Computer Networks 190 (2021) 107969

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning

Libing Wang [a], Xin Hu [a,*], Yin Wang [a], Sujie Xu [a], Shijun Ma [a], Kexin Yang [b], Zhijun Liu [a], Weidong Wang [a]

[a] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
[b] International School, Beijing University of Posts and Telecommunications, Beijing 100876, China

ARTICLE INFO

Keywords:
Smart manufacturing
Job-shop scheduling
Deep reinforcement learning
Proximal policy optimization

ABSTRACT

Job-shop scheduling problem (JSP) is used to determine the processing order of the jobs and is a typical scheduling problem in smart manufacturing. Considering the dynamics and the uncertainties such as machine breakdown and job rework of the job-shop environment, it is essential to flexibly adjust the scheduling strategy according to the current state. Traditional methods can only obtain the optimal solution at the current time and need to rework if the state changes, which leads to high time complexity. To address the issue, this paper proposes a dynamic scheduling method based on deep reinforcement learning (DRL). In the proposed method, we adopt the proximal policy optimization (PPO) to find the optimal policy of the scheduling to deal with the dimension disaster of the state and action space caused by the increase of the problem scale. Compared with the traditional scheduling methods, the experimental results show that the proposed method can not only obtain comparative results but also can realize adaptive and real-time production scheduling.

1. Introduction

With the development of Internet of Things technology, the concepts of smart manufacturing and smart factories have emerged. Job-shop scheduling problem (JSP) is a typical scheduling problem in smart manufacturing and has attracted attention since it can improve production efficiency. JSP is essentially a combinatorial optimization problem and has been proven to be a Nondeterministic Polynomial-time (NP) hard problem [1]. Since large-scale combinatorial optimization problems are too difficult to describe mathematically, many previous studies have focused on meta-heuristic algorithms, including simulated annealing (SA) [2,3], tabu search (TS) [4], genetic algorithm (GA) [5,6]. However, these methods need to reschedule jobs when the environment state changes. This feature makes it difficult for meta-heuristic algorithms to cope with the dynamics and the uncertainties, such as machine breakdown and job rework of the job-shop environment. To solve this problem, a dynamic scheduling system for the job-shop is necessary to be studied.

In recent years, deep reinforcement learning (DRL) has achieved remarkable results on various scheduling problems, including dynamic robot control [7,8], dynamic resource management [9–14], Internet of Things communications [21–22] and smart manufacturing [15–17]. Due to the advantage of real-time decision-making, deep reinforcement learning can give immediate action for the problem if the model is well-trained. Besides, research has found that deep reinforcement learning technology has great potential in solving combinatorial optimization problems. The deep reinforcement learning method is applied to the two-dimensional bin packing problem (2D-BPP) [18], the travelling salesman problem (TSP) [19] and the vehicle routing problem (VRP) [20], which are combinatorial optimization problems similar to the job-shop scheduling problem, and obtain better performance compared to the traditional heuristic algorithms. Inspired by these applications, we model the job-shop scheduling problem as a Markov decision process (MDP) and use the deep reinforcement learning to solve the dynamic job-shop scheduling problem. In addition, to solve the high-dimensional disaster problem in the state and action space, we introduced the proximal policy optimization (PPO) algorithm into the job-shop scheduling with low complexity and good performance.

The rest of this paper is organized as follows. Section II introduces the JSP and models the JSP as a sequential decision process. Section III

* Corresponding author.
E-mail addresses: wanglibing@bupt.edu.cn (L. Wang), huxin2016@bupt.edu.cn (X. Hu), wangyin@bupt.edu.cn (Y. Wang), xsj_610@bupt.edu.cn (S. Xu), mashijun@bupt.edu.cn (S. Ma), yangkexin@bupt.edu.cn (K. Yang), lzj2017110489@bupt.edu.cn (Z. Liu), wangweidong@bupt.edu.cn (W. Wang).

https://doi.org/10.1016/j.comnet.2021.107969
Received 14 November 2020; Received in revised form 5 February 2021; Accepted 24 February 2021
Available online 26 February 2021

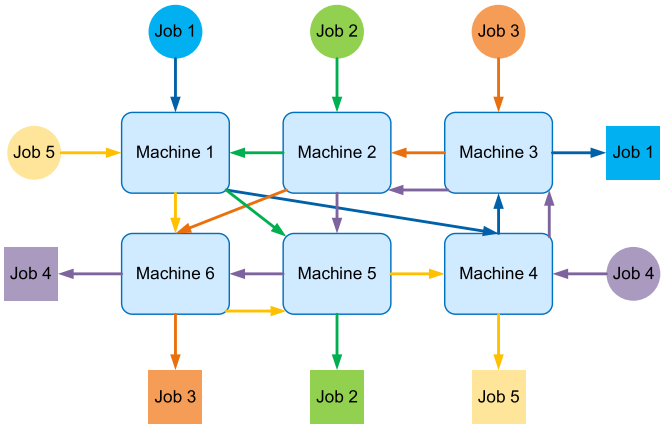

Fig. 1. Job-shop scheduling problem.

presents the DRL architecture based on the PPO algorithm to solve the problem mentioned in Section II. Section IV presents the environmental results and analysis. Finally, the conclusion is drawn in Section V.

2. System model and problem description

2.1. System model

The job-shop scheduling problem can be described as follows: suppose there are n jobs $J = \{J_1, J_2, \cdots, J_i, \cdots, J_n\}$, each job $J_i = \{O_{i1}, O_{i2}, \cdots, O_{il}, \cdots, O_{im}\}$ has m operations to be processed on m machines $M = \{M_1, M_2, \cdots, M_j, \cdots, M_m\}$, respectively. Each operation has its designated processing machine and corresponding processing time. Moreover, there is a processing order restriction for the operations in the same job. Define machine designated matrix $OM = \{m_{il} | m_{il} = M_1, M_2, \cdots, M_m\}(i = 1, 2, ..., n, l = 1, 2, ..., m)$ and processing time matrix $OT = \{t_{il} | t_{il} > 0\}(i = 1, 2, ..., n, l = 1, 2, ..., m)$, which indicate operation O_{il} is processed by machine m_{il} and t_{il} is the processing time of O_{il}.

