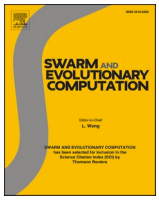
Swarm and Evolutionary Computation 94 (2025) 101902

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



Distributed heterogeneous flexible job-shop scheduling problem considering automated guided vehicle transportation via improved deep Q network

Minghai Yuan *, Songwei Lu, Liang Zheng, Qi Yu, Fengque Pei, Wenbin Gu

College of Mechanical and Electrical Engineering, Hohai University, Changzhou, PR China

ARTICLE INFO

Keywords:
Distributed heterogeneous flexible job-shop scheduling
AGV
Deep reinforcement learning
Deep Q network
Combination dispatching rule

ABSTRACT

Distributed manufacturing has become a research hotspot in the context of economic globalization. The distributed heterogeneous flexible job-shop scheduling problem considering automated guided vehicle transportation (DHFJSP-AGV) extends the classic flexible job-shop scheduling problem (FJSP) but remains underexplored. DHFJSP-AGV involves four subproblems: assigning jobs to heterogeneous factories, scheduling jobs to machines, sequencing operations on machines and transporting jobs between machines using AGVs. Due to its complexity, this study proposes an improved deep Q network (DQN) real-time scheduling method aimed at minimizing makespan. A mixed integer linear programming model (MILP) of DHFJSP-AGV is developed and transformed into a Markov decision process (MDP). Eight general state features are extracted and normalized to represent the state space, while appropriate combination dispatching rules are selected as the action space. The state features of each scheduling point are input to the DQN, determining the factory, job, machine, and AGV for each process. Additionally, double DQN and an improved ε-greedy exploration are used to enhance the DQN. Numerical comparison experiments under different production configurations and real-world application in distributed flexible job-shop with dynamic map environment demonstrate the effectiveness and generalization capabilities of improved DQN.

1. Introduction

Recently, intense competition and high equipment costs have shifted production structures from centralized to distributed systems [1]. Distributed manufacturing optimally utilizes existing resources through labor division and cooperation, effectively reducing costs and managing risks [2]. Distributed heterogeneous flexible job-shop scheduling problem (DHFJSP) widely exists in industries such as automotive and semiconductor manufacturing. It is characterized by factory heterogeneity, including variations in machine technology, transportation conditions, and production capacity [3]. This heterogeneity significantly improves production efficiency while complicates resource management, requiring more flexible scheduling strategies. Compared to the flexible job-shop scheduling problem (FJSP), DHFJSP is more challenging as it allows jobs to be processed in different heterogeneous factories, leading to more complex topologies and larger solution space [4]. With the rise of Industry 5.0, automatic guided vehicles (AGV) are increasingly employed in modern manufacturing systems. As intelligent transportation devices, AGVs enhance efficiency by automatically navigating and transporting materials [5]. In practice, transportation times between machines must not be overlooked. Thus, incorporating AGV transportation in DHFJSP better aligns with real-world production requirements.

While FJSP is known to be NP-hard, the DHFJSP considering AGV transportation (DHFJSP-AGV) presents even greater complexity due to considering factory and AGV selection [6]. Recent advances in deep reinforcement learning (DRL), particularly deep Q network (DQN), offer promising solutions to DHFJSP-AGV. This paper applies an improved DQN to address DHFJSP-AGV, designing adaptable state features for various production scenarios and systematically solving through combination dispatching rules. The trained agent adeptly navigates diverse production scenarios.

The main contributions are listed as follows: 1) A mixed integer linear programming model (MILP) for DHFJSP-AGV is developed, aiming to minimizing makespan. 2) Eight state features are designed and normalized to [0,1] for each scheduling point. Suitable combination

* Corresponding author.
E-mail addresses: ymhai@hhu.edu.cn (M. Yuan), 231319010011@hhu.edu.cn (S. Lu), fq_pei@hhu.edu.cn (F. Pei), guwenbing@hhuc.edu.cn (W. Gu).

https://doi.org/10.1016/j.swevo.2025.101902
Received 11 August 2024; Received in revised form 12 January 2025; Accepted 2 March 2025
Available online 7 March 2025

dispatching rules are selected to simultaneously determine factory allocation, job selection, machine selection, and AGV allocation. 3) For the complex DHFJSP-AGV, double DQN and an improved dynamic exploration rate are introduced to enhance the basic DQN. 4) Numerical experiments across various production scenarios show that the improved DQN outperforms well-known combination dispatching rules, basic DQN and double DQN, while maintaining generalization ability in a real-world case.

The rest of this paper is organized as follows: Section 2 reviews the distributed FJSP (DFJSP) research, AGV scheduling and DRL methods. Section 3 provides a detailed overview of DHFJSP-AGV. In Section 4, an improved DQN framework is developed. The experimental results are discussed in Section 5. Section 6 concludes this paper.

2. Literature review

2.1. Distributed flexible job-shop scheduling

Scheduling problems focus on optimizing specific objectives by efficiently allocating jobs, selecting machines and sequencing operations. As an extension of the traditional FJSP, DFJSP has gained significant attention in recent years [7,8]. Xie *et al.* [9] proposed a hybrid genetic tabu search algorithm (HGTSA), combining the global search capability of genetic algorithm (GA) with the local search ability of tabu search (TS) to solve DFJSP challenges. To optimize the makespan in DFJSP, Meng *et al.* [10] developed a constraint programming (CP) model that integrates interval decision variables and domain filtering algorithms. Cheng and Du [11] introduced an energy-efficient DFJSP model with robotic transportation, employing a hybrid algorithm that combines artificial immune systems and variable neighborhood search techniques. For DFJSP considering energy consumption, Meng *et al.* [12] created a new MILP for small-scale instances, complemented by an enhanced shuffled frog leaping algorithm (HSFLA) for larger instances. Li *et al.* [13] improved the grey wolf optimizer (GWO) with novel encoding and decoding schemes for job assignment, factory selection, machine selection and operation scheduling in DFJSP. Zhu *et al.* [14] designed an improved memetic algorithm, featuring a five-layer coding structure and tailored initialization methods for the dynamic DFJSP with operation inspection. Luo et al. [15] addressed the multi-objective DFJSP with job transfers between factories, proposing an enhanced memetic algorithm featuring advanced chromosome representations and three effective neighborhood structures to minimize makespan, workload, and energy consumption.

While these studies have advanced DFJSP research, they generally assume factory homogeneity or do not explicitly address factory heterogeneity. In real-world production, significant differences in machine capabilities, transportation conditions, and production capacities between factories increase scheduling complexity, necessitating strategies that account for such heterogeneity. Li *et al.* [16] proposed a DQN-based co-evolution algorithm for energy-aware DHFJSP, aiming to optimize both energy consumption and makespan. Zhang *et al.* [17] studied the DHFJSP with varying processing times using a DQN to guide the design of a variable neighborhood search algorithm. Chen *et al.* [18] developed a low-carbon DHFJSP model incorporating factory heterogeneity, job insertion and transfer, and low-carbon goals, solved via a DRL framework. For dual resource constraints problem, Zhang *et al.* [19] considered human factors in various distributed factories and developed a Q-learning-based multi-objective grey wolf optimization algorithm, using Q-learning strategies and factory neighborhood structures to enhance solution space exploration.

2.2. AGV scheduling

In scheduling problems that incorporate transportation elements such as AGVs, common methods include GA [20] and particle swarm optimization [21,22], among others. For FJSP with multiple AGVs, Wen *et al.* [23] developed an effective hybrid algorithm with improved elitist strategy and neighborhood search function based on problem knowledge to minimize makespan, AGV running time and machine load. Han *et al.* [24] proposed a new MILP and a dual-population cooperative genetic algorithm (DCGA) to solve the integration problem of FJSP and AGV under minimizing the makespan. Meng *et al.* [25] solved the scheduling problem of considering a limited number of AGVs, and designed an improved genetic algorithm (IGA) to minimize the makespan. Xin *et al.* [26] proposed a hybrid algorithm of multi-view modelling to solve the flexible job-shop collaborative scheduling problem (FJCSP) by combining the estimation of distribution algorithm and ant colony optimization algorithm. Zhou *et al.* [27] constructed a green low-carbon FJSP with multiple transportation equipment, and proposes a heuristic strategy NSGA-II for solving the problem using real-encoded embedding loops to replace duplicate individuals. Lei *et al.* [28] developed a memetic algorithm integrated with waiting time calculation method for flexible flow-shop scheduling problem (FFSP) with dynamic transportation waiting time. Crane transportation similar to AGV has also been widely studied in the scheduling problem [29,30,31]. Du *et al.* [32] used a DQN model to address the multi-objective FJSP with crane transportation and setup time, designing eight crane transportation stages, three machine states, and multiple optimization strategies. Aiming at the FJSP with crane transportation, Li et al. [33] proposed an efficient hybrid optimization algorithm by combining iterative greedy and simulated annealing algorithm. In addition, in order to balance the exploration ability and time complexity of the algorithm, an exploration heuristic algorithm for specific problems is developed. An *et al.* [34] considered an FJSP with forklift transportation and imperfect cutting tool maintenance solved by a hybrid multi-objective EA.

2.3. DRL methods

Most of the algorithms in the existing research are based on the classical evolutionary algorithms to improve their local search strategies to obtain high-quality solutions [35]. However, the scheduling optimization under the distributed manufacturing mode has the characteristics of large scale, many sub-problems and strong coupling. Especially further considering AGV transportation, the traditional meta-heuristic algorithm often shows the disadvantages of slow convergence speed and weak generalization in solving such problems.

In recent years, machine learning (ML), particularly deep learning (DL), has emerged as a powerful tool in fields such as optimization, fault diagnosis, and data prediction [36,37]. Nguyen Ngoc *et al.* [38] introduced an innovative structural health monitoring (SHM) method for truss bridges by combining a deep neural network (DNN) with the evolutionary artificial rabbit optimization algorithm (EVARO). For FJSP, based on the new heterogeneous graph representation of scheduling state, Wen *et al.* [39] proposed a framework based on heterogeneous graph neural network (GNN) to capture the complex relationship between operations and machines. Tran *et al.* [40] developed a hybrid approach using an artificial neural network (ANN) and balanced composite motion optimization (BCMO) to solve the optimization problems and predict the random vibration and buckling behavior of functionally graded porous microplates with material performance uncertainties.

