\theta(a_t|S_t)}{p_{\theta'}(a_t|S_t)}A^{\theta'}(S_t,a_t),\\ clip(\frac{p_\theta(a_t|S_t)}{p_{\theta'}(a_t|S_t)},1-\mu,1+\mu)*A^{\theta'}(S_t,a_t)\end{array}\right], \quad (9)$$

$$A(s_t,a_t) = \sum_{t'>t}\gamma^{t'-t}r_{t'} - V(s_t). \quad (10)$$

The advantage function eq. (10) is defined by integrating $V(S_t)$ with the discounted reward, and it represents the additional return from taking action a_t.The baseline value $V(S_t)$ is minus so that the variance is more minor, and the network is trained by the Adam optimizer.

In this paper, the agent is used to interact with the production environment to generate scheduling data (S_t), such as processing time, machine allocation, scheduling current operation, etc. The data is collected and stored in the buffer. After a trajectory, all operations are allocated and processed, and the actor-network and critic-network use the stored scheduling data (S_t) to learn experience. The temporal difference error is used for gradient descent to update the critic-network, and the policy gradient is used for gradient ascent to update the actor-network, and find the best actor-critic networks to deal with variable production status. The role of the network is to form the model with the maximum cumulative rewards, seeking a mapping relationship between states and actions. The specific process of scheduling is described in Figure 3.

As described in subsection 2.1, RL has the deficiency of being time-consuming in network training. Therefore, a more efficient action selection strategy and more stable network updates would be designed in this paper, called E2APPO.

5.2 Design of the E2APPO algorithm

5.2.1 Explicit exploration of the action space and asynchronous update of networks

The E2APPO algorithm, which serves as the critical part of the adaptive scheduling system framework, is proposed with the following pseudo-code by designing an innovation action selection strategy and an asynchronous update method.

In the following pseudo-code shown in Table 3, step 1 uses Markov to design the key elements (s_t, a_t, r_t) used in the RL process, which will be covered in detail in the following part

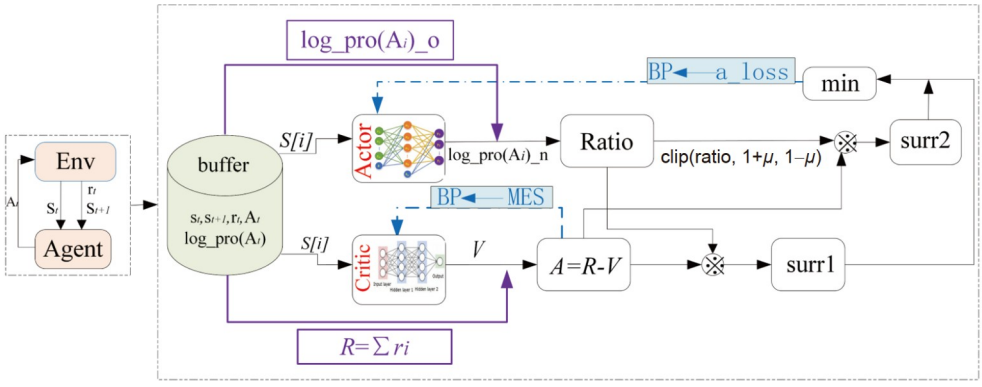

Figure 3 (Color online) Algorithm process based on PPO.

Table 3 The pseudo-code of E2APPO for job shop scheduling

Explicit exploration and asynchronous update proximal policy optimization algorithm
Input: actor-network π_θ with trainable parameter θ; critic-network v_w with trainable parameter w, clipping ratio μ. Batch size N, K is the multiple of the update frequency of the critic compared to the actor, discounting factor γ, policy loss coefficient c_p, value function loss coefficient c_v, entropy loss coefficient c_e.
Step 1. Markov modeling of job environments, design of environment states (S_t), actions (a_t), and rewards (r_t);
Step 2. Initialize π_θ, $\pi_{\theta old}$, and v_w,
Step 3. for i =0,1,2…N do
Step 4. for j=0,1,2…J do
Step 5. Observe $s_{i,j}$, select action $a_{i,j}$ based on innovation strategy $\pi_{innor}(a_{i,j}\|s_{i,j})$
Step 6. Receive reward $r_{i,j}$ and next state $s_{i,j+1}$
Step 7. Estimate advantages: $\widehat{A_{i,j}} = \sum_0^j \gamma^j r_{i,j} - V_\varphi(s_{i,j})$, $r_{i,j}(\theta) = \dfrac{\pi_\theta(a_{i,j}\|s_{i,j})}{\pi_{\theta old}(a_{i,j}\|s_{i,j})}$
Step 8. If $s_{i,j+1}$ is terminal then
Step 9. break;
Step 10. Collecting $\{s_{i,j}, r_{i,j}, a_{i,j}\}$
Step 11. End
Step 12. $L_i^p(\theta) = \sum_0^j \min(r_{i,j}(\theta)\widehat{A_{i,j}}, clip(r_{i,j}(\theta), 1-\mu, 1+\mu)\widehat{A_{i,j}})$
Step 13. $L_i^v(w) = \sum_0^j (v_w(s_{i,j}) - \widehat{A_{i,j}})^2$
Step 14. $L_i^s(\theta) = \sum_0^j S(\pi_\theta(a_{i,j}\|s_{i,j}))$
Step 15. Aggregate losses: $L_i(\theta, w) = c_p L_i^p(\theta) - c_v L_i^v(w) + c_e L_i^s(\theta)$
Step 16. Update critic $w \leftarrow \sum_{\min}(y - Q_{\theta_i}(s,a))^2$
Step 17. if i%k=0
Step 18. Update actor θ by a gradient method;
Step 19. θ = argmax($\sum_i^N L_i(\theta, w)$)
Step 20. $\pi_{\text{old}} \leftarrow \pi_\theta$
Step 21. End
Step 22. End

5.2.2. N is the number of trajectories, and J is the training steps per trajectory. In each trajectory, steps 2–11 represent the agent interacting with the production environment and collecting data.