The description of the JSP is shown in Fig. 1. Five different jobs with multiple operations need to be processed on six machines. Each job's operations have a specific order, and the next operation must be processed after the previous operation is completed. For example, Job 1 need to go through Machine 1 first, then Machine 4, and finally Machine 3 to get the output. It should be noted that each job's procedure is not exactly the same. The goal is to assign Job 1-Job 5 to be processed on machines to minimize the maximum duration for all jobs to be completed.

We then establish the job-shop scheduling problem as an optimization problem with constraints, and its mathematical description can be shown as Eq. (1).

$$\begin{aligned} opt.P &= \min\left\{\max_{1\le i\le n} c_{imm_{im}}\right\} \\ s.t.C1 &: c_{il} - t_{il} \ge c_{i(l-1)} \\ C2 &: c_{ilm_{il}} - t_{il} \ge c_{hgm_{hg}}, m_{il} = m_{hg}, h \ne i, g \ne l \\ & i, h = 1, ..., n, g = 1, ..., m, l = 2, ..., m \end{aligned} \tag{1}$$

where $c_{ilm_{il}}$ indicates the processing completion time of operation O_{il} on machine m_{il}. Therefore, P means to minimize the completion time of the last operation O_{im} of all jobs, that is, the makespan. $C1$ is the constraint on the processing sequence of adjacent operations in the same job. $C2$ denotes that each machine can only process one job at any time. For each machine, the next operation cannot be processed until the previous operation is completed.

In the JSP problem, machine designated matrix OM and processing time matrix OT are given. If each job's scheduling plan on each machine is determined, then the completion time $c_{ilm_{il}}$ of each operation O_{il} can be calculated according to Eq. (2). The completion time of an operation is the maximum of its previous operations' completion time in the same job and on the same machine plus its processing time. According to this rule, the makespan can be obtained.

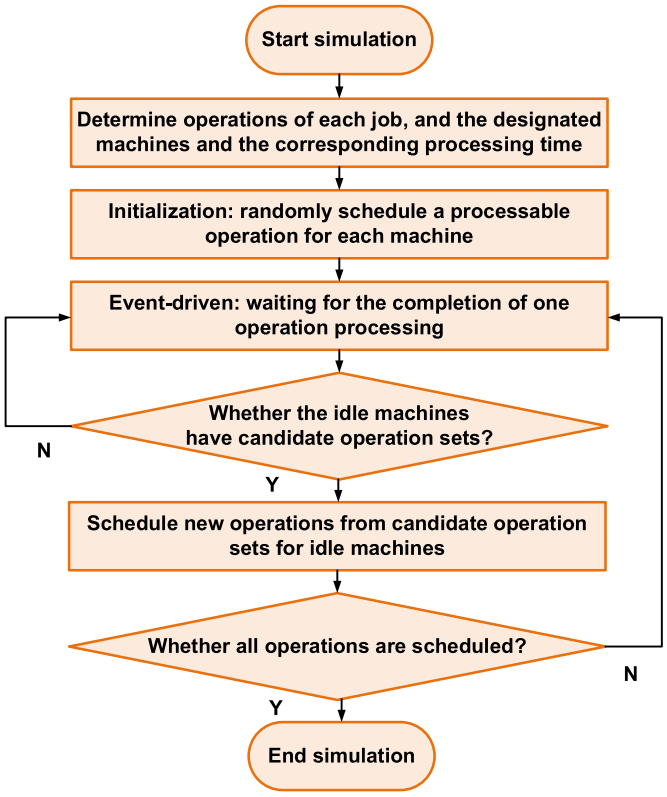

Fig. 2. Sequential decision process description.

$$\begin{aligned} & c_{ilm_{il}} = \max\left(c_{i(l-1)m_{i(l-1)}}, c_{hgm_{hg}}\right) + t_{il}, m_{il} = m_{hg} \\ & i, h = 1, ..., n, g = 1, ..., m, l = 2, ..., m \end{aligned} \tag{2}$$

where $c_{i(l-1)m_{i(l-1)}}$ and $c_{hgm_{hg}}$ are the completion time of the previous operation $O_{i(l-1)}$ in the same job J_i and the previous operation O_{hg} on the same machine m_{il}, respectively. t_{il} is the processing time of O_{il}.

2.2. Problem formulation

The goal of the job-shop scheduling is to make the machine work continuously and each job's operations be processed continuously as much as possible, so as to minimize the makespan. Therefore, the job-shop scheduling problem can be regarded as a sequential decision problem for assigning jobs to idle machines. The process is driven by the processing completion event of operations on machines. Considering the processing sequence constraints of the operations in the same jobs and the constraints that the operations must be processed on the specified machines, when assigning jobs to the idle machines, it is necessary to consider whether the machines have operations that can be processed at this time. If there are multiple optional operation sets, select one of them for processing. Otherwise, it needs to wait for the arrival of the next operation processing completion event. The process can be described as Fig. 2.

In this sequential decision problem, since the decisions are performed when any operation is completed, the state should include the job processing status. Besides, known information such as the designated machine and processing time of each job's operations should also be included in the state, because these information are closely related to the job processing status and scheduling goals. In summary, the state is defined as three matrices: the job processing status matrix, the machine