DRL represented by DQN, effectively combines DL with reinforcement learning (RL), making it particularly well-suited for addressing complex combinatorial optimization problems due to its ability to adaptively select feasible actions at each scheduling point. Aiming at the DFJSP with insufficient transportation resources, Zhang *et al.* [41] proposed a framework based on heterogeneous GNN and DRL and considered the influence of different AGV distribution modes on scheduling objectives. Zhang *et al.* [42] proposed a memetic algorithm based on DQN for the energy-efficient FJSP with multiple AGVs. The strength Pareto evolutionary algorithm (SPEA2) is used to quickly explore the target space and the DRL is used to help the evolutionary algorithm select the optimal operator. Wang *et al.* [43] proposed a

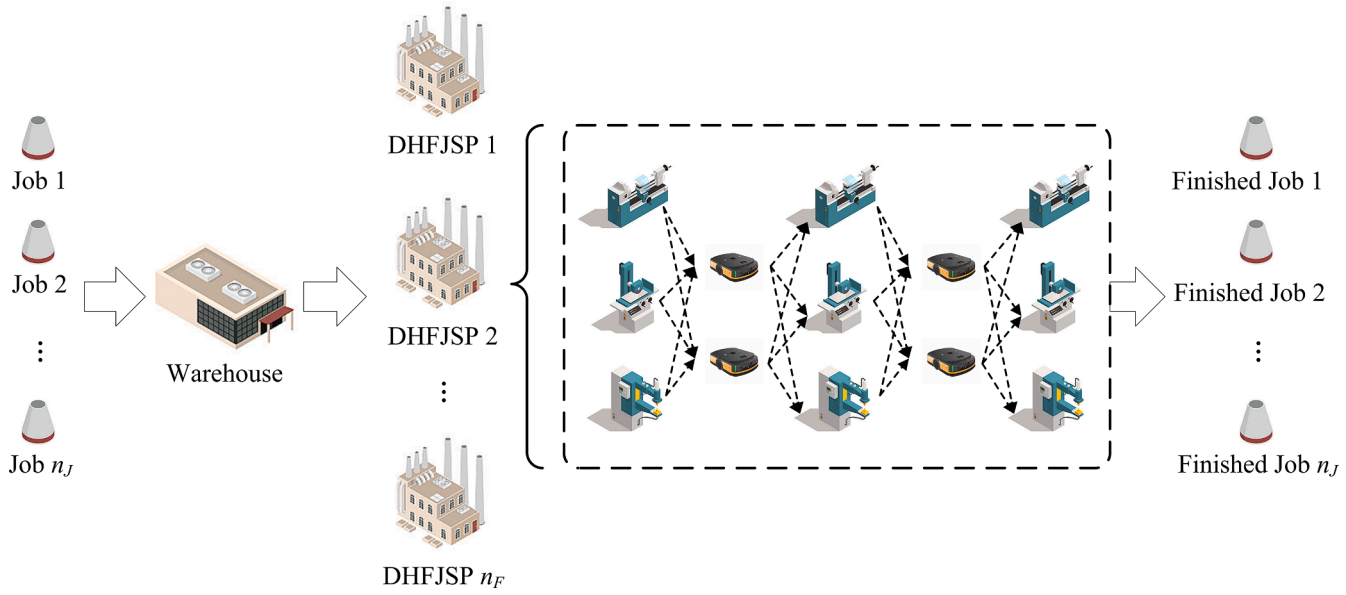


Fig. 1. Schematic diagram of DHFJSP-AGV.

Table 1
Processing time of four jobs in two heterogeneous factories.

Job	Operation	F_1			F_2			AGV
		M_{11}	M_{12}	M_{13}	M_{21}	M_{22}	M_{23}	
J_1	O_{11}	–	18	23	–	17	20	A_{11}
	O_{12}	27	–	30	24	18	–	A_{21}
J_2	O_{21}	–	27	21	–	33	25	
	O_{22}	19	30	–	21	25	–	
J_3	O_{31}	33	–	29	21	–	18	
	O_{32}	–	30	25	27	19	–	
	O_{33}	24	–	24	–	24	19	
J_4	O_{41}	19	–	30	25	17	21	
	O_{42}	18	25	–	19	–	24	

Table 2
Transportation time between different machines in DHFJSP-AGV.

	LU	M_1	M_2	M_3
$\boldsymbol{LU}$	0	6	5	6
$\boldsymbol{M_1}$	3	0	2	7
$\boldsymbol{M_2}$	6	10	0	4
$\boldsymbol{M_3}$	7	4	8	0

dynamic multi-objective FJSP to simulate the real production environment, designing a DRL-based scheduling algorithm. By utilizing two DQNs and a real-time processing framework, this method efficiently handles dynamic events and generates complete scheduling solutions. Chen *et al.* [44] introduced an RL framework for the classical job-shop scheduling problem (JSSP), integrating node2vec embedding with an enhanced multi-head attention Transformer. This approach effectively captures scheduling features and delivers high-quality solutions, showing strong performance on large-scale problems.

In summary, few studies have applied DRL methods to solve DHFJSP, and to the best of our knowledge, no previous studies on DHFJSP have considered AGV transportation. Therefore, our study addresses this gap by utilizing an improved DQN to solve the DHFJSP-AGV with the objective of minimizing makespan.

3. Problem description

3.1. Description of DHFJSP-AGV

As shown in Fig. 1, DHFJSP-AGV can be described as: n_J jobs are processed in n_F heterogeneous factories. Each factory has n_M machines and n_A AGVs. The number of machines and AGVs in every factory is the same, but the processing capacity of machines is different. After the job is assigned to the factory, it cannot be transferred to other factories for processing. Each job contains n_{J_i} operations that should be processed according to the predetermined sequence. Each operation can be processed on a set of candidate machines $\mathcal{M}_{ij}$. The job is transported to different machines for processing through AGV. In the DHFJSP-AGV, it is necessary to reasonably select factories, jobs, machines and AGVs to achieve the scheduling goal of minimizing the makespan.

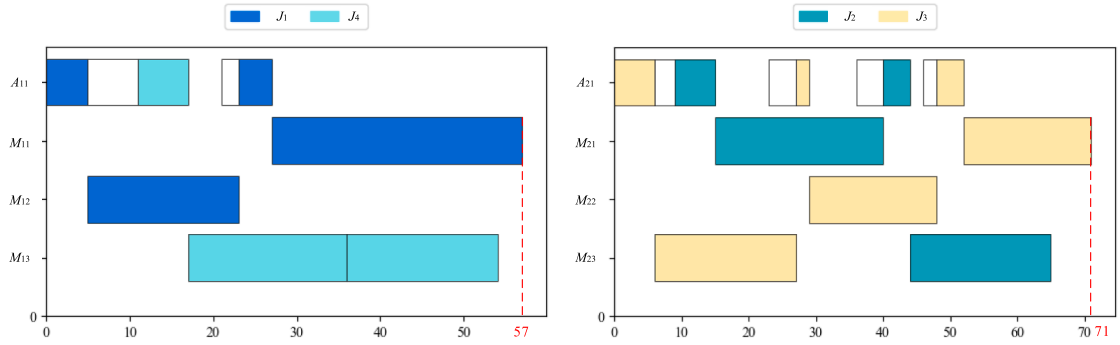
To clearly illustrate DHFJSP-AGV, Table 1 presents a simple instance with four jobs processed across two heterogeneous factories, each equipped with one AGV and three machines. For example, the processing time for the first operation O_{31} of job J_3 on machine M_{11} is 33, while on machine M_{21}, it is 21. A "–" indicates that the operation cannot be processed on the respective machine. Table 2 provides the transportation time for the AGVs between different machines within each factory, assuming that both factories share an identical machine layout and follow a common transportation schedule. LU denotes the loading and unloading station, where jobs are initially stored. For example, the transportation time of AGV from machine M_1 to machine M_2 is 2, while the time to LU is 3.

Fig. 2 illustrates a feasible scheduling Gantt chart. White blocks represent the AGV no-load time, while colored blocks represent machine processing times or AGV transport times for each job. The maximum completion time for factory F_1 is 57, and for factory F_2 is 71, resulting in a total maximum completion time of 71. Table 3 lists the notations used in DHFJSP-AGV.

3.2. Mathematical model of DHFJSP-AGV

The assumptions in this article are described as follows:

(1) All factories, jobs, machines and AGVs are in a ready state at the initial time.



(a) F_1 (b) F_2

Fig. 2. A feasible scheduling Gantt chart of DHFJSP-AGV.

Table 3
Notations used in DHFJSP-AGV.

Notations	Description
i	Index of job, $i \in \{1,2,\cdots,n_J\}$
j	Index of operation, $j \in \{1,2,\cdots,n_{O_i}\}$
$.f.$	Index of factory, $f \in \{1,2,\cdots,n_F\}$
k	Index of machine, $k \in \{1,2,\cdots,n_M\}$
$.l.$	Index of AGV, $l \in \{1,2,\cdots,n_A\}$
J_i	The ith job
O_{ij}	The jth operation belonging to the job i
F_f	The fth factory
M_k	The kth machine
A_l	The lth AGV
$\mathscr{O}_i$	Operation set of job i
$\mathscr{M}_{ij}$	The optional machine set of operation O_{ij}
CTJ_i	The makespan of operation O_{ij}
CTF_f	The makespan of factory F_f
CTM_k	The makespan of machine M_k
CTA_l	The makespan of AGV A_l
UM_k	Utilization of machine M_k
UA_l	Utilization of AGV A_l
CO_i	The number of operations processed in job i
X_{if}	1: The job J_i is processed in factory F_f 0: The job J_i is not processed in factory F_f
Y_{ijl}	1: The job J_i is transported by AGV A_l 0: The job J_i is not transported by AGV A_l
Z_{ijk}	1: The job J_i is processed on machine M_k 0: The job J_i is not processed on machine M_k
$YP_{ijf'l}$	1: If operation O_{ij} is transported by AGV A_l earlier than operation $O_{i'f'}$ 0: If operation O_{ij} is transported by AGV A_l later than operation $O_{i'f'}$
$ZP_{ijf'k}$	1: If operation O_{ij} is processed by M_k earlier than operation $O_{i'f'}$ 0: If operation O_{ij} is processed by M_k later than operation $O_{i'f'}$
pt_{ijfk}	The processing time of operation O_{ij} on machine M_k of factory F_f
pt^s_{ijfk}	The start processing time of operation O_{ij} on machine M_k of factory F_f
pt^e_{ijfk}	The end processing time of operation O_{ij} on machine M_k of factory F_f
$tt_{ikk'}$	The transportation time of job J_i from machine M_k to machine $M_{k'}$
et_{ijl}	The no-load transportation time after operation O_{ij} is assigned to AGV A_l
et^s_{ijl}	The no-load transportation start time after operation O_{ij} is assigned to AGV A_l
et^e_{ijl}	The no-load transportation end time after operation O_{ij} is assigned to AGV A_l
lt_{ijl}	The load transportation time after operation O_{ij} is assigned to AGV A_l
lt^s_{ijl}	The load transportation start time after operation O_{ij} is assigned to AGV A_l
lt^e_{ijl}	The load transportation end time after operation O_{ij} is assigned to AGV A_l

(2) Each machine can only process one job at the same time, and the processing process will not be terminated in the middle.
(3) The processing time of the job on the machine is certain, not fuzzy.
(4) At the same time, each job can only be processed on one machine in one factory, and once the job is assigned to a factory, it cannot be transferred to other factories.
(5) The job is transported by AGV to different machines for processing.
(6) Each factory has the same machine layout and shares a common transportation schedule.
(7) The number of machines and AGVs in heterogeneous factories is the same, but the processing capacity of the machines between different factories is different.
(8) The time of the same machine processing different operations is generally different, and not every machine can handle all operations.
(9) The buffer of the machine is set to infinity.