Step 5 is the action selection strategy, representing rules for the agent to select an action based on state. The softmax is the action selection strategy of the original PPO, which is based on a stochastic sampling of output probabilities, and the exploration efficiency needs improvement. Therefore, this paper proposes an innovation strategy to select the action with the highest probability, as shown in eq. (11). The highest probability action is selected within 1–ε, and the rest are randomly sampled according to the distribution of the output. The proposed innovation strategy reduces meaningless exploration and gives the agent a more explicit direction for action exploration. ε means the balance between exploration and exploitation, generally adjusted between 0.05–0.15. By the experimental comparison, 0.1 was adopted.

$$\begin{cases} \pi_{innov} = argmax(\pi(\cdot|s_t)), \ if\ rand > \varepsilon, \\ \pi_{innov} = random.randint(\ ), \ if\ rand < \varepsilon, \end{cases} \tag{11}$$

where π_{innov} is the innovation strategy, and $\pi(\cdot|s_t)$ is the output probability of actions.

At the end of a trajectory, step 7 evaluates the advantage function. Steps 17–21 draw on a delay strategy to form the asynchronous updating mechanism between the actor-network and the critic-network. The asynchronous updating mechanism reduces erroneous updates through that the actor updates slower than the critic-network. Such advantages can avoid unnecessary repeated updates, and reduce the cumulative error in repeated updates. K represents the actor update delay coefficient, and the optimal value is 2 through the training experiment.

Unlike most other RL algorithms, the smooth loss function is used instead of the mean square error loss function in the E2APPO. Smooth is insensitive to outliers and guarantees stability [36]. In job shop scheduling, outliers are inevitable when exploring spatial values. The model generated from the smooth loss function in eq. (12) has better robustness and could adapt to different scheduling cases.

$$loss(x,y) = \frac{1}{n}\sum_i z_i, \tag{12}$$

$$z_i = \begin{cases} 0.5(x_i - y_i)^2, & |x_i - y_i| < 1, \\ |x_i - y_i| - 0.5, & \text{otherwise}, \end{cases}$$

where steps 17–21 represent the network update, the activation function Swish as eq. (13) is adopted in our neural network to maximize model performance. It can be regarded as a smooth function between the linear function and the

Relu function, combining the advantages of both above. According to Google's paper [37], the Swish performs better than the Relu activation function. The swish activation function is used in our experiment, which shows better accuracy.

$$f(x) = x \,.\, sigmoid(\beta x), \tag{13}$$

where β is a trainable parameter.

5.2.2 Markov process modeling for job shop scheduling

The process of the job shop schedule should be translated into the Markov decision process (MDP), as shown in Figure 4. The agent observes the job shop scheduling state and selects the mapping action. The agent would receive an immediate reward with the execution of the operation, and maximize the cumulative reward to learn the optimal scheduling strategy. The critical elements, which play an essential role in the E2APPO, are state, action, and reward as follows.

(1) Feature extraction of job shop state based on the graph neural networks (GNN)

The job shop scheduling status can be represented by the disjunctive graph, which offers a comprehensive view that includes processing time, and pre-constraints sequence on each machine [38]. The state at the decision point of shop scheduling is expressed as a disjunctive graph $G=(N, A, E)$, $N = O \cup \{O_s, O_e\} = \left\{O_s, O_{1,1}, \cdots, O_{1,vn}, \cdots, O_{n,1} \cdots, O_{n,vn}, O_e\right\}$, where the set describes the set O of all operations, including the start and the end dummy nodes. The set A of conjunctions indicates the pre-constraints of the operations for the same job, and A contains a directed edge $O_{j,k} \rightarrow O_{j,k+1}$ for each $j \in J$. The set E of disjunctions reflects the undirected arcs, each connects to a pair of operations that require the same machine for processing. Thus, looking for a solution for a job scheduling instance is the same as determining the direction of each disjunction.

GNN is an effective method to extract disjunctive graph features, which takes the disjunctive graph as input and updates the new graph. The technique-based spatial domain has three steps to obtain the features: sampling the neighborhood, subsequently calculating the correlation between the target node and its neighbor nodes, then aggregating a single vector from the received message, which represents the shop state. Taking $G=(N,A,E)$ as an instance, GNN is used to take on iterations to attain a multi-dimensional embedding for each node [39], and the update equation is described as eq. (14). This method of extracting scheduling state features by the neural network has obvious advantages compared with other manually extracting feature methods, which require expert knowledge, consider only local information, and lead to different scheduling performance in varied instances. Features extracted by GNN are based on the raw data to express the current state better and avoid artificial deficiency.