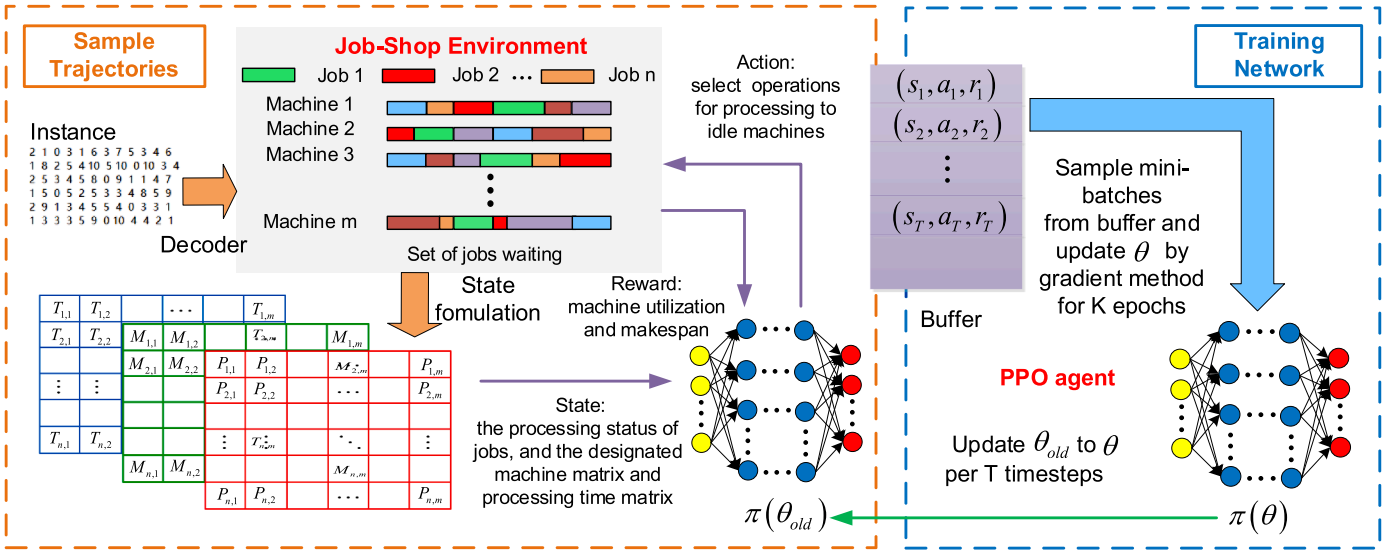

Fig. 3. DRL architecture for dynamic job-shop scheduling.

designated matrix and the processing time matrix of the operations in jobs, as shown in Eq. (3).

$$s_t = \{P_t, OM, OT\} \tag{3}$$

where $P_t = \{(p_t)_{il} | (p_t)_{il} = 0, 1\}, i = 1, \ldots, n, l = 1, \ldots, m$ is the job processing status matrix, $(p_t)_{il}$ indicates whether operation O_{il} is processed, and $(p_t)_{il} = 1$ means affirmation. OM and OT are the corresponding machine and processing time matrix for each operation in jobs, which do not change over time. The shape of the above three matrices is $n \times m$, where n is the number of jobs, m is the number of machines.

Since the job processing state at the next moment only changes based on the current moment, the job-shop scheduling problem can be modelled as a Markov decision process.

For the state of the environment, the agent provides an action a_t from a policy $\pi(a_t|s_t)$ in the corresponding timeslot. And the environment state changes after the action is taken. For the job-shop scheduling problem, the action is to assign jobs to idle machines, that is, select an operation from the set of optional operations of idle machines for processing. The action is defined as Eq. (4).

$$a_t = \{O_{1t}, O_{2t}, \cdots, O_{jt}, \cdots, O_{mt}\} \tag{4}$$

where O_{jt} represents the operation selected from the candidate operation set for machine M_j at time t. It should be noted that only idle machines can be assigned operations for processing, and there may be no candidate operation set available for idle machines at this moment. When the above situation occurs, the action will not be executed, but the agent will wait for the next schedulable moment's arrival.

For each action executed by the agent, the environment will feedback a corresponding reward. Since the reward is used to evaluate the action performed, reward setting directly affects the change of agent strategy. In order to obtain the optimal policy to solve the JSP, setting a valid reward is very important. As mentioned above, the goal of the JSP is to minimize the makespan. However, in the scheduling process, not all jobs are scheduled, so the final makespan cannot be obtained. So measuring the impact of scheduling decisions on makespan and setting appropriate rewards must be considered. Considering that makespan is related to machine utilization, the higher the machine utilization, the smaller the makespan. Therefore, we set the reward in the scheduling process to be related to machine utilization. After the final makespan is obtained after the scheduling process is over, add the makespan to reward setting. In summary, the reward function is set as Eqs. (5)-(6).

$$r_t = U_t - U_{t-1} \tag{5}$$

$$r_t = \begin{cases} U_t - U_{t-1} + 100/(makespan - L), makespan \neq L \\ U_t - U_{t-1} + 100, makespan = L \end{cases} \tag{6}$$

where L is the theoretical optimal makespan. U_t is the machine utilization at time t, it is defined as the ratio of the total processing time of all scheduled jobs at time t to the current completion time.

3. DRL architecture for job-shop scheduling

This section presents the DRL architecture of the dynamic job-shop scheduling based on the proximal policy optimization algorithm.

Deep reinforcement learning aims to enable the agent to learn to take actions to maximize the cumulated reward from the process of interacting with the environment. For the job-shop scheduling problem, DRL can implement real-time scheduling and change strategies based on the current state of the environment when dealing with emergencies. In reinforcement learning methods, comparing value-based methods, policy-based methods are more suitable for continuous state and action space problems. Regarding the high-dimensional disaster of the state space and action space brought by the increase in the JSP problem scale, it is necessary to use policy-based methods. As one of the policy-based methods, the PPO algorithm can obtain good performance by limiting the policy update to reduce parameter settings' sensitivity. Because of the above advantages, this paper uses the PPO algorithm to solve the JSP problem.

Fig. 3 shows the DRL architecture for dynamic job-shop scheduling. In this architecture, there are two main parts: the interaction part between the agent and the environment and training part of the agent. As shown in Fig. 3, there are two policies in the PPO algorithm. Among them, $\pi_{old}(\theta')$ is used to interact with the environment to collect data,

Table 1
DRL architecture for dynamic job-shop scheduling.