The objective function of this paper is to minimize the makespan described by Eq. (1), where CTJ_i denotes the completion time of jobJ_i.

$$f = \min(\max_{1\le i\le n_J} CTJ_i) \tag{1}$$

The required constraints are described as follows:

Eq. (2) indicates that each job can only select one factory for processing.

$$\sum_{f=1}^{n_F} X_{if} = 1 \tag{2}$$

Eq. (3) indicates that each job can only be transported by one AGV at the same time.

$$\sum_{l=1}^{n_A} Y_{ijl} = 1 \tag{3}$$

Eq. (4) indicates that each job can only be processed on one machine at the same time.

$$\sum_{k=1}^{n_M} Z_{ijk} = 1 \tag{4}$$

Eq. (5) indicates that when the operation is assigned to the AGV, the no-load transportation start time of AGV cannot be earlier than the start processing time of the operation.

$$et^s_{ijl} \geqslant pt^s_{ijfk} \tag{5}$$

Eq. (6) indicates that the no-load transportation start time of AGV is not earlier than the load transportation end time of AGV in the previous operation.

$$et^s_{ijw} \geqslant lt^e_{i'j'l} \times \left(1 - YP_{iji'j'l}\right) \tag{6}$$

Eq. (7) indicates that the no-load transportation end time of AGV is the sum of the no-load transportation start time of AGV and the no-load transportation time of AGV, that is, once AGV starts no-load transportation, it cannot be interrupted.

$$et^e_{ijl} = et^s_{ijl} + et_{ijl} \tag{7}$$

Table 4
Feature and description of the state space.

State	Feature	Description
factory	S_{11}	Average completion rate of jobs in the factory
	S_{12}	Standard deviation of the job completion rate in the factory
job	S_{21}	Average completion rate of operations
	S_{22}	Standard deviation of the operation completion rate
machine	S_{31}	Average utilization rate of machines
	S_{32}	Standard deviation of the machine utilization rate
AGV	S_{41}	Average utilization rate of AGVs
	S_{42}	Standard deviation of the AGV utilization rate

Eq. (8) indicates that the load transportation start time of AGV is not earlier than the no-load transportation end time of AGV.

$$lt^s_{ijl} \geqslant et^e_{ijl} \tag{8}$$

Eq. (9) indicates that the load transportation start time of the AGV is not earlier than the end processing time of the previous operation.

$$lt^s_{ijl} \geqslant pt^e_{i(j-1)fk} \tag{9}$$

Eq. (10) indicates that the load transportation end time of AGV is equal to the sum of the load transportation start time of AGV and the load transportation time of AGV, that is, once AGV starts load transportation, it cannot be interrupted.

$$lt^e_{ijl} = lt^s_{ijl} + lt_{ijl} \tag{10}$$

Eq. (11) indicates that the start processing time of the operation on the machine is not earlier than the load transportation end time of AGV.

$$pt^s_{ijfk} \geqslant lt^e_{ijl} \tag{11}$$

Eq. (12) indicates that the start processing time of the operation on the machine is not earlier than the end processing time of the previous operation of the machine.

$$pt^s_{ijfk} \geqslant pt^e_{i'j'fk} \times \left(1 - ZP_{iji'j'k}\right) \tag{12}$$

Eq. (13) indicates that the end processing time of the operation is equal to the sum of the start processing time of the operation and the processing time of the operation, that is, the job cannot be interrupted once the processing begins.

$$pt^e_{ijfk} = pt^s_{ijfk} + pt_{ijfk} \tag{13}$$

Eq. (14) indicates that the completion time of the job is the end time of the last operation of the job.

$$CTJ_i = pt^e_{in_{O_i}fk} \tag{14}$$

4. Proposed method

In this section, we first design the state space and action space of the problem, then define the reward function. Finally, we give the scheduling framework of DHFJSP-AGV and training method based on improved DQN.

4.1. Definition of state space

In the DHFJSP-AGV, the state space should consider the state information of factory, job, machine and AGV at the same time. Therefore, the features and description of the state space corresponding to the four resources of factory, job, machine and AGV are designed, as shown in Table 4. The state space is defined as $S = [S_{11}, S_{12}, S_{21}, S_{22}, S_{31}, S_{32}, S_{41}, S_{42}]$, where $[S_{11}, S_{12}]$ is the state feature set of the factory, $[S_{21}, S_{22}]$ is the state feature set of the job, $[S_{31}, S_{32}]$ is the state feature set of the machine, $[S_{41}, S_{42}]$ is the state feature set of the AGV. At the same time, in order to improve the generalization ability and stability of the algorithm in different workshops, all state features are normalized to [0, 1], so as to avoid the excessive influence of some state features on model training.

The equation of factory state features, job state features, machine state features and AGV state features are defined as follows:

The completion rate of the jobs in the factory F_f at time t is shown in Eq. (15), where X_{if} is the decision variable of whether the job J_i is assigned to the factory F_f, $CO_i(t)$ is the number of operations that the job J_i has completed at time t, $|\mathscr{O}_i|$ is the total number of operations of the job J_i, and $\lfloor\rfloor$ is the downward rounding symbol.

$$CRJ_f(t) = \frac{\sum_{i=1}^{n_J} X_{if} \cdot \left\lfloor \frac{CO_i(t)}{|\mathscr{O}_i|} \right\rfloor}{n_J} \tag{15}$$

Eq. (16) is the first state feature of the factory, which represents the average completion rate of jobs in the factory.

$$S_{11} = \frac{\sum_{f=1}^{n_F} CRJ_f(t)}{n_F} \tag{16}$$

Eq. (17) is the second state feature of the factory, which represents the standard deviation of the job completion rate in the factory.

$$S_{12} = \sqrt{\frac{\sum_{i=1}^{n_F} \left(CRJ_f(t) - S_{1,1}\right)^2}{n_F}} \tag{17}$$

Eq. (18) is the first state characteristic of the job, which represents the average completion rate of the operations.

$$S_{21} = \frac{\sum_{i=1}^{n_J} CO_i(t)}{\sum_{i=1}^{n_J} |\mathscr{O}_i|} \tag{18}$$

Eq. (19) is the second state feature of the job, which represents the standard deviation of the operation completion rate.

$$S_{22} = \sqrt{\frac{\sum_{i=1}^{n_J} \left(\frac{CO_i(t)}{|\mathscr{O}_i|} - S_{1,1}\right)^2}{n_J}} \tag{19}$$

The utilization rate of the machine at time t is shown in Eq. (20), where pt_{ijk} is the processing time of the operation O_{ij} on the machine M_k, Z_{ijk} is the decision variable of whether the operation $O_{ij}$is processed on the machine M_k, and $CTM_k(t)$ is the maximum completion time of the machine M_k at time t.

$$UM_k(t) = \frac{\sum_{i}^{n_J} \sum_{j}^{n_{J_i}} pt_{ijk} \cdot Z_{ijk}}{CTM_k(t)} \tag{20}$$

Eq. (21) is the first state feature of the machine, which represents the average utilization rate of machines.

$$S_{31} = \frac{\sum_{k=1}^{n_M} UM_k(t)}{n_M} \tag{21}$$

Eq. (22) is the second state feature of the machine, which represents the standard deviation of the machine utilization rate.

$$S_{32} = \sqrt{\frac{\sum_{k=1}^{n_M} \left(UM_k(t) - S_{2,1}\right)^2}{n_M}} \tag{22}$$

The utilization rate of AGV at time t is shown in Eq. (23), where et_{ijl} is the no-load transportation time when AGV A_l travels to the processing location of the operation O_{ij}, lt_{ijl} is the load transportation time of the operation O_{ij} on AGV A_l, Y_{ijl} is the decision variable of whether the

Table 5
The set of dispatching rules.

Type	Name	Description
Factory dispatching rule	$MinCT_F$	Select the factory with the shortest completion time
	$MaxCT_F$	Select the factory with the longest completion time
	$MinNJ_F$	Select the factory with the least allocated jobs
	$MaxNJ_F$	Select the factory with the most allocated jobs
Job dispatching rule	SPT_J	Select the job with the shortest processing time
	LPT_J	Select the job with the longest processing time
	$SPTR_J$	Select the job with the shortest remaining processing time
	$LPTR_J$	Select the job with the longest remaining processing time
	$SPTSO_J$	Select the job with the shortest processing time of subsequent operation
	$LPTSO_J$	Select the job with longest processing time of subsequent operation
	$LRUO_J$	Select the job with the least remaining unprocessed operations
	$MRUO_J$	Select the job with the most remaining unprocessed operations
Machine dispatching rule	SPT_M	Select the machine with the shortest processing time
	LPT_M	Select the machine with the longest processing time
	LU_M	Select the machine with the lowest utilization rate
	HU_M	Select the machine with the highest utilization rate
	$MinCT_M$	Select the machine with the minimum completion time
	$MaxCT_M$	Select the machine with the maximum completion time
AGV dispatching rule	STT_A	Select the AGV with the shortest transportation time
	LTT_A	Select the AGV with the longest transportation time
	LU_A	Select the AGV with the lowest utilization rate
	HU_A	Select the AGV with the highest utilization rate
	$MinCT_A$	Select the AGV with the minimum completion time
	$MaxCT_A$	Select the AGV with the maximum completion time

operation O_{ij} is transported by AGV A_l, and $CTA_l(t)$ is the maximum completion time of AGV A_l at time t.