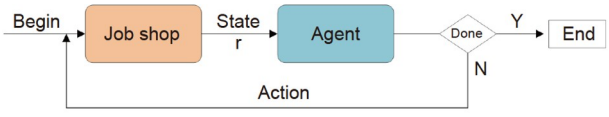

Figure 4 (Color online) MDP process of job shop scheduling.

$$h_v^{(k)} = \sigma * (W * mean(\left\{h_v^{k-1}\right\} \cup \left\{h_u^{k-1}, \forall u \in N(v)\right\})), \tag{14}$$

where σ is the non-linearity, W is the weight matrices, h is the node feature, k is the depth, and neighborhood function N.

(2) Action modeling of the agent in the job shop

$A(i,j)$ represents the set of actions in each decision-making point. In a job shop scheduling, action space generally refers to the operations or heuristic rules that can be selected. In addition, there are some other different forms, such as equipment sets and parameter selection. As mentioned in subsection 2.1, this paper has reviewed articles on job shop scheduling with RL in recent years and listed them in Table 1, which lists four different action space types designed by scholars. Considering the execution efficiency of actions, the candidate operations of workpieces are designed as action space in our experiment. Operation $O_t \in A_t$ is selected as action at the decision-making point. Given that each job can only prepare one operation at time t, the action set size equals the number of jobs, and decreases as the completion of jobs.

(3) Reward modeling of the agent in the job shop

The reward function leads the agent to achieve maximum cumulative rewards. Our agent aims to minimize the makespan $C_{\max}$ with an optimal scheduling strategy. $C_{\max}$ is the maximum completion time for all jobs, the same as the entire range of the schedule. The reward function is defined as eq. (15) as follows, where $r(a_t, s_t)$ stands for the immediate reward between state s_t and state s_{t+1}. And $r(a_t, s_t)$ is positively related to the magnitude of the effect after executing action a_t. Maximizing the cumulative sum of immediate reward is the same as minimizing the makespan. The reward design is the key to the convergence of RL, and this work is considered the most crucial part of the design of the environment for RL.

$$r(a_t, s_t) = T(s_t) - T(s_{t+1}), \tag{15}$$

where $T(s_t)$ stands for the estimated completion time of production in state s_t, $T(s_{t+1})$ stands for the estimated completion time of production in the next state.

6 Numerical experiments

In this part, the parameter settings and optimization of the training process are presented first. Then this paper provides the performance comparison between the proposed E2APPO algorithm and classical meta-heuristic algorithms, other accredited dispatching rules, as well as other two RL methods.

6.1 Experiment parameter

The training process is under the proposed scheduling sys-

tem framework, and the processing time of training instances in various sizes is randomly generated in the range of 1–99. Experimentally, the number of training trajectories 10000 can reach convergence. The proposed E2APPO was coded in Pytorch and performed on PC with Intel Core i7 -6700@4.0 GHz CPU, GEFORCE RTX 2080Ti GPU, and 8 GB RAM. Table 4 shows the parameters of the training process, which are an elaborately set of preliminary experiments. The new instance is generated randomly at each epoch to promote the generality of the E2APPO algorithm in complicated manufacturing environments during the training process. After each training epoch, the trained E2APPO model is tested on a validation instance to decide whether to save the current model as the best one.

Table 4 Parameter settings in the training process

Parameter	Value
Number of training epochs	10000
Replay memory size	10^6
Clipping parameter μ	0.2
Innovation strategy parameter ε	0.05–0.15
Learning rate lr	2×10^{-5}
Delay coefficient K	2
Discount factor γ	1
GAE parameter λ	0.98
Optimizer	Adam

The innovation action selection strategy can be seen as compatible with the advantages of both the stochastic and deterministic strategies, which avoids falling into a local optimum and has a more precise direction of exploration, preventing meaningless exploration and consumption. Figure 5(a) shows the convergence of the innovation action selection strategy and the original softmax strategy. The reward curve of the proposed innovation action selection strategy is essentially above the other strategy, indicating that the cumulative reward value of the innovation strategy is greater than the other. The proposed innovation action selection strategy outperformed the softmax strategy for searching in the action space.

The parameter ε is the balance of space exploration and exploitation, as shown in Figure 5(b). The innovation strategy parameter ε, the probability of exploration, was optimized in the range of 0.05–0.15, compared with ε=1, which means the pure random action without learning. The experimental result is shown that the reward curve tends to increase gradually for several epsilons, except for ε=1, where the reward curve fluctuates irregularly. After about 3000 rounds, the ε=0.1 curve has been at the top, and the reward value of ε=0.15 decreases in the later section. The reason may be that ε increases, leading to inadequate exploitation. During the training process, the comparison resulted in an optimal value of 0.1 for ε.