1: Choose a JSP instance and generate JSP scenario
2: Initialize the network parameters of the PPO algorithm, initialize $\pi(\theta) = \pi(\theta_{old})$
3: For iteration=1,2,…, do:
4: For horizon=1,2,…,T do:
5: The agent observes the state s_t: the processing status of jobs matrix, the machine designated matrix and the processing time matrix
6: The agent run policy $\pi(\theta_{old})$ to take action a_t to assign jobs to idle machines for processing
7: The environment feeds back a reward r_t for a_t, the next state s_{t+1}, and the indication of whether the scheduling process is over. If the scheduling process is not over, the reward is about machine utilization, and otherwise, the reward is about both machine utilization and makespan
8: Store (s_t, a_t, r_t) to buffer $[(s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)]$
9: End For
10: For epoch=1,2,…, K do:
11: Sample mini-batches from the buffer
12: Update θ by gradient method to optimize the objective function $L_t(\theta)$
13: End For
14: Update $\pi(\theta_{old})$ to $\pi(\theta)$

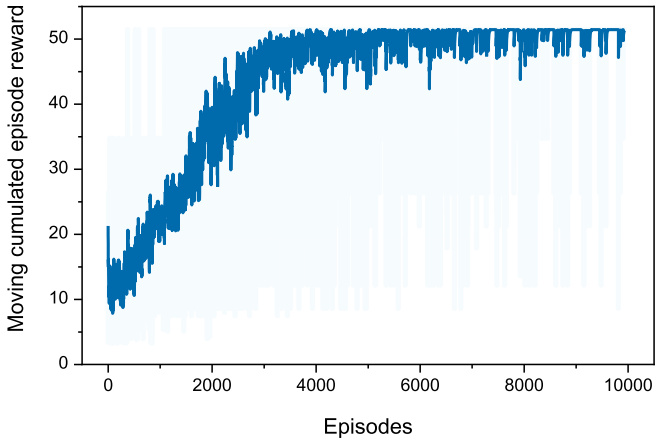

Fig. 4. Training reward based on PPO algorithm.

and $\pi(\theta)$ is used to be updated with the collected samples. And at the beginning of each episode, $\pi_{old}(\theta')$ is updated to $\pi(\theta)$.

In the interaction between the agent and the environment, the environment is defined as a set of job lists with the assigned machines and the processing time. For the job-shop environment, the agent needs to observe the environment state information at each moment, that is the processing status of jobs and the designated machine matrix and the processing time matrix, then take action to select operations for idle machines to make jobs be processed continuously and orderly, and shorten the maximum completion time as much as possible. After the action is executed, the agent receives the reward function about the machine utilization and makespan given by the environment and the next state of the environment, triggering the system's next time step.

In the agent policy update part, mini-batches are sampled from the buffer to update $\pi(\theta)$. Specifically, the parameter θ is updated according to the stochastic gradient ascent method to optimize the objective function $L_t(\theta)$ of the PPO algorithm.

In this way, the dynamic job-shop scheduling system based on PPO is constructed. And Table I shows the job-shop scheduling system flow based on the PPO algorithm.

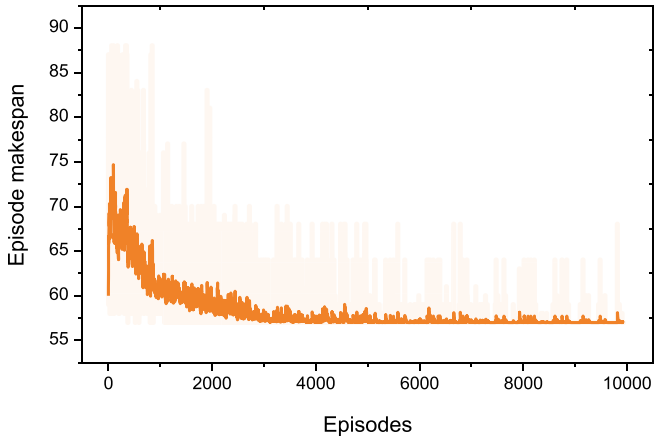

Fig. 5. Makespan curve in training process.

At the beginning of training, the agent interacts with the environment in a trial and error manner. Once the agent learns the optimal policy, it can make real-time decisions on the job-shop environment using the trained model, thereby realizing real-time scheduling. Furthermore, when the agent encounters an unknown environment state, it can make good decisions by updating the policy network online in real time, so as to achieve adaptive scheduling.

4. Simulation result and analysis

This section evaluates the use of the proposed DRL architecture in the JSP and compares the effectiveness with theoretical optimal solutions, heuristic rules and meta-heuristic algorithm. We also explored the proposed DRL model's generalization performance to verify the advantages of the DRL model over traditional methods.

4.1. Simulation results

This part introduces the static performance of the proposed DRL model in the JSP problems. The source of the JSP problems is some benchmark problems in OR-library [23].

First, we take ft06 instance with six jobs and six machines for experimental evaluation [24], as shown in Eq. (7). Build the job-shop environment with this instance and use the PPO algorithm to schedule the job-shop, the training process and the change curve of makespan are shown in Figs. 4 and 5, in which the abscissa is the number of episodes, the ordinates are the cumulative reward and the makespan for each episode, respectively.

$$J_1 = \begin{bmatrix} (2,1) & (0,3) & (1,6) & (3,7) & (5,3) & (4,6) \\ (1,8) & (2,5) & (4,10) & (5.10) & (0,10) & (3,4) \\ (2,5) & (3,4) & (5,8) & (10,9) & (1,1) & (4,7) \\ (1,5) & (0,5) & (2,5) & (3,3) & (4,8) & (5,9) \\ (2,9) & (1,3) & (4,5) & (5,4) & (0,3) & (3,1) \\ (1,3) & (3,3) & (5,9) & (0,10) & (4,4) & (2,1) \end{bmatrix} \tag{7}$$

In Figs. 4 and 5, it can be seen that the value of the reward shows an upward trend and the makespan shows a downward trend. The reward begins to converge around 4000 episodes and is nearly stable after 6000