$$UA_l(t) = \frac{\sum_{i}^{n_J}\sum_{j}^{n_{J_i}}\left(et_{ijl} + lt_{ijl}\right)\cdot Y_{ijl}}{CTA_l(t)} \tag{23}$$

Eq. (24) is the first state feature of AGV, which represents the average utilization rate of AGVs.

$$S_{41} = \frac{\sum_{l=1}^{n_A} UA_l(t)}{n_A} \tag{24}$$

Eq. (25) is the second state feature of AGV, which represents the standard deviation of the AGV utilization rate.

$$S_{42} = \sqrt{\frac{\sum_{l=1}^{n_A}\left(UA_l(t) - S_{3,1}\right)^2}{n_A}} \tag{25}$$

To avoid the dimensionality problem caused by using matrices to describe the state space, this paper adopts a state vector to represent the state at each scheduling point. This approach ensures that the dimension of the state space remains constant even when disturbances occur in the workshop, allowing the agent to demonstrate better generalization performance when handling instances of varying scales in different environments. The state vector designed in this paper is shown in Eq. (26).

$$S = (S_{11}, S_{12}, S_{21}, S_{22}, S_{31}, S_{32}, S_{41}, S_{42}) \tag{26}$$

4.2. Definition of action space

In the DHFJSP-AGV, factory allocation, job selection, machine allocation and AGV allocation must be considered at each scheduling point. Due to the complex constraints and large solution space inherent in DHFJSP-AGV, this paper adopts a combination of factory, job, machine and AGV dispatching rules as the action space, effectively leveraging expert knowledge. Unlike directly specifying factories, machines, jobs and AGVs, this approach uses a discrete action space, where the total number of possible combinations is fixed. This ensures that the size of the action space remains constant across different instances, improving solution efficiency and enhancing interpretability. Based on 113 simple dispatching rules proposed by Panwalkar *et al.* [45], we design factory dispatching rules, job dispatching rules, machine dispatching rules and AGV dispatching rules, as shown in Table 5. At each scheduling point, the agent selects the appropriate factory, job, machine and AGV based on these combination dispatching rules. The action at a scheduling point is defined as $A = \{A_f, A_j, A_m, A_a\}$, where A_f is the action space of the factory, which represents the set of selectable factory dispatching rules, A_j is the action space of the job, which represents the set of selectable job dispatching rules, A_m is the action space of the machine, which represents the set of selectable machine dispatching rules, A_a is the action space of AGV, which represents the set of selectable AGV dispatching rules.

In DRL, to obtain the optimal strategy, the greedy-based selection strategy is typically selected [46], where the action that maximizes the action value is chosen as the agent's next action, known as utilization. However, blindly selecting the action with the maximum action value can cause the agent to fall into a local optimum. To address this, random action selection is sometimes employed to explore other potential states. This is implemented through the ε-greedy exploration strategy in basic DQN, as shown in Eq. (27), where r is a randomly generated number in the range of [0, 1], and ε is the exploration rate.

$$a = \begin{cases} \underset{a' \in A}{\text{argmax}} Q(a') & r > \varepsilon \\ \text{randomselection} & r \le \varepsilon \end{cases}, r \in [0,1] \tag{27}$$

However, the static exploration rate in basic DQN may result in inadequate exploration during the early stages and excessive randomness in the later stages. To better balance exploration and utilization, this paper introduces a new dynamic exploration rate ε to improve basic DQN, as shown in Eq. (28), where e is the current training episode, and E is the total number of training episodes. In the early stages of training, the ε value is large and the model relies more on random exploration (i.e., selecting random actions), which allows for more comprehensive exploration of the environment and helps avoid premature convergence to a local optimum. As training progresses, ε value gradually decreases and the model becomes more dependent on the current strategy (i.e., selecting the optimal action predicted by the model), enabling more effective decision-making based on the knowledge already learned.

$$\varepsilon = \begin{cases} 1 - \dfrac{1.1e}{E} & \varepsilon \ge 0.01 \\ 0.01 & \varepsilon < 0.01 \end{cases} \tag{28}$$

4.3. Definition of reward function

The reward function has a direct impact on the learning strategy and convergence ability of the algorithm. Therefore, the reward function should be closely related to the objective function of the scheduling problem and can guide the agent to make effective decisions. Aiming at makespan of the scheduling target in the DHFJSP-AGV, the corresponding reward function is designed as Eq. (29), where r_t is the reward function value at the time t, $CTJ_{\max}$ is the maximum completion time of

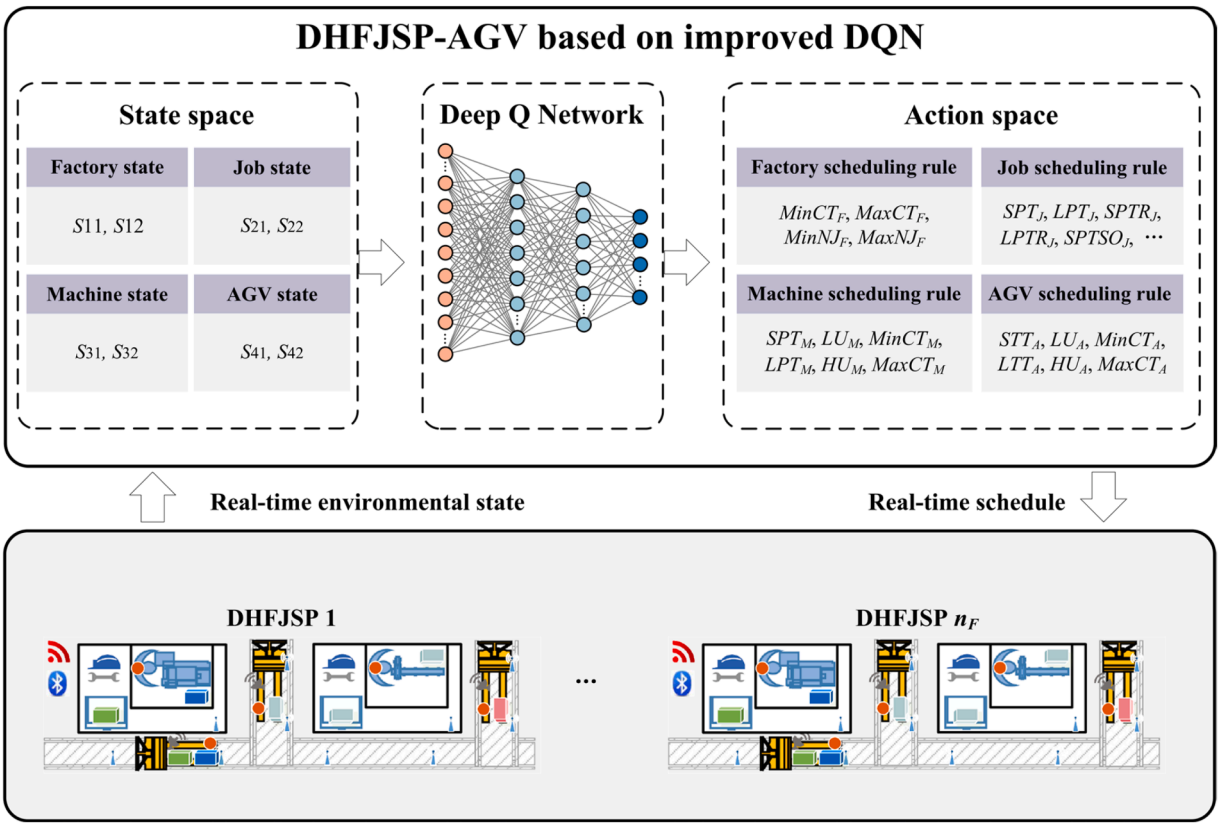


Fig. 3. Scheduling framework of DHFJSP-AGV based on improved DQN.

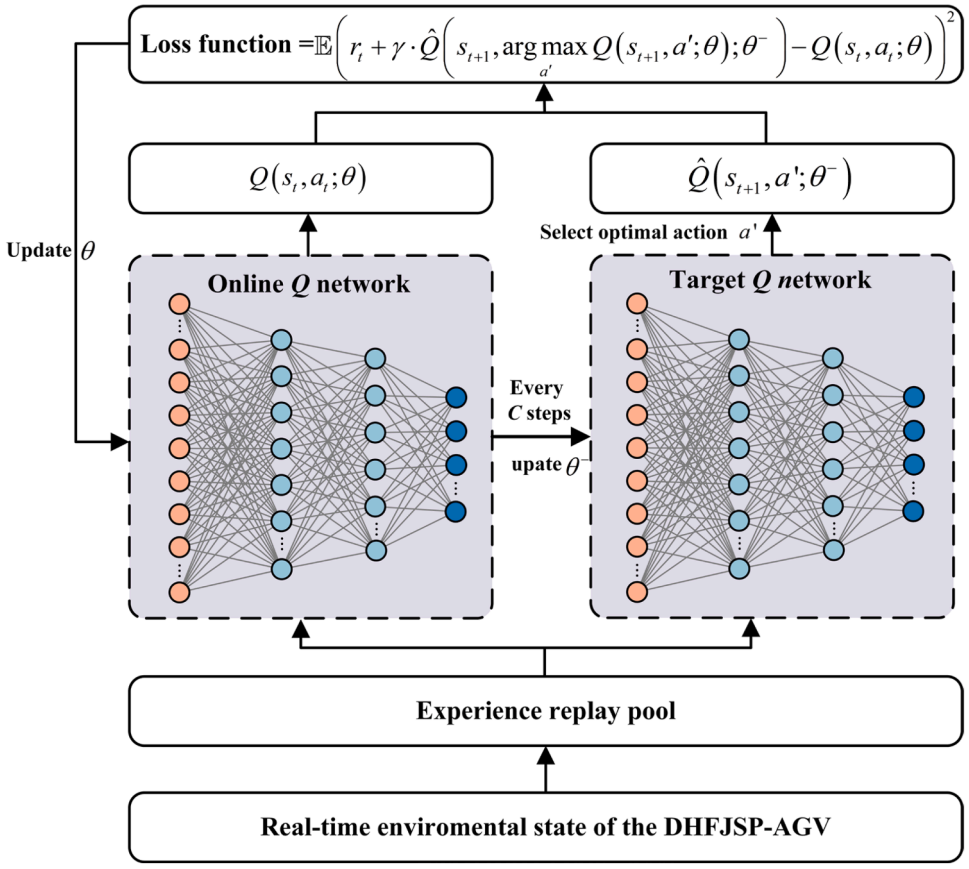


Fig. 4. Structure of the improved DQN.