In the asynchronous update mechanism, parameter k represents the frequency of delayed updates of the actor-network compared with the critic-network. The optimal value of multiple k was selected from 1–3. To better display the convergence under different coefficients k, this experiment training number expands to 16000. As shown in Figure 5(c), the convergence curves for k=1 and k=2 were consistently higher. k=1 is at a high level at the beginning of the training stage, but lowered below the k=2 curve in the latter part due to frequent updates of the actor when the critic-network was

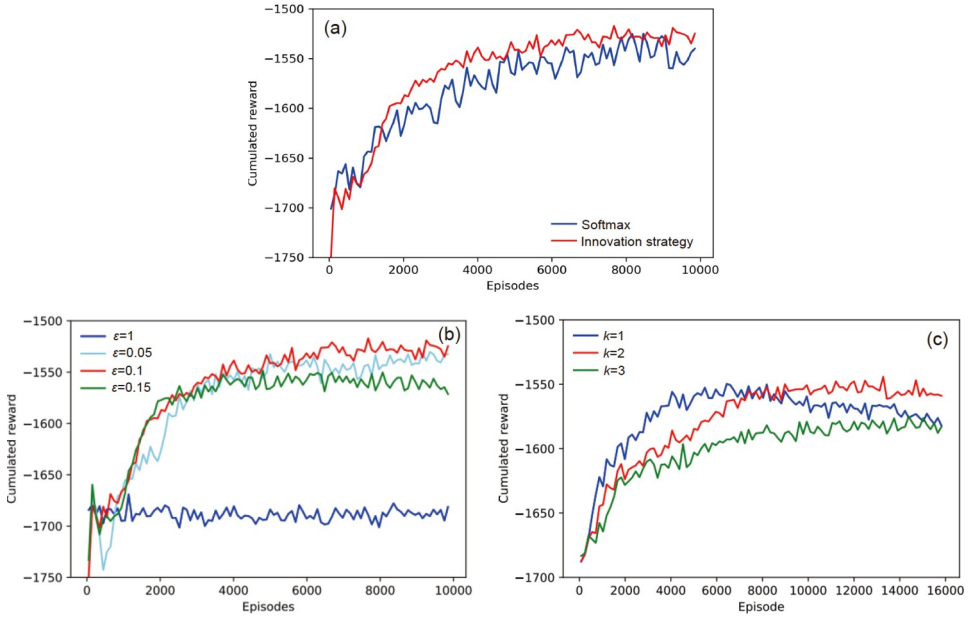

Figure 5 (Color online) Convergence curves under different experiments. (a) Convergence of different action selection strategies; (b) convergence of different ε; (c) convergence of different coefficients k.

uncertain. It can be concluded that the asynchronous update strategy with coefficient k=2 stabilizes the entire training and converges to the highest point in the later stage of training compared to k=1.

6.2 Performance metrics and benchmark

For this paper, the goal is to find a scheduling scheme that minimizes the makespan. To evaluate various scheduling methods comprehensively, the performance score represents the gap between the minimum makespan obtained by different methods and the OR-Tools as shown in eq. (16). The higher the performance score, the more effective the method is.

$$\text{Performance scores}=(1-\frac{M_i-M_{\text{best}}}{M_{\text{best}}})*100\%, \tag{16}$$

where M_i is the makespan by different methods, M_{best} is the makespan by the OR-Tools solution.

The proposed method was evaluated on various scale instances, from 6×6 to 100×20. Two benchmark datasets used in this paper were well-known public JSSP benchmarks and generated instances, nearly 90 cases were selected from the public benchmarks. Among them, small and medium-scale examples were derived from FT [40], LA [41], and ORB [42]. The large-scale examples were selected from the DMU [43] and TA [44] data set to compare with the literature [17]. We adopted the same generated instances from the literature [45] to facilitate comparison with the algorithm.

6.3 Verification of the DT-enabled adaptive scheduling strategy

6.3.1 Comparisons with meta-heuristic and heuristic algorithms

To prove the superiority of the proposed E2APPO algorithm over the meta-heuristic and heuristic algorithms, we made a comparison with the genetic algorithm (GA) and several common high-performance priority rules. Meta-heuristic and heuristic algorithms have good performance for solving JSSP problems, but they are confronted with recalculation when encountering a different JSSP instance and expend a lot of time again.

The proposed E2APPO algorithm is compared with GA on 25 well-known instances as required in the literature mentioned above. Thanks to the excellent generalization ability of the proposed algorithm, we only trained the 6×6 and 10×10 scale models. The above two models were used in other instances to save a lot of training time. As shown in Figure 6, the mean value of ten times is presented. The performance of the proposed E2APPO outperforms GA in 17 cases, equals GA in 3 cases, and slightly underperforms GA in the remaining 5 cases. It is not difficult to find that the E2APPO algorithm has no absolute advantages in quality compared with the GA, but the trained model can be generalized to similar scale scheduling problems and obtain approximate excellent solutions. For large-scale cases, the running time of the proposed method is only in seconds, which shows a significant advantage over meta-heuristic algorithms that are often several minutes or even several hours. This advantage is attributed to the abstraction and generality of state information extracted by neural networks in the learning process, and thanks to the ability of the agent to learn experiences and lessons. Finally, the model with excellent self-learning ability and adaptability is generated.

The compared rules are as follows.

(1) Shortest Processing Time (SPT): selects the next operation with the shortest processing time.

(2) First In First Out (FIFO): selects the next operation of the earliest arriving job.