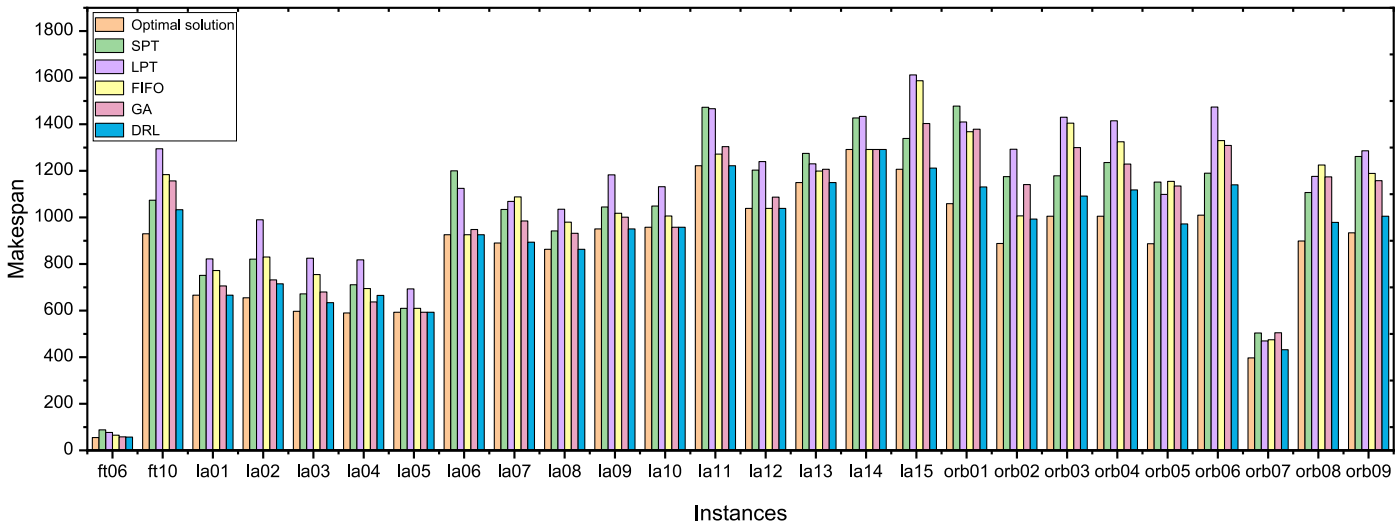

Fig. 6. Makespan with different algorithms on OR instances.

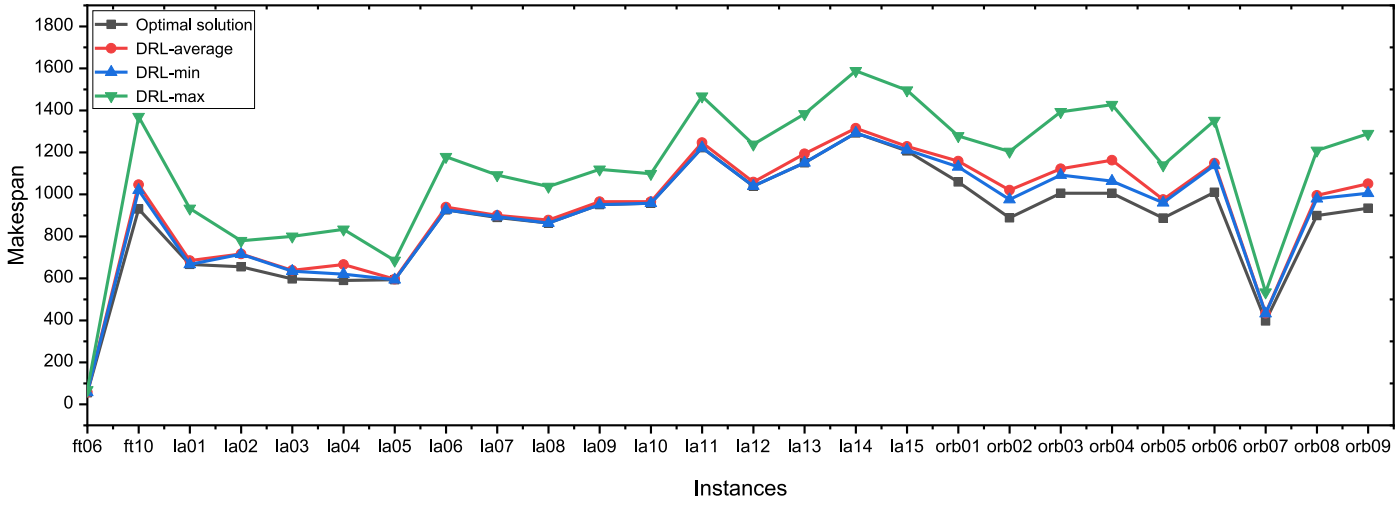

Fig. 7. Makespan with DRL method on OR instances.

Table 2
Simulation parameters.

Algorithm Parameters	values
Proximal Policy Optimization	
Learning rate	10^{-4}
Discount factor γ	0.99
Clip range	0.1
Entropy coefficient	0.0
Number of steps per update	512
Batch size	64
Number of training epochs per update	10
Genetic Algorithm	
Iteration	2000
Population size	30
Crossover probability	0.8
Mutation probability	0.2

episodes. The convergence of the reward indicates the effectiveness of the DRL model. We can see that the makespan can reach 57 in the later stage of training, which is quite close to the theoretical optimal solution 55 [24–26]. Due to the probabilistic nature, there will still be a small amount of fluctuation after the model converges, but it can be seen from Fig. 5 that the maximum makespan is 68 after the model converges, which is still very small compared with the makespan in the initial training stage. To be more rigorous, we will consider both the average value and the maximum and minimum values in the subsequent evaluation of makespan.

To further evaluate the performance of the proposed method, we take more instances and compare it with the following two methods:

(1) Heuristic rules: LPT, SPT, FIFO. These methods are based on a fixed rule to schedule the job, so there is no adaptability. Nevertheless, its decision-making is instant.
(2) Meta-heuristic algorithms: Genetic algorithm. This method has good performance and becomes a common method for solving the JSP. The disadvantage is that it needs to resolve when face different JSP problems, which takes much time on decision making.

We use 26 instances with different size to evaluate the above methods. There are ft06 instance with 6×6 size [24], ft10 instance with 10×10 [24] and orb01-orb09 instances with 10×10 [26], la01-la05 instances with 5×10, la06-la10 instances with 5×15, la11-la15 instances with 5×20 [25]. Table 3 and Fig. 6 shows the results of the five methods on these instances. For quantitative evaluation, we introduce

the theoretically optimal solutions [24–26]. Besides, as mentioned above, there is still a small fluctuation after the model converges, so the minimum makespan, maximum makespan and average makespan that the model can reach are considered, as shown in Fig. 7.

The algorithm parameters of the simulation design in this paper are shown in Table 2.