Algorithm 1
The improved DQN-based training method

Input:Episode E, batch size N, target Q network parameter update frequency C, learning rate α, Initial greedy strategy ε
Output:Online Q network parameter θ
Initialize online Q network with random weights θ
Initialize target Q network by copying the online Q network $\theta^{-} \leftarrow \theta$
Initialize the experience replay pool RB, batch size N
for episode = 1, 2, …, E **do**
  Initialize environment, initialize time step $t = 0$
while the scheduling is not completed **do**
    Obtain the current state features s_t
    According to Eq. (27) choose an action a_t
    Execute action a_t, obtain new state features s_{t+1}, reward r_t
    Append (s_t, a_t, r_t, s_{t+1}) to RB;
    if $len(RB) \geq N$ **then**
    Randomly sample N (s_t, a_t, r_t, s_{t+1}) from RB for training
      $y_t \leftarrow r_t + \gamma \widehat{Q}\left(s_{t+1}, \text{argmax}_{a'} Q(s_{t+1}, a'; \theta); \theta^{-}\right)$
  // Calculate the target Q-value
      $L(\theta) \leftarrow \mathbb{E}\left(y_t - Q(s_t, a_t; \theta)\right)^2$ // Minimize the loss function
      $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta L(\theta)$
        // Update online Q network weights θ using gradient descent
      if $t\%C = 0$ **then**
      $\theta^{-} \leftarrow \theta$ // Update target Q network weights θ^{-}
    end if
    $t \leftarrow t + 1$, and according to Eq. (28) update ε
  end if
end while
end for

the job at the time $t-1$, and $CTJ_i$is the maximum completion time of the job J_i.

$$r_t = CTJ_{\max} - \max_{i=1,2,\cdots,n_J} (CTJ_i) \tag{29}$$

4.4. Scheduling framework based on improved DQN

To address the DHFJSP-AGV, this paper proposes a scheduling framework based on improved DQN, as shown in Fig. 3. This framework captures the real-time status of factories, jobs, machines and AGVs by continuously gathering and aggregating the information from all distributed heterogeneous factories. The state features are then fed into the improved DQN, which evaluates these inputs and selects optimal combination dispatching rules from the action space as the agent's action. Based on the chosen combination dispatching rules, the agent identifies the specific factory, job, machine, and AGV, enabling real-time scheduling across different factories.

Basic DQN uses the same neural network to both select and evaluate actions, which often leads to the overestimation of Q-values due to the maximization bias. Specifically, when updating Q-values, the network may overestimate the value of an action by selecting the action with the highest Q-value using the same network that is also used to evaluate the action. This issue can destabilize training and hinder the model's learning process.

To alleviate this problem, we adopt double DQN as an improvement to the basic DQN framework. Double DQN decouples the action selection and action evaluation processes, reducing the overestimation of Q-values. In double DQN, two separate neural networks are employed:

(1) The online Q network is responsible for selecting actions based on the current state.
(2) The target Q network is used to evaluate the Q-value of the selected action, providing a more stable target for the Q-value update.

This approach is formally described in Eq. (30), where θ represents the parameters of online Q network and θ^{-} represents the parameters of target Q network. The target Q network is updated less frequently than the online Q network, which further helps to stabilize training by preventing rapid shifts in the Q-value targets.

$$y_t = r_t + \gamma \widehat{Q}\left(s_{t+1}, \underset{a'}{\text{argmax}} Q(s_{t+1}, a'; \theta); \theta^{-}\right) \tag{30}$$

The structure of the improved DQN, including the separate online and target networks, is illustrated in Fig. 4. This architecture enables the model to better handle the exploration-exploitation trade-off and refine its policy over time without being misled by overestimated action values.

4.5. Training method

The training method is based on the framework of improved DQN, which is provided in Algorithm 1. The improved dynamic exploration rate and double DQN are mentioned in Section 4.2 and Section 4.4.

5. Numerical experiments

In this section, we first design test instances and provide a detailed explanation of the experimental parameters and the training process of the improved DQN. To rigorously evaluate the effectiveness and generalization capability of the improved DQN across various DHFJSP-AGV production configurations, comprehensive numerical comparison experiments are conducted. Finally, a real-world case with dynamic transportation times is used to further validate the performance of the improved DQN.

5.1. Test instances

Since DHFJSP-AGV is not studied before, and no public dataset is available. Therefore, this study generates 42 test instances based on reference [16], with the dataset accessible at: https://github.com/Hinachanyasashii/DHFJSP_AGV_DQN.git. In the generated instances, the number of factories ranges from $n_F \in \{2,3,4,5,6,7,8\}$, and the number of jobs ranges from $n_J \in \{10,20,30,40,50,80,100,120,150,200,250,300\}$. Each factory contains either 5 or 10 machines, reflecting different factory sizes. Each job consists of five operations, and each operation can be processed by one or more compatible machines. The processing time for each operation falls within the range $pt_{ijfk} \in [5, 20]$, with varying times for the same operation across different factories. Jobs are transported from the starting location LU to different machines for processing, with

Table 6
The hyperparameter configurations of the algorithm.

Hyperparameter	Value
Discount factor γ	0.95
Learning rate lr	0.001
Initial greedy strategyε	0.99
The minimum value of greedy strategy	0.01
Number of target Q network update steps$C$	200
Batch size	64

the transportation time $tt_{kk'}$ generated by a random function. Since the machine layout is identical across factories, a shared transportation time matrix is used, as shown in Eq. (31).

$$TT = \begin{bmatrix} 0 & 6 & 8 & 6 & 8 & 10 & 12 & 10 & 12 & 8 \\ 8 & 0 & 2 & 8 & 2 & 4 & 6 & 4 & 6 & 8 \\ 6 & 10 & 0 & 10 & 8 & 2 & 4 & 6 & 4 & 10 \\ 12 & 4 & 6 & 0 & 6 & 8 & 10 & 8 & 10 & 6 \\ 10 & 2 & 4 & 6 & 0 & 6 & 8 & 2 & 8 & 2 \\ 8 & 8 & 2 & 8 & 6 & 0 & 6 & 4 & 2 & 8 \\ 6 & 10 & 8 & 10 & 8 & 6 & 0 & 6 & 4 & 4 \\ 12 & 4 & 6 & 4 & 2 & 8 & 10 & 0 & 10 & 4 \\ 10 & 6 & 4 & 6 & 4 & 2 & 8 & 2 & 0 & 10 \\ 2 & 12 & 4 & 10 & 12 & 4 & 10 & 2 & 4 & 0 \end{bmatrix} \tag{31}$$

5.2. The training process of improved DQN

This study utilizes Python programming within the PyCharm 2023.1 environment and leverages the PyTorch framework to implement the proposed algorithm. The code is executed on a personal computer equipped with a 12th Gen Intel (R) Core (TM) i9–12900KF and 64 GB of memory. The algorithm's performance is highly sensitive to parameter settings, so various experiments were conducted to fine-tune these values. The resulting hyperparameter configurations for the training process are listed in Table 6.

Fig. 5 presents the convergence curves for four instances (50J5F5M3A, 100J7F5M3A, 150J6F5M4A, and 200J7F5M5A) after 10,000 training episodes. The "50J5F5M3A" indicates that 50 jobs are assigned to 5 heterogeneous factories for processing, with 5 machines and 3 AGVs in each factory. Other instances follow a similar structure. The horizontal axis represents the number of iterations, while the vertical axis represents the cumulative reward value. For the convenience of observation, the Savitzky-Golay filter (S-G filter) is applied to smooth the reward value curves. Convergence is defined as the point where the reward value stabilizes and further performance improvements become steady. In our experiments, a fixed training period of 10,000 episodes was set to ensure sufficient exploration and learning. During the training process, we closely monitor convergence by tracking the cumulative reward and fluctuations.

At the early stages of training, the reward value fluctuates significantly, indicating that the agent is constantly exploring the

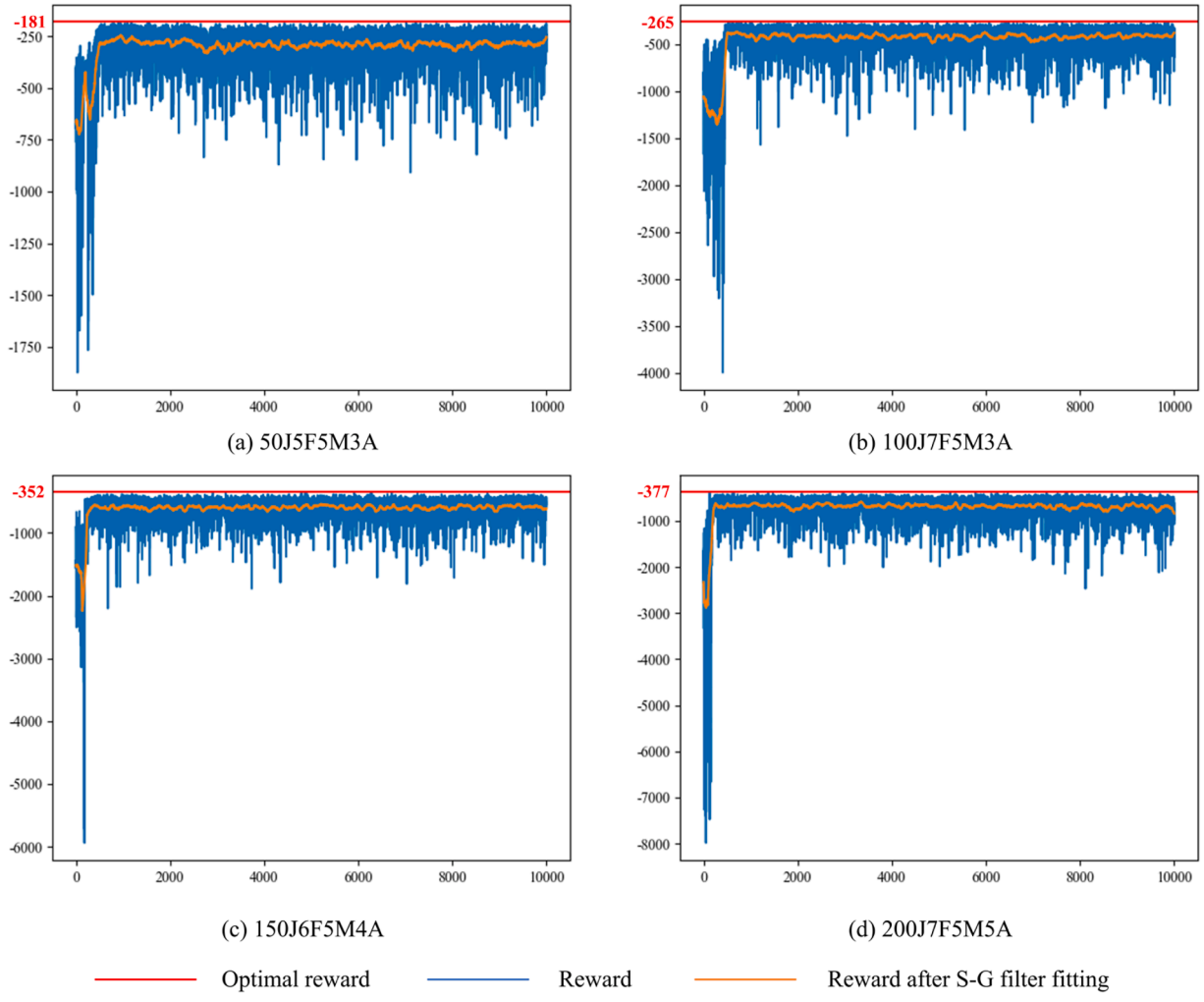


Fig. 5. Reward convergence curves of 50J5F5M3A, 100J7F5M3A, 150J6F5M4A and 200J7F5M5A.