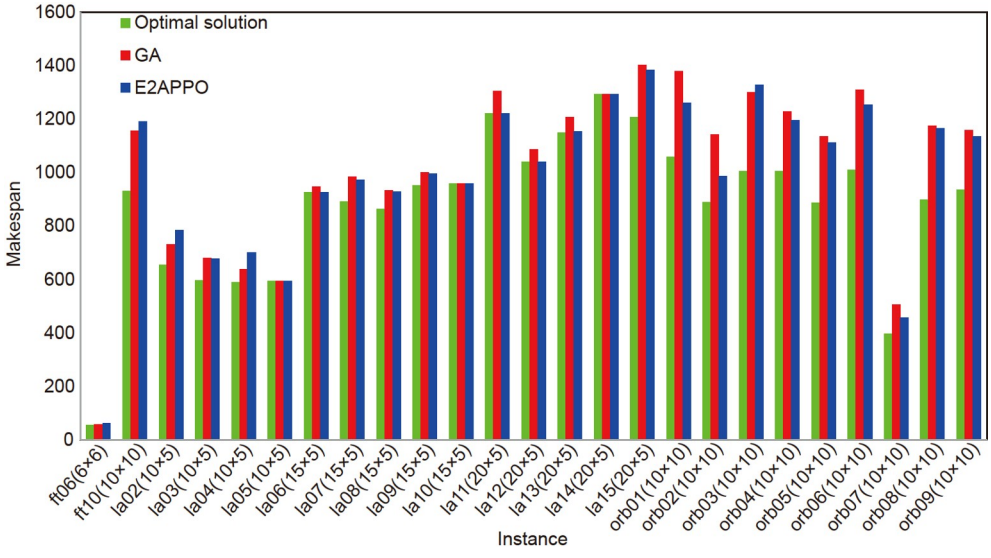

Figure 6 (Color online) Comparison between E2APPO and GA.

(3) Longest Processing Time (LPT): selects the next operation with the longest processing time.

(4) Most Operation Remaining (MOPR): The job with the most remaining operations to be completed is processed first.

(5) Most Work Remaining (MWKR): Highest priority is given to the operation belonging to the job with the most total processing time remaining to be done.

(6) Minimum ratio of Flow Due Date to Most Work Remaining (FDD): Highest priority is given to the job with an earliest due date.

The comparison between scheduling rules and the E2APPO algorithm is shown in Table 5. Like the comparison experiment with GA, each case ran ten times, and the mean was taken as the counterweight. The E2APPO algorithm outperformed the rule in 18 cases, with a 72% exceedance rate, which indicates the advantages of the E2APPO algorithm over rules. Meanwhile, from the last column in the table, the running time of the proposed E2APPO is very short, no more than 1 s, which indicates a good balance between quality and efficiency. To further prove the advantages of the E2APPO in adaptive ability, the well-trained model of 30×20 was generalized to large-size instances from 40×15 to 100×20. The mean value of 10 times is compared with the well-known rules, as shown in Figure 7. It shows that the curve of the E2APPO algorithm is always at the lower left of the well-known rules, which not only proves the superiority of the proposed method, but also proves that the 30×20 model can also quickly solve the optimal value of similar scales.

It can be concluded that the E2APPO algorithm has strong generalization ability and adaptive performance. Thanks to asynchronous update mechanisms and action selection strategy, the proposed method is more suitable for complex and uncertain production environments.

6.3.2 *Comparisons with existing RL scheduling algorithms*

To further confirm the advantage of the E2APPO algorithm, the traditional PPO agent [45] and the DQN agent [17] algorithms were chosen for comparison. As shown in Figure 8, it can be first observed that the proposed scheduling algorithm can outperform the performance of the traditional PPO and obtain higher scheduling scores in almost all instances.

Table 5 Results of dispatching rules and E2APPO

Instance	Optimal solution	SPT (score%)	LPT (score%)	FIFO (score%)	E2APPO (score%)	E2APPO running time (s)
ft06(6×6)	55	88(40)	77(60)	65(81.8)	**63**(85.45)	0.774
ft10(10×10)	930	**1074**(84.52)	1295(60.75)	1184(72.6)	1190(72.04)	0.907
la02(10×5)	655	821(74.66)	990(48.85)	830(73.3)	**785**(80.2)	0.832
la03(10×5)	597	**672**(87.44)	825(61.81)	755(73.5)	681(85.9)	0.786
la04(10×5)	590	711(79.49)	818(61.3)	**695**(82.2)	701(81.19)	0.812
la05(10×5)	593	610(97.13)	693(84.14)	610(97.1)	**593**(100)	0.808
la06(10×5)	926	1200(70.4)	1125(78.51)	926(100)	**926**(100)	0.872
la07(15×5)	890	1034(83.82)	1069(79.89)	1088(77.75)	**973**(90.6)	0.866
la08(15×5)	863	942(90.85)	1035(80.07)	980(86.44)	**929**(92.35)	0.822
la09(15×5)	951	1045(90.12)	1183(75.6)	1018(92.9)	**996**(95.27)	0.845
la10(15×5)	958	1049(90.50)	1132(81.84)	1006(95)	**958**(100)	0.864
la11(20×5)	1222	1473(79.46)	1467(79.95)	1272(95.9)	**1222**(100)	0.959
la12(20×5)	1039	1203(84.22)	1240(80.65)	1039(100)	**1039**(100)	0.963
la13(20×5)	1150	1275(89.13)	1230(93.04)	1199(95.7)	**1150**(100)	0.903
la14(20×5)	1292	1427(89.55)	1434(89.01)	1292(100)	**1292**(100)	0.941
la15(20×5)	1207	**1339**(89.06)	1612(66.45)	1403(83.76)	1404(83.68)	0.919
orb01(10×10)	1059	1478(60.43)	1410(66.86)	1379(69.7)	**1261**(80.9)	0.958
orb02(10×10)	888	1175(67.68)	1293(54.39)	1141(71.51)	**987**(88.85)	0.914
orb03(10×10)	1005	**1179**(82.69)	1430(57.7)	1300(70.6)	1327(67.96)	0.926
orb04(10×10)	1005	1236(77.01)	1415(59.2)	1229(77.7)	**1196**(81)	0.944
orb05(10×10)	887	1152(70.12)	**1099**(76.1)	1135(72.04)	1113(74.52)	0.88
orb06(10×10)	1010	**1190**(82.18)	1474(54.06)	1309(70.4)	1254(75.84)	0.953
orb07(10×10)	397	504(73.05)	470(81.61)	505(72.8)	**458**(84.63)	0.919
orb08(10×10)	899	1107(76.86)	1176(69.19)	1174(69.4)	1166(70.30)	0.832
orb09(10×10)	934	1262(64.88)	1286(62.31)	1158(76.02)	**1134**(78.59)	0.923