It can be obtained from Table 3 and Fig. 6 that the makespan with the proposed method is smaller than that with the heuristic rule method for all instances, and is comparable with the meta-heuristic method. It can be seen that the overall effect of DRL is better than the meta-heuristic method. On most instances, the makespan of DRL is smaller than that of GA; on a few instances, the makespan of the two methods are equal; and on only one instance, the makespan of DRL is higher than that of GA. Besides, compared with the optimal solution, most of the relative errors of DRL are below or about 10%, indicating that the proposed method can achieve an approximate theoretical optimal solution. Specifically, for instances with fewer machines, such as ft06 with six machines, la01-la15 with five machines, the relative error is mostly close to 0. Otherwise, for instances with more machines, such as orb01-orb09 and ft10 with ten machines, the relative error is greater than 5%. This is because the number of machines affects the dimension of the DRL agent's actions. The smaller the action dimension, the easier it is to converge to the optimal solution. Moreover, we can see that for la02-la04 instances, the relative error is large. Similarly, for other methods, there are also large errors on these three instances. This may be because the DRL model converged to the local optimal solution due to their unique environment built by these instances. The above results obtained by DRL are the values that the model can converge to.

In Fig. 7, the minimum makespan and the maximum makespan of DRL are obtained by testing 1000 times on the trained DRL model. It can be seen that the model after convergence still has certain fluctuations. However, the average makespan is very close to the minimum makespan and the best solution, indicating that there are very few fluctuations in 1000 times test. Besides, the maximum makespan is within an acceptable range compared with the optimal solution, and is about the same as the results obtained by heuristic rules and meta-heuristic methods. Therefore, we believe that the results obtained by DRL are impressive.

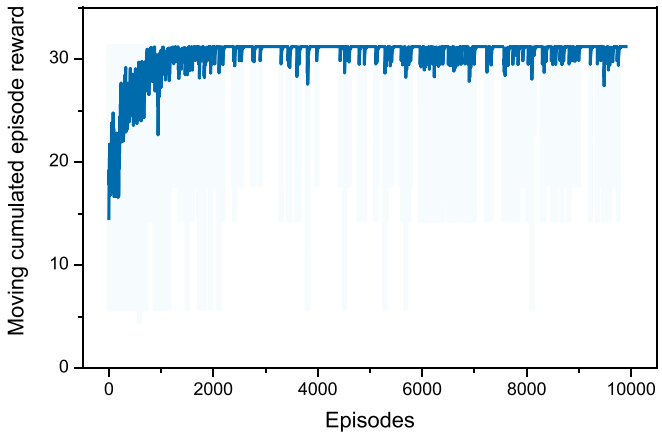

Fig. 8. Training process of J2.

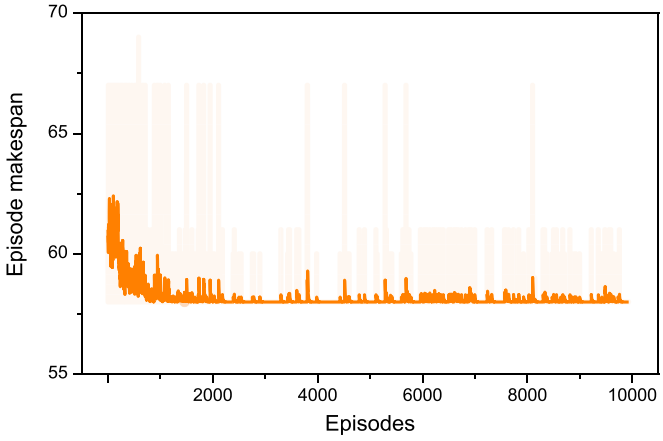

Fig. 9. Makespan curve of J2.

Table 3
Experimental results.

Instances	Optimal solution	SPT	LPT	FIFO	GA	DRL	Relative error (%)
ft06(6 × 6)	55	88	77	65	58	57	3.63
ft10(10×10)	930	1074	1295	1184	1157	1033	11.07
la01(5 × 10)	666	751	822	772	706	666	0
la02(5 × 10)	655	821	990	830	732	715	9.16
la03(5 × 10)	597	672	825	755	680	634	6.19
la04(5 × 10)	590	711	818	695	637	665	12.71
la05(5 × 10)	593	610	693	610	593	593	0
la06(5 × 15)	926	1200	1125	926	948	926	0
la07(5 × 15)	890	1034	1069	1088	985	894	0.44
la08(5 × 15)	863	942	1035	980	932	863	0
la09(5 × 15)	951	1045	1183	1018	1001	951	0
la10(5 × 15)	958	1049	1132	1006	958	958	0
la11(5 × 20)	1222	1473	1467	1272	1304	1222	0
la12(5 × 20)	1039	1203	1240	1039	1087	1039	0
la13(5 × 20)	1150	1275	1230	1199	1207	1150	0
la14(5 × 20)	1292	1427	1434	1292	1292	1292	0
la15(5 × 20)	1207	1339	1612	1587	1403	1212	0.41
Orb01(10×10)	1059	1478	1410	1368	1379	1131	6.79
Orb02(10×10)	888	1175	1293	1007	1141	993	11.82
Orb03(10×10)	1005	1179	1430	1405	1300	1092	8.65
Orb04(10×10)	1005	1236	1415	1325	1229	1118	11.24
Orb05(10×10)	887	1152	1099	1155	1135	972	9.58
Orb06(10×10)	1010	1190	1474	1330	1309	1140	12.87
Orb07(10×10)	397	504	470	475	505	432	8.81
Orb08(10×10)	899	1107	1176	1225	1174	979	8.89
Orb09(10×10)	934	1262	1286	1189	1158	1005	7.60

Table 4
Execution time.

	SPT	LPT	FIFO	GA	DRL
Execution time(s)	0.12	0.12	0.12	28.002	0.001
Makespan	89	77	68	58	67

4.2. Generalization and flexibility

Some existing approaches can find approximal optimal scheduling for one specific problem and have to reschedule when facing new problems. In a real-world job-shop environment, the production line flows one by one, and there may not be so much time to reschedule, which will delay the progress of subsequent jobs or cause errors. Our goal is to find an adaptive system to schedule jobs dynamically. It can generalize well when encountering unexpected situations such as machine breakdown or the change of the processing time and machine order of jobs. This paper uses deep reinforcement learning to schedule the job-shop adaptively. Below, we design a further experiment to explore the generalization abilities of the trained model.