Table 7
Mean value of makespan and the percentage difference by improved DQN and well-known combination dispatching rules.

Instance	CDR1	CDR2	CDR3	CDR4	CDR5	CDR6	Improved DQN
10J2F5M2A	208	288	310	182	145	147	**114**
	82.46 %	152.63 %	82.46 %	59.65 %	27.19 %	28.95 %	**0.00 %**
20J2F5M2A	291	341	432	321	252	276	**237**
	22.78 %	43.88 %	22.78 %	35.44 %	6.33 %	16.46 %	**0.00 %**
20J3F5M2A	256	210	342	270	184	170	**147**
	74.15 %	42.86 %	74.15 %	83.67 %	25.17 %	15.65 %	**0.00 %**
30J2F5M2A	498	461	542	449	384	353	**335**
	48.66 %	37.61 %	48.66 %	34.03 %	14.63 %	5.37 %	**0.00 %**
30J3F5M2A	348	342	399	319	280	267	**237**
	46.84 %	44.30 %	46.84 %	34.60 %	18.14 %	12.66 %	**0.00 %**
40J2F5M3A	491	410	538	461	337	333	**308**
	59.42 %	33.12 %	59.42 %	49.68 %	9.42 %	8.12 %	**0.00 %**
40J3F5M2A	393	393	471	458	322	353	**312**
	25.96 %	25.96 %	25.96 %	46.79 %	3.21 %	13.14 %	**0.00 %**
40J4F5M2A	364	338	404	363	239	251	**237**
	53.59 %	42.62 %	53.59 %	53.16 %	0.84 %	5.91 %	**0.00 %**
50J 3F5M3A	479	409	428	381	299	292	**270**
	77.41 %	51.48 %	77.41 %	41.11 %	10.74 %	8.15 %	**0.00 %**
50J4F5M3A	412	301	390	340	242	245	**211**
	95.26 %	42.65 %	95.26 %	61.14 %	14.69 %	16.11 %	**0.00 %**
50J5F5M3A	313	263	416	287	197	202	**181**
	72.93 %	45.30 %	72.93 %	58.56 %	8.84 %	11.60 %	**0.00 %**
80J4F5M3A	537	494	481	458	399	372	**341**
	57.47 %	44.86 %	41.06 %	34.31 %	17.00 %	9.09 %	**0.00 %**
80J5F5M3A	447	426	375	392	314	331	**283**
	57.95 %	50.53 %	32.50 %	38.51 %	10.95 %	16.96 %	**0.00 %**
80J6F5M3A	398	423	368	319	293	274	**241**
	65.16 %	75.52 %	52.70 %	32.37 %	21.58 %	13.69 %	**0.00 %**
100J4F5M3A	525	528	542	551	438	433	**414**
	26.81 %	27.54 %	26.81 %	33.09 %	5.80 %	4.59 %	**0.00 %**
100J5F5M3A	493	447	518	442	342	356	**329**
	49.85 %	35.87 %	49.85 %	34.35 %	3.95 %	8.21 %	**0.00 %**
100J6F5M3A	382	372	515	414	391	303	**274**
	39.42 %	35.77 %	39.42 %	51.09 %	42.70 %	10.58 %	**0.00 %**
100J7F5M3A	414	396	430	370	255	267	**249**
	66.27 %	59.04 %	66.27 %	48.59 %	2.41 %	7.23 %	**0.00 %**
100J2F10M8A	611	572	527	536	452	447	**378**
	61.64 %	51.32 %	39.42 %	41.80 %	19.58 %	18.25 %	**0.00 %**
100J3F10M6A	522	551	474	443	362	382	**334**
	56.29 %	64.97 %	41.92 %	32.63 %	8.38 %	14.37 %	**0.00 %**
120J5F5M4A	548	474	495	464	397	410	**353**
	55.24 %	34.28 %	40.22 %	31.44 %	12.46 %	15.81 %	**0.00 %**
120J6F5M4A	476	398	432	411	347	361	**302**
	57.61 %	31.78 %	43.04 %	36.09 %	14.90 %	19.53 %	**0.00 %**
120J7F5M4A	455	436	347	315	274	291	**252**
	80.56 %	73.02 %	37.70 %	25.00 %	8.73 %	15.48 %	**0.00 %**
120J2F10M8A	695	658	601	567	522	519	**457**
	52.08 %	43.98 %	31.51 %	24.07 %	14.22 %	13.57 %	**0.00 %**
120J3F10M6A	702	648	588	532	483	464	**410**
	71.22 %	58.05 %	43.41 %	29.76 %	17.80 %	13.17 %	**0.00 %**
150J5F5M4A	562	492	649	531	432	426	**397**
	41.56 %	23.93 %	41.56 %	33.75 %	8.82 %	7.30 %	**0.00 %**
150J6F5M4A	535	452	503	499	368	381	**352**
	51.99 %	28.41 %	51.99 %	41.76 %	4.55 %	8.24 %	**0.00 %**
150J7F5M4A	457	383	495	469	328	346	**316**
	44.62 %	21.20 %	44.62 %	48.42 %	3.80 %	9.49 %	**0.00 %**
150J3F10M8A	698	662	591	535	479	447	**395**
	76.71 %	67.59 %	49.62 %	35.44 %	21.27 %	13.16 %	**0.00 %**
150J4F10M6A	632	572	548	493	457	449	**384**
	64.58 %	48.96 %	42.71 %	28.39 %	19.01 %	16.93 %	**0.00 %**
200J6F5M5A	584	576	621	534	452	410	**377**
	54.91 %	52.79 %	54.91 %	41.64 %	19.89 %	8.75 %	**0.00 %**
200J7F5M5A	523	452	594	462	397	382	**331**
	58.01 %	36.56 %	58.01 %	39.58 %	19.94 %	15.41 %	**0.00 %**
200J3F10M8A	783	731	745	644	613	594	**512**
	52.93 %	42.77 %	45.51 %	25.78 %	19.73 %	16.02 %	**0.00 %**
200J4F10M7A	703	685	646	563	488	520	**435**
	61.61 %	57.47 %	48.51 %	29.43 %	12.18 %	19.54 %	**0.00 %**
250J6F5M5A	841	856	778	694	611	597	**510**
	64.90 %	67.84 %	52.55 %	36.08 %	19.80 %	17.06 %	**0.00 %**
250J8F5M5A	758	729	597	657	481	512	**436**
	73.85 %	67.20 %	36.93 %	50.69 %	10.32 %	17.43 %	**0.00 %**
250J4F10M6A	897	847	785	699	632	618	**546**
	64.29 %	55.13 %	43.77 %	28.02 %	15.75 %	13.19 %	**0.00 %**

(*continued on next page*)

Table 7 (*continued*)

Instance	CDR1	CDR2	CDR3	CDR4	CDR5	CDR6	Improved DQN
250J6F10M6A	681	657	609	593	622	506	**415**
	64.10 %	58.31 %	46.75 %	42.89 %	49.88 %	21.93 %	**0.00 %**
300J6F5M5A	1289	1047	1154	961	839	791	**687**
	87.63 %	52.40 %	67.98 %	39.88 %	22.13 %	15.14 %	**0.00 %**
300J8F5M5A	1112	1089	925	837	790	703	**595**
	86.89 %	83.03 %	55.46 %	40.67 %	32.77 %	18.15 %	**0.00 %**
300J4F10M8A	832	747	671	682	551	579	**431**
	93.03 %	73.32 %	55.68 %	58.24 %	27.84 %	34.34 %	**0.00 %**
300J6F10M8A	1045	996	903	866	710	735	**556**
	87.95 %	79.14 %	62.41 %	55.76 %	27.70 %	32.19 %	**0.00 %**

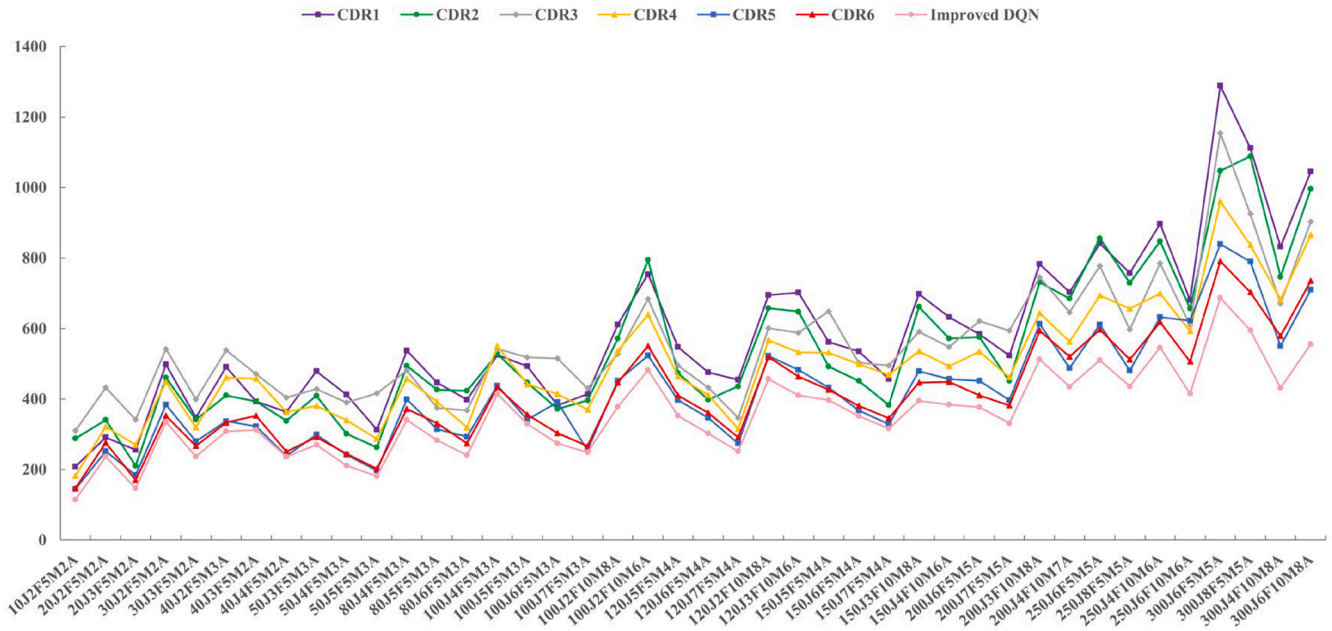


Fig. 6. Comparison of improved DQN and combination dispatching rules.