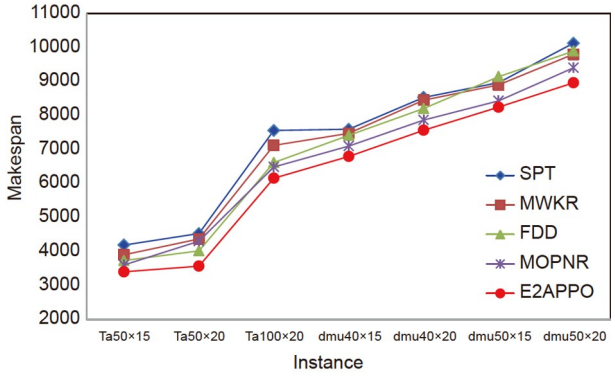

Figure 7 (Color online) Generalization of E2APPO for large scale.

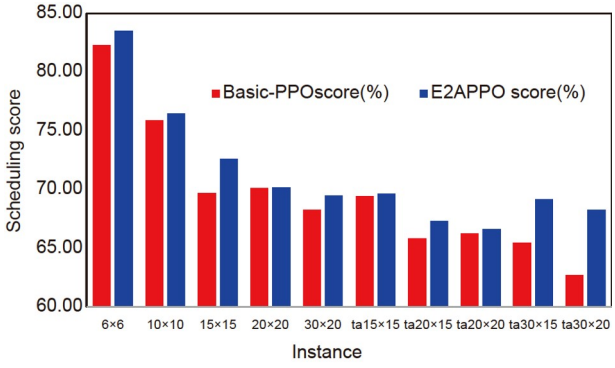

Figure 8 (Color online) Scheduling score of E2APPO and traditional PPO.

Especially for the instance ta30×20, the mean of the scheduling score is increased by up to 5.6%, which proves the effectiveness of the innovation action selection strategy.

Meanwhile, Table 6 presents the test results of several well-known rules, improved DQN (MDQN), and E2APPO algorithms on the DMU dataset. The best values are in bold. Compared with MDQN, the makespan of E2APPO for all the instances is significantly improved, with an average decrease

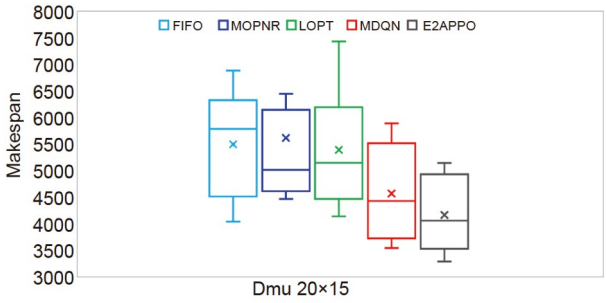

Figure 9 (Color online) Comparison of E2APPO and MDQN in training stability.

of 8.9%. It can be concluded from Figure 9 that E2APPO's performance is evenly distributed without extreme deviation values, which has clear superiority for considering stability. It confirms the effectiveness of the proposed asynchronous update mechanism.

6.3.3 Performance validating of the DT-enabled scheduling methodology

From the process of the DT-enabled adaptive system framework, it can be seen that the current model is evaluated and adjusted through virtual scheduling. To prove the self-regulation effect of the DT-enabled scheduling methodology, the current model 6×6 is taken as an example in this section. The comparative experiments are conducted on the solution quality before and after adjustment of the scheduling model when the order changes to 15×15, 20×20, and 30×20. Each experiment runs 100 times, and the mean value is taken for comparison, as shown in Figure 10.

The DT-enabled scheduling regulates the model on demand by testing and verifying through the virtual-real interaction. The graph represents that the scheduling shows a shorter makespan after regulation, which reflects the positive effect of the model adjustment. The results indicate the excellent adaptability of the DT-enabled scheduling method to