For example, we randomly change the processing time and the machine sequence of job J_1 to generate a new instance J_2, to characterize emergencies such as machine failure or job rework, as shown in Eq. (8). It should be noted that J_2 must be similar to the experimental instance J_1 to some extent, at least the same in shape.

$$J_2 = \begin{bmatrix} (0,3) & (2,1) & (1,6) & (3,7) & (5,3) & (4,6) \\ (1,7) & (2,5) & (4,10) & (5.10) & (0,10) & (3,4) \\ (2,5) & (3,4) & (5,8) & (10,9) & (1,1) & (4,7) \\ (1,5) & (0,5) & (2,5) & (3,3) & (4,8) & (5,9) \\ (2,9) & (1,3) & (4,5) & (5,4) & (0,3) & (3,1) \\ (1,3) & (3,3) & (5,9) & (0,10) & (4,4) & (2,1) \end{bmatrix} \tag{8}$$

For new instance, we use the trained optimal policy to make a decision. Figs. 8 and 9 show the training process and the makespan change of J_2 using the trained model, respectively.

To compare the generalization performance of the DRL model, we consider the execution time and makespan of different algorithms. The execution time and makespan of these methods when facing new JSP problems are shown in Table 4.

It can be seen from Figs. 8 and 9 that when the model trained for J_1 is used for J_2, the makespan is about 67 in the beginning, this is the instant solution given by the current DRL model when responding to emergencies. With the training process continuing, the model can quickly converge again at 1500 episodes and find the optimal policy for instance J_2, and the makespan is stable at 58. This is because the deep reinforcement learning is based on experiential learning and has a strong learning ability. It can quickly find the optimal policy in the face of the changing environment.

In Table 4, we can see that the execution time of heuristic rules is 12 ms, so immediate decisions can be made when the job-shop environment changes. However, because fixed rules cannot be applied to each JSP problem, good results cannot be obtained. The meta-heuristic method can obtain a better solution after many iterations, but due to the high computational complexity, its decision-making time is 28 s, which is very long to respond to the changes in the job-shop environment. However, the proposed DRL method can make a decision better than the heuristic rules in 1 ms when dealing with emergencies, indicating the DRL model can adapt to the changes of the environment immediately, so as to achieve adaptive scheduling. Therefore, it can be considered that the trained model has a certain generalization ability, and it can deal with uncertain factors in the scheduling process. Besides, it needs only a few seconds to find the optimal policy again. Using the online DRL method, we can choose to obtain an immediate decision or an optimal decision according to specific needs.

5. Conclusion

We studied the job-shop scheduling problem in smart manufacturing. A dynamic adaptive scheduling system using deep reinforcement learning is introduced to cope with the unexpected conditions in a real-world environment. For the disaster problem of the state and action space dimensions caused by the increase in the scale of the problem, the PPO algorithm is used to reduce the complexity significantly while achieving good performance. To evaluate the performance of the proposed DRL model, we compare its results with heuristic rules and meta-heuristic algorithm. The experimental results show that, compared with the above methods, the proposed method can not only obtain comparative results but also can realize adaptive scheduling. Furthermore, the simulation results demonstrate that the proposed model has a certain generalization capability when facing unexpected situations.

Author statement

Libing Wang: Methodology, Writing - Original Draft, Software. **Xin Hu:** Conceptualization, Writing - Review & Editing. **Yin Wang:** Investigation, Writing - Review & Editing. **Sujie Xu:** Validation, Formal analysis. **Shijun Ma:** Supervision, Investigation. **Kexin Yang:** Formal analysis, Writing - Review & Editing. **Zhijun Liu:** Supervision, Data Curation. **Weidong Wang:** Resources, Supervision.

Declaration of Competing Interest

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

References

[1] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, Math. Oper. Res. 1 (2) (May 1976) 117–129.
[2] M. Miki, T. Hiroyasu, J. Wako, T. Yoshida, Adaptive temperature schedule determined by genetic algorithm for parallel simulated annealing, in: The 2003 Congress on Evolutionary Computation, 2003. CEC '03, Canberra, ACT, Australia 1, 2003, pp. 459–466, https://doi.org/10.1109/CEC.2003.1299611. .
[3] S. Horn, G. Weigert, E. Beier, Heuristic optimization strategies for scheduling of manufacturing processes, in: 2006 29th International Spring Seminar on Electronics Technology, 2006, pp. 422–427, https://doi.org/10.1109/ISSE.2006.365142. St. Marienthal.
[4] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop problem, Manage. Sci. 42 (1996) 797–813.
[5] B. Ombuki, M. Ventresca, Local search genetic algorithms for the job shop scheduling problem, Appl. Intell. 21 (2004) 99–109.
[6] Y. Zhan, C. Qiu, Genetic algorithm application to the hybrid flow shop scheduling problem, in: 2008 IEEE International Conference on Mechatronics and Automation, Takamatsu, 2008, pp. 649–653, https://doi.org/10.1109/ICMA.2008.4798833.
[7] T. Xuan Tung, T. Dung Ngo, Socially aware robot navigation using deep reinforcement learning, in: 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec City, QC, 2018, pp. 1–5, https://doi.org/10.1109/CCECE.2018.8447854.
[8] M. Gromniak, J. Stenzel, Deep reinforcement learning for mobile robot navigation, in: 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Nagoya, Japan, 2019, pp. 68–73, https://doi.org/10.1109/ACIRS.2019.8935944.
[9] X. Hu, Z.Liu X.Liao, X.Ding S.Liu, M. Helaoui W. Wang, F.M. Ghannouchi, Multi-agent deep reinforcement learning-based flexible satellite payload for mobile terminals, IEEE Trans. Veh. Technol. (2020), https://doi.org/10.1109/TVT.2020.3002983.