environment. As the number of training episodes increases to about 2000 times, the reward curves of most instances gradually flatten out and begins to converge, signalling that the agent has acquired enough empirical knowledge to approach the optimal solution for each decision. However, oscillations in the reward value persist after convergence due to the ongoing exploration, as the exploration rate is not set to zero. These oscillations, though present, do not detract from the agent's ability to select near-optimal solutions, as it continues to apply the most suitable dispatching rules at different scheduling points.

The results show that the algorithm converges relatively quickly, achieving the optimal solution multiple times for all instances. The optimal makespan values for the four instances are 181, 265, 352, and 377, respectively, indicating that the agent has effectively learned to select the appropriate dispatching rules to minimize makespan across various scheduling scenarios.

The consistent convergence observed across instances of different scales underscores the robustness and reliability of the improved DQN model, demonstrating its effectiveness in solving the scheduling problem even as the problem size increases.

5.3. Comparisons with the well-known combination dispatching rules

In order to verify the effectiveness of the improved DQN, the makespan is selected as the evaluation index to compare with the well-known combination dispatching rules. These rules were carefully designed to ensure their representativeness and relevance, drawing on classic principles frequently discussed in the scheduling literature [46, 47]. The selected combination dispatching rules are as follows:

1) CDR1: $MinNJ_F + SPT_J + MinCT_M + MinCT_A$, Select the factory with the least allocated jobs, the job with the shortest processing time, the machine with the minimum completion time, and the AGV with the minimum completion time.
2) CDR2: $MinNJ_F + SPT_J + SPT_M + MinCT_A$, Select the factory with the least allocated jobs, the job with the shortest processing time, the machine with the shortest processing time, and the AGV with the minimum completion time.
3) CDR3: $MinNJ_F + SPTR_J + MinCT_M + MinCT_A$, Select the factory with the allocated jobs, the job with the shortest remaining processing time, the machine with the minimum completion time, and the AGV with the minimum completion time.
4) CDR4: $MinNJ_F + SPTR_J + SPT_M + MinCT_A$, Select the factory with the least allocated jobs, the job with the shortest remaining processing time, the machine with the shortest processing time, and the AGV with the minimum completion time.
5) CDR5: $MinNJ_F + MRUO_J + MinCT_M + MinCT_A$, Select the factory with the least allocated jobs, the job with the most remaining unprocessed operations, the machine with the shortest completion time, and the AGV with the minimum completion time.
6) CDR6: $MinNJ_F + MRUO_J + SPT_M + MinCT_A$, Select the factory with the least allocated jobs, the job with the most remaining unprocessed operations, the machine with the shortest processing time, and the AGV with the minimum completion time.

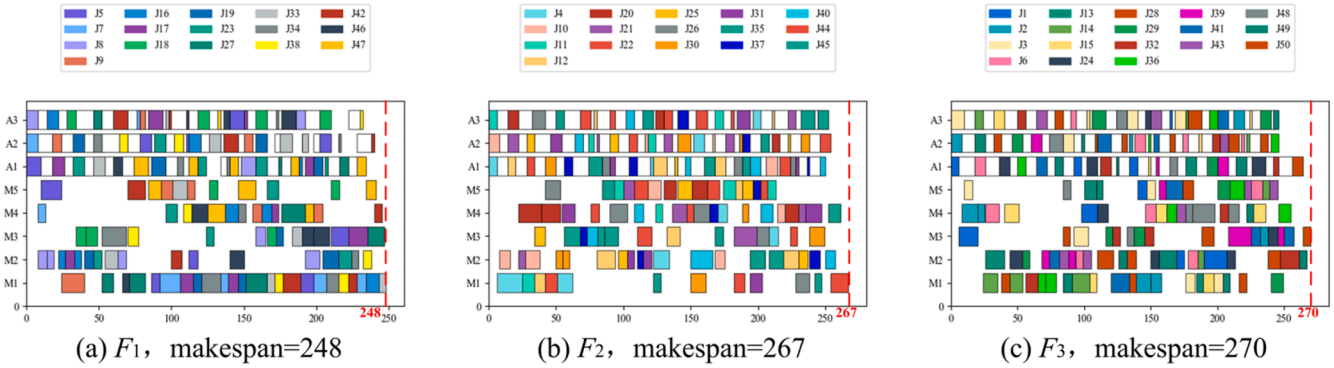


Fig. 7. Scheduling Gantt chart of the 50J 3F5M3A.

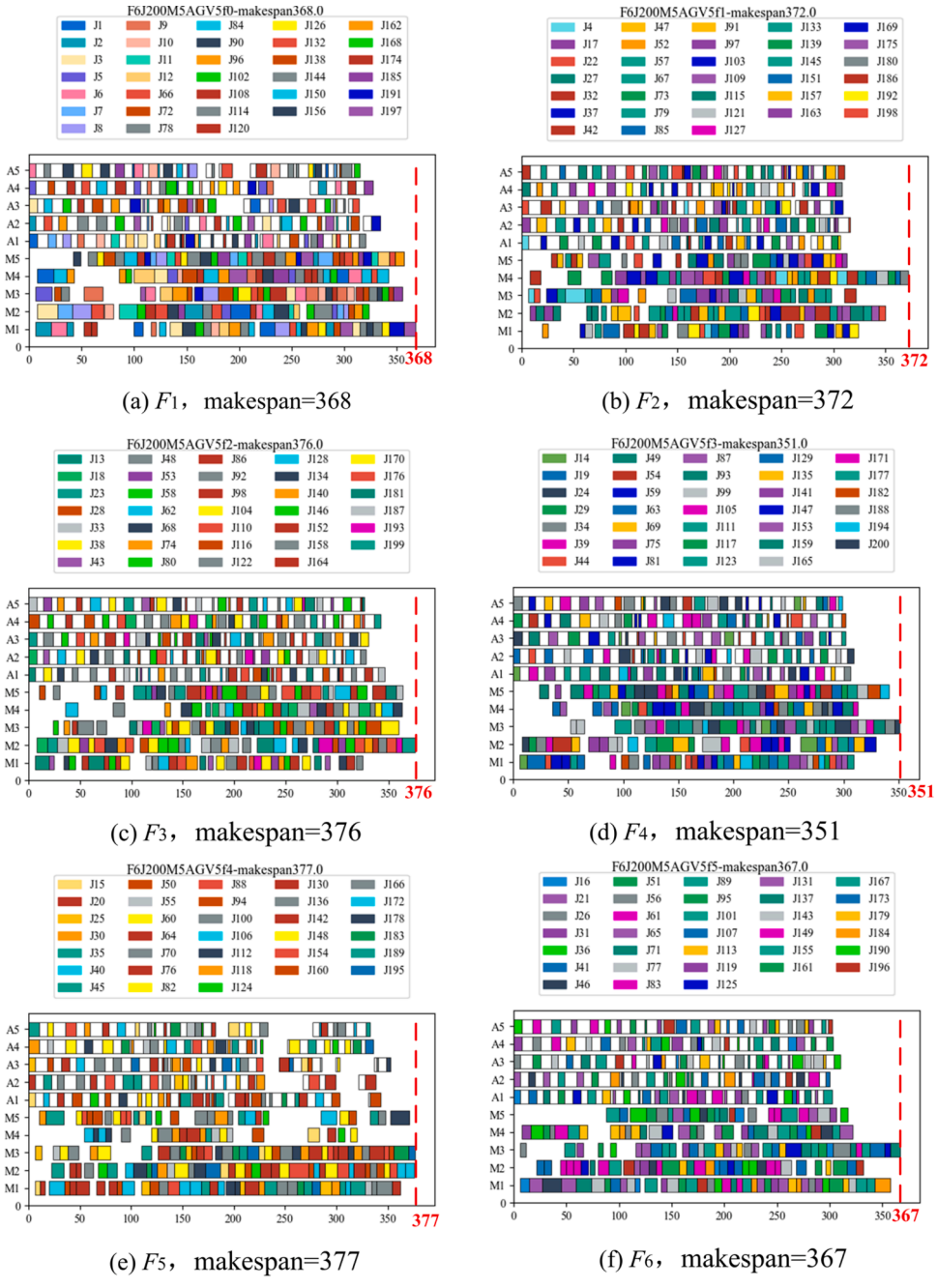


Fig. 8. Scheduling Gantt chart of the 200J6F5M5A.

Table 8
Mean value of makespan and training time by basic DQN, double DQN and improved DQN.