Table 6 The comparison of MDQN and E2APPO on DMU benchmark

Instance	FIFO	SPT	LPT	MOPNR	LOPT	SQN	LQN	MDQN	E2APPO
dmu01	4668	32364	28233	4618	4123	29029	27411	3520	**3260**
dmu02	4282	33442	28571	4720	4323	31364	31676	3765	**3673**
dmu03	5600	32125	34467	4560	4772	30028	31107	3953	**3556**
dmu04	4020	28414	34550	4752	4597	36116	30830	3521	**3358**
dmu05	4575	35099	33981	4456	4500	37235	30996	3790	**3624**
dmu41	5980	25095	23095	9397	5516	20223	23496	4881	**4409**
dmu42	6261	23942	25140	6459	6341	23102	23937	5895	**4895**
dmu43	6575	20003	22662	5911	6129	25026	21344	5559	**4558**
dmu44	6901	26281	22429	6055	7459	22341	23753	5502	**5023**
dmu45	6102	24615	23394	5268	6162	24987	21562	5189	**5141**
Average	5496	28137	27652	5319.6	5392	27975	26611	4557	**4149.7**

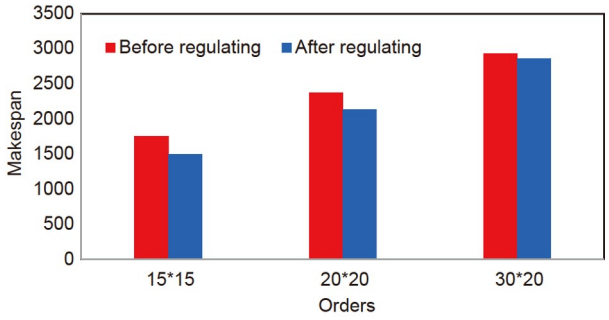

Figure 10 (Color online) Experiment of self-regulation of the DT-enabled adaptive scheduling.

the variable orders and complicated manufacturing industry.

7 Conclusions

In this paper, based on the virtual-real interaction of DT and the self-learning capability of RL, a DT-enabled adaptive scheduling system framework embedded improved optimization algorithm called E2APPO is studied. The E2APPO algorithm is developed with an innovative action selection strategy and asynchronous update mechanism to minimize the makspan. Thanks to the improvement above, the well-trained scheduling model of E2APPO has a better adaptive performance of different scales than heuristic and meta-heuristic algorithms. The DT-enabled scheduling strategy enhances adaptability to complicate shop environments and variable orders, and achieves an optimal balance between quality and time cost.

Numerical experiments are carried out on many instances, including well-known benchmarks and randomly generated instances as a realistic representation of actual manufacturing, to demonstrate the advantage of the proposed adaptive scheduling. Compared with the meta-heuristic algorithms and priority rules, the results prove the superiority of the E2APPO algorithm, especially the generalization performance to similar scales. Compared with existing RL algorithms, the E2APPO algorithm achieves better outcomes for our purpose. Finally, the optimization objectives before and after regulating the scheduling model are compared to prove the self-adaption ability of the DT-enabled scheduling.

Although the proposed approach has shown improved performance in simulation experiments, there are still some limitations in its application in the actual production process and the possible future work, including: (1) some disturbance factors are not considered in the proposed method, such as machine breakdown, random arrival of workpieces, processing time changes, etc. These dynamical disturbances, which have an important impact on mathematical models and digital twin models, are inevitable in the actual production process. (2) The proposed method is a theoretical exploration of job shop self-adaptive scheduling by combining deep reinforcement learning and digital twinning technology. However, the flexible job shop in actual manufacturing is more in line with modern production requirements. (3) In the simulation experiment, only a single optimization objective is considered. Many other objectives in the actual production process should be considered in the following research, such as tardiness, machine utilization, cost, etc.

This work was supported by the National Key R&D Program of China (Grant No. 2020YFB1713300), the Joint Open Fund of Wuhan Textile University (Grant No. KT202201005), and the Foundation of Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, Guizhou University (Grant No. GZUAMT2021KF11). The authors thank for the computing support of the State Key Laboratory of Public Big Data, Guizhou University.

1 Błażewicz J, Domschke W, Pesch E. The job shop scheduling problem: Conventional and new solution techniques. Eur J Operational Res, 1996, 93: 1–33
2 Fisher M L. Optimal solution of scheduling problems using Lagrange multipliers: Part I. Oper Res, 1973, 21: 1114–1127
3 Lomnicki Z A. A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem. J Operational Res Soc, 1965, 16: 89–100
4 Zhou H, Cheung W, Leung L C. Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. Eur J Operational Res, 2009, 194: 637–649
5 Salido M A, Escamilla J, Giret A, et al. A genetic algorithm for energy-efficiency in job-shop scheduling. Int J Adv Manuf Technol, 2016, 85: 1303–1314
6 Lin Q, Li J, Du Z, et al. A novel multi-objective particle swarm optimization with multiple search strategies. Eur J Operational Res, 2015, 247: 732–744
7 Liu X, Ni Z, Qiu X. Application of ant colony optimization algorithm in integrated process planning and scheduling. Int J Adv Manuf Technol, 2016, 84: 393–404
8 Karaboga D. Artificial bee colony algorithm. Scholarpedia, 2010, 5: 6915
9 Pan Q K. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. Eur J Operational Res, 2016, 250: 702–714
10 Qu M, Zuo Y, Xiang F, et al. An improved electromagnetism-like mechanism algorithm for energy-aware many-objective flexible job shop scheduling. Int J Adv Manuf Technol, 2022, 119: 4265–4275
11 Li X Y, Xie J, Ma Q J, et al. Improved gray wolf optimizer for distributed flexible job shop scheduling problem. Sci China Tech Sci, 2022, 65: 2105–2115
12 Wang X, Wang R, Shu G Q, et al. Energy management strategy for hybrid electric vehicle integrated with waste heat recovery system based on deep reinforcement learning. Sci China Tech Sci, 2022, 65: 713–725
13 Emary E, Zawbaa H M, Grosan C. Experienced gray wolf optimization through reinforcement learning and neural networks. IEEE Trans Neural Netw Learn Syst, 2018, 29: 681–694
14 Shahrabi J, Adibi M A, Mahootchi M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. Comput Industrial Eng, 2017, 110: 75–82
15 Zhao H, Zhang C. A decomposition-based many-objective artificial bee colony algorithm with reinforcement learning. Appl Soft Computing, 2020, 86: 105879
16 Stricker N, Kuhnle A, Sturm R, et al. Reinforcement learning for adaptive order dispatching in the semiconductor industry. CIRP Ann,