[10] X. Hu, Y. Zhang, X. Liao, Z. Liu, W. Wang, F.M. Ghannouchi, Dynamic beam hopping method based on multi-objective deep reinforcement learning for next generation satellite broadband systems, IEEE Trans. Broadcast. 66 (3) (Sept. 2020) 630–646.
[11] X. Liao, Z.Liu X.Hu, S. Ma, X.Li L.Xu, W. Wang, F.M. Ghannouchi, Distributed intelligence: a verification for multi-agent DRL based multibeam satellite resource allocation, IEEE Commun. Lett. (2020), https://doi.org/10.1109/LCOMM.2020.3019437.
[12] X. Hu, S. Liu, R. Chen, et al., A deep reinforcement learning based framework for dynamic resource allocation in multibeam satellite systems, IEEE Commun. Lett. 22 (8) (Aug. 2018) 1612–1615.
[13] X. Hu, Shuaijun Liu, Yipeng Wang, Lexi Xu, Yuchen Zhang, Cheng Wang, Weidong Wang, Deep reinforcement learning-based beam Hopping algorithm in multibeam satellite systems, IET Commun. 13 (16) (Jun. 2019) 2485–2491.
[14] Shuaijun Liu, Xin Hu, Weidong Wang, Deep reinforcement learning based dynamic channel allocation algorithm in multibeam satellite systems, IEEE Access 6 (2018) 15733–15742.
[15] B. Simsek, S. Albayrak, A. Korth, Reinforcement learning for procurement agents of factory of the future, in: Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA 2, 2004, pp. 1331–1337, https://doi.org/10.1109/CEC.2004.1331051 (IEEE Cat. No.04TH8753).
[16] S. Qu, T. Chu, J. Wang, J. Leckie, W. Jian, A centralized reinforcement learning approach for proactive scheduling in manufacturing, in: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 2015, pp. 1–8, https://doi.org/10.1109/ETFA.2015.7301417.
[17] E. Schmidl, E. Fischer, M. Wenk, J. Franke, Knowledge-based generation of a plant-specific reinforcement learning framework for energy reduction of production plants, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 1–4, https://doi.org/10.1109/ETFA46521.2020.9211957.
[18] O. Kundu, S. Dutta, S. Kumar, Deep-pack: a vision-based 2D online bin packing algorithm with deep reinforcement learning, in: 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 2019, pp. 1–7, https://doi.org/10.1109/RO-MAN46459.2019.8956393.
[19] S. Fairee, C. Khompatraporn, S. Prom-on, B. Sirinaovakul, Combinatorial artificial bee colony optimization with reinforcement learning updating for travelling salesman problem, in: 2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Pattaya, Chonburi, Thailand, 2019, pp. 93–96, https://doi.org/10.1109/ECTI-CON47248.2019.8955176.
[20] J.J.Q. Yu, W. Yu, J. Gu, Online vehicle routing with neural combinatorial optimization and deep reinforcement learning, in: IEEE Transactions on Intelligent Transportation Systems 20, Oct. 2019, pp. 3806–3817, https://doi.org/10.1109/TITS.2019.2909109.
[21] J. Wang, J. Hu, G. Min, A. Zomaya, N. Georgalas, Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Trans. Parallel Distrib. Syst. 32 (1) (2021) 242–253.
[22] Z. Chen, J. Hu, G. Min, A. Zomaya, T. El-Ghazawi, Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning, IEEE Trans. Parallel Distrib. Syst. 31 (4) (2020) 923–934.
[23] J.E. Beasley, 'OR-library: distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (Nov. 1990) 1069.
[24] J.F. Muth, G.L. Thompson, Industrial Scheduling, Prentice-Hall, Englewood Cliffs, NJ, 1963.
[25] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.
[26] D. Applegate, W. Cook, A computational study of job-shop scheduling, ORSA J. Comput. 3 (2) (1991) 149–156.

Libing Wang She received the B.Sc. degree in Electronic and Information Engineering from Henan Polytechnic University, China, in 2019, and she is currently pursuing the M.Sc degrees in Electronics and Communications Electronics from Beijing University of Posts and Telecommunications, China. Her current research interests include dynamic satellite resource management based on deep reinforcement learning algorithm.

Xin Hu He earned the B.Sc. and Ph.D. degrees in electrical information engineering from Huazhong University of Science and Technology , Wuhan, China, and Institute of Electrics, Chinese Academy of Sciences, Beijing, China, in 2007 and 2012, respectively. His current research interests include dynamic wireless resources management, multibeam satellite communications and deep reinforcement learning applications.

Yin Wang He received the B.Sc. degree in Communication Engineering from Hebei University, China, in 2020. He is currently pursuing the M.Sc degrees in Electronics Science and Technology from Beijing University of Posts and Telecommunications, China. His research interests include satellite resource management based on deep reinforcement learning.

Sujie Xu She received the B.Sc. degree in Electronics Science and Technology from Beijing University of Posts and Telecommunications, China, in 2019, and she is currently pursuing the M.Sc degrees in Science and Technology Engineering from Beijing University of Posts and Telecommunications, China. Her current research interests include dynamic satellite resource management based on deep reinforcement learning algorithm.

Shijun Ma He received the B.Sc. degree in Electronics Science and Technology from Chongqing University of Posts and Telecommunications, China, in 2018, and he is currently pursuing the M.Sc degrees in Electronics and Communications Engineering from Beijing University of Posts and Telecommunications, China. His current research interests include dynamic satellite resource management based on deep reinforcement learning algorithm.

Kexin Yang She is currently pursuing the B.Sc. degree in E-commerce with Law from Beijing University of Posts and Telecommunications, China. Her current research interests include application of deep learning in business and management and investment strategy theory and resource management based on deep reinforcement learning.

Zhijun Liu He received the B.Sc. degree in Electronics Science and Technology from Nanjing University of Posts and Telecommunications, China, in 2017, and he is currently pursuing the Ph.D. degrees in Electronics Science and Technology from Beijing University of Posts and Telecommunications, China. His research interests include digital predistortion of nonlinear power amplifiers and the application of deep learning techniques to RF and microwave problems.

Weidong Wang He received the Ph.D. degree from Beijing University of Posts and Telecommunications in 2002. He is now the Professor of School of Electronic Engineering at Beijing University of Posts and Telecommunications. He takes the role of expert of National Natural Science Foundation and member of China Association of Communication. His research interests include communication system, radio resource management, Internet of things and signal processing.