Instance	Basic DQN		Double DQN		Improved DQN	
	Objective/Gap	Time (s)	Objective/Gap	Time (s)	Objective/Gap	Time (s)
10J2F5M2A	137/20.18 %	0.07	119*/4.39 %	0.07	**114/0.00 %**	0.08
20J2F5M2A	258/13.16 %	0.13	242/6.14 %	0.14	**228/0.00 %**	0.17
20J3F5M2A	192/30.61 %	0.14	151*/2.72 %	0.15	**147/0.00 %**	0.18
30J2F5M2A	376/13.25 %	0.18	376/13.25 %	0.20	**332/0.00 %**	0.26
30J3F5M2A	276/18.97 %	0.19	244*/5.17 %	0.22	**232/0.00 %**	0.27
40J2F5M3A	358/14.74 %	0.31	333/6.73 %	0.33	**312/0.00 %**	0.37
40J3F5M2A	391/25.32 %	0.31	339/8.65 %	0.34	**312/0.00 %**	0.38
40J4F5M2A	300/27.12 %	0.32	247*/4.66 %	0.34	**236/0.00 %**	0.38
50J 3F5M3A	399/47.78 %	0.40	291/7.78 %	0.43	**270/0.00 %**	0.48
50J4F5M3A	304/42.06 %	0.40	260/21.50 %	0.45	**214/0.00 %**	0.49
50J5F5M3A	262/53.21 %	0.39	204/19.30 %	0.44	**171/0.00 %**	0.50
80J4F5M3A	470/37.83 %	0.52	395/15.84 %	0.55	**341/0.00 %**	0.61
80J5F5M3A	388/37.10 %	0.52	351/24.03 %	0.55	**283/0.00 %**	0.61
80J6F5M3A	355/47.30 %	0.51	302/25.31 %	0.54	**241/0.00 %**	0.60
100J4F5M3A	670/61.45 %	0.71	553/33.25 %	0.83	**415/0.00 %**	1.09
100J5F5M3A	587/73.15 %	0.74	487/43.66 %	0.85	**339/0.00 %**	1.08
100J6F5M3A	427/50.35 %	0.76	396/39.44 %	0.88	**284/0.00 %**	1.10
100J7F5M3A	339/33.46 %	0.94	303/19.29 %	1.12	**254/0.00 %**	1.15
100J2F10M8A	588/55.56 %	1.10	512/35.45 %	1.15	**378/0.00 %**	1.23
100J3F10M6A	509/52.40 %	1.11	470/40.72 %	1.15	**334/0.00 %**	1.26
120J5F5M4A	525/48.73 %	1.13	472/33.71 %	1.25	**353/0.00 %**	1.47
120J6F5M4A	442/46.36 %	1.13	385/27.48 %	1.26	**302/0.00 %**	1.47
120J7F5M4A	423/67.86 %	1.12	339/34.52 %	1.25	**252/0.00 %**	1.49
120J2F10M8A	690/50.98 %	1.22	623/36.32 %	1.43	**457/0.00 %**	1.79
120J3F10M6A	648/58.05 %	1.29	591/44.15 %	1.50	**410/0.00 %**	1.83
150J5F5M4A	623/56.93 %	1.25	564/42.07 %	1.47	**397/0.00 %**	1.88
150J6F5M4A	527/54.09 %	1.35	487/42.40 %	1.53	**342/0.00 %**	1.91
150J7F5M4A	501/63.19 %	1.64	441/43.65 %	1.79	**307/0.00 %**	1.96
150J3F10M8A	627/53.13 %	1.98	571/44.56 %	2.11	**395/0.00 %**	2.28
150J4F10M6A	588/51.44 %	2.04	554/44.27 %	2.18	**384/0.00 %**	2.39
200J6F5M5A	634/65.53 %	2.36	580/51.44 %	2.44	**383/0.00 %**	2.73
200J7F5M5A	603/72.28 %	2.40	574/64.00 %	2.60	**350/0.00 %**	2.86
200J3F10M8A	887/73.24 %	3.13	796/55.47 %	3.31	**512/0.00 %**	3.56
200J4F10M7A	722/65.98 %	3.21	651/49.66 %	3.37	**435/0.00 %**	3.60
250J6F5M5A	832/63.14 %	3.73	753/47.65 %	3.75	**510/0.00 %**	3.84
250J8F5M5A	801/83.72 %	3.70	711/63.07 %	3.75	**436/0.00 %**	3.82
250J4F10M6A	975/78.57 %	3.96	798/46.15 %	4.09	**546/0.00 %**	4.22
250J6F10M6A	771/85.78 %	3.92	665/60.24 %	4.04	**415/0.00 %**	4.13
300J6F5M5A	1259/83.26 %	6.27	1102/60.41 %	6.50	**687/0.00 %**	6.98
300J8F5M5A	1084/82.35 %	5.84	1043/75.29 %	6.23	**595/0.00 %**	6.61
300J4F10M8A	836/93.96 %	6.82	725/68.21 %	7.01	**431/0.00 %**	7.28
300J6F10M8A	987/77.52 %	6.53	844/62.59 %	6.79	**556/0.00 %**	7.04

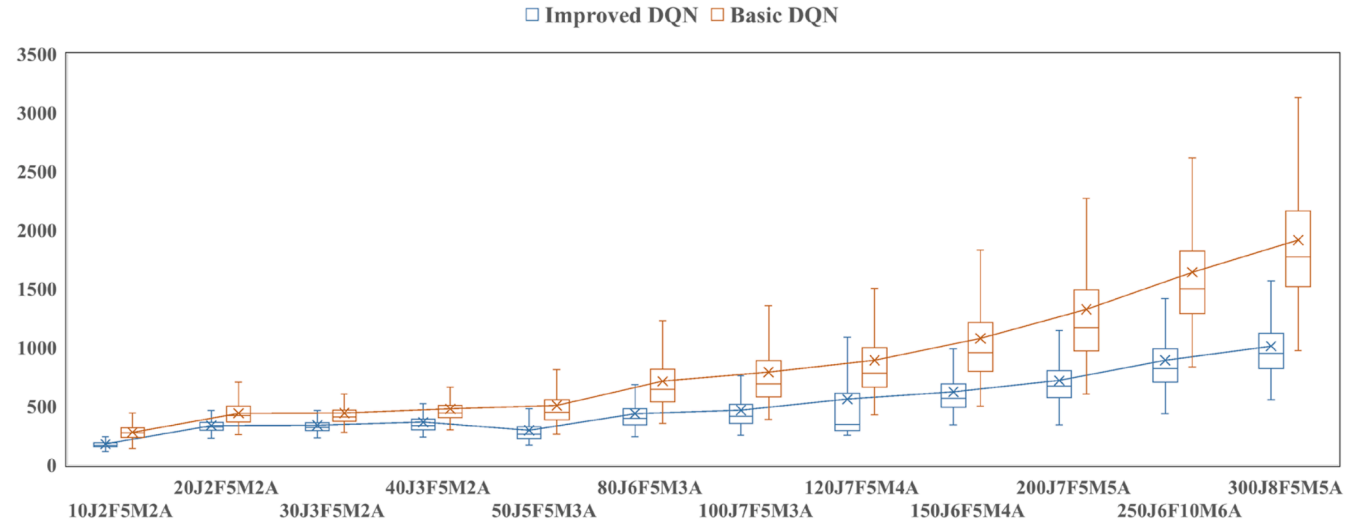


Fig. 9. Box plots of makespan for different instances.

These rules are based on widely used dispatching heuristics, such as shortest processing time (SPT), shortest remaining processing time (SRPT), and first in first out (FIFO), which have been proven effective in classical scheduling problems. For fairness, we ensured that the comparison environment and problem constraints were consistent across all experiments.

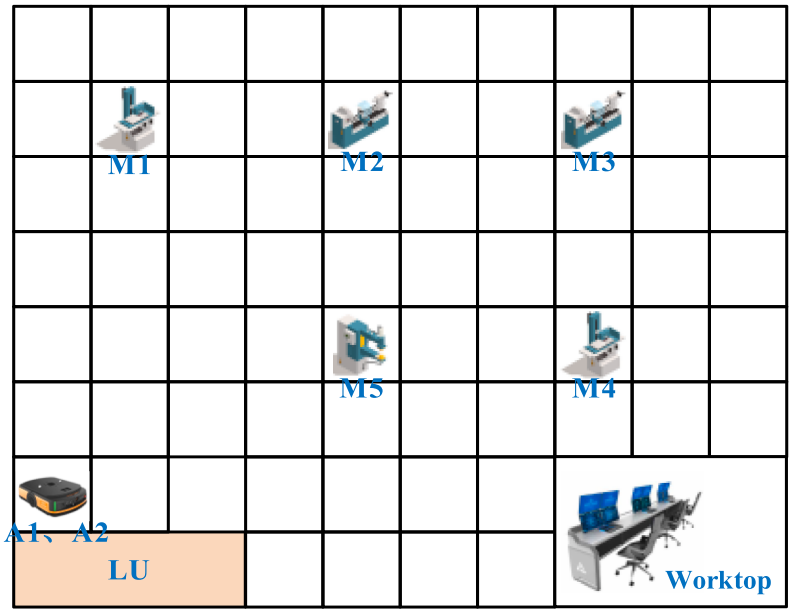


Fig. 10. Grid map of the workshop.

Table 9
The coordinate position of each device.

Device	Coordinate	Device	Coordinate
A1	(0, 2)	M3	(14, 12)
A2	(0, 2)	M4	(14, 6)
M1	(2, 12)	M5	(8, 6)
M2	(8, 12)		

For the 42 test instances, we conducted 20 independent simulation experiments and took the average of the results. The comparison of the makespan and gap from the optimal scheduling results, derived from applying the improved DQN and the well-known combination dispatching rules (with the best results highlighted in bold) is presented in Table 7. The performance of different combination dispatching rules varies significantly across different instance sizes. The maximum gaps for CDR1 and CDR2 from the optimal results exceeded 83 %. In contrast, the maximum gaps for CDR6 remained within 34.132 %, ranking just below the improved DQN. However, their performance shows considerable fluctuations and lacks stability across various scales. Overall, the optimal solutions obtained using the improved DQN consistently outperform those derived from the combination dispatching rules, demonstrating the superiority of the proposed improved DQN.

In order to more intuitively show the scheduling performance comparison between the improved DQN and the famous combination dispatching rules, the results are plotted as a line chart as shown in Fig. 6. It can be intuitively seen that the scheduling results of the improved DQN (the pink line in the figure) on 42 instances are better than the other 6 different combination dispatching rules. In addition, different combination dispatching rules show different performance when dealing with different instances, which proves that no single dispatching rule or single combination dispatching rule can adapt to all scheduling scenarios. The optimal scheduling Gantt chart of the two instances of 50J 3F5M3A and 200J6F5M5A is shown in Figs. 7 and 8, respectively. It can be seen that the makespan of the two instances is 270 and 377, respectively. The AGV use efficiency of all factories is relatively high, and basically all are in the transportation state. Especially in Fig. 8, the processing time of jobs on the same machine does not overlap, the number of jobs allocated by different factories is basically the same, and the maximum completion time of each factory is not much different, indicating that the trained agent is more reasonable for scheduling tasks. It is concluded that the algorithm proposed in this paper has high effectiveness in scheduling results.

5.4. Comparisons