2018, 67: 511–514
17 Lin C C, Deng D J, Chih Y L, et al. Smart manufacturing scheduling with edge computing using multiclass deep Q network. IEEE Trans Ind Inf, 2019, 15: 4276–4284
18 Shi D, Fan W, Xiao Y, et al. Intelligent scheduling of discrete automated production line via deep reinforcement learning. Int J Prod Res, 2020, 58: 1–19
19 Palombarini J A, Martínez E C. Automatic generation of rescheduling knowledge in socio-technical manufacturing systems using deep reinforcement learning. In: Proceedings of 2018 IEEE Biennial Congress of Argentina. San Miguel de Tucuman, 2018. 1–8
20 Xia K, Sacco C, Kirkpatrick M, et al. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. J Manuf Syst, 2021, 58: 210–230
21 Park I B, Huh J, Kim J, et al. A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. IEEE Trans Autom Sci Eng, 2020, 17: 1420–1431
22 Zhou L, Zhang L, Horn B K P. Deep reinforcement learning-based dynamic scheduling in smart manufacturing. Procedia CIRP, 2020, 93: 383–388
23 Tao F, Zhang M, Nee A Y C. Digital Twin Driven Smart Manufacturing. Elsevier Academic Press, 2019
24 Grieves M. Digital twin: Manufacturing excellence through virtual factory replication. White Paper, 2014. 1–7
25 Zhang J, Deng T, Jiang H, et al. Bi-level dynamic scheduling architecture based on service unit digital twin agents. J Manuf Syst, 2021, 60: 59–79
26 Tao F, Zhang M, Liu Y, et al. Digital twin driven prognostics and health management for complex equipment. CIRP Ann, 2018, 67: 169–172
27 Schluse M, Priggemeyer M, Atorf L, et al. Experimentable digital twins—Streamlining simulation-based systems engineering for Industry 4.0. IEEE Trans Ind Inf, 2018, 14: 1722–1731
28 Li Y, Tao Z, Wang L, et al. Digital twin-based job shop anomaly detection and dynamic scheduling. Robotics Comput-Integrated Manuf, 2023, 79: 102443
29 Yan Q, Wang H, Wu F. Digital twin-enabled dynamic scheduling with preventive maintenance using a double-layer Q-learning algorithm. Comput Operations Res, 2022, 144: 105823
30 Zhang M, Tao F, Nee A Y C. Digital twin enhanced dynamic job-shop scheduling. J Manuf Syst, 2021, 58: 146–156
31 Negri E, Ardakani H D, Cattaneo L, et al. A digital twin-based scheduling framework including equipment health index and genetic algorithms. In: Proceedings of 13th International-Federation-of-Automatic-Control (IFAC) Workshop on Intelligent Manufacturing Systems (IMS). Oshawa, 2019. 52: 43–48
32 Pinedo M. Scheduling: Theory, Algorithms, and Systems. New York: NYU Stern School of Business, 2016. 207–214
33 Wang W Q, Ye C M, Tan X J. Job shop dynamic scheduling based on Q-learning algorithm. Comput Syst Appl, 2020, 29: 218–226
34 Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job shop scheduling. Manage Sci, 1988, 34: 391–401
35 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv: 1707.06347, 2017
36 Girshick R. Fast R-CNN. In: Proceedings of IEEE International Conference on Computer Vision. Santiago, 2015. 1440–1448
37 Zoph B, Le Q V. Searching for activation functions. In: Proceedings of International Conference of Learning Representation. Vancouver, Canada, 2018. 1–13
38 Niu G G, Sun S D, Lafon P, et al. A decomposition approach to job-shop scheduling problem with discretely controllable processing times. Sci China Tech Sci, 2011, 54: 1240–1248
39 Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst, 2020, 32: 4–24
40 Fisher H. Probabilistic learning combinations of local job-shop scheduling rules. Ind Sched, 1963, 225–251
41 Lawrence S. Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). Dissertation for the Doctoral Degree. Pennsylvania: Carnegie-Mellon University. 1984
42 Applegate D, Cook W. A computational study of the job-shop scheduling problem. ORSA J Computing, 1991, 3: 149–156
43 Demirkol E, Mehta S, Uzsoy R. Benchmarks for shop scheduling problems. Eur J Operational Res, 1998, 109: 137–141
44 Taillard E. Benchmarks for basic scheduling problems. Eur J Operational Res, 1993, 64: 278–285
45 Zhang C, Song W, Cao Z, et al. Learning to dispatch for job shop scheduling via deep reinforcement learning. Adv Neural Inf Process Syst, 2020, 33: 1621–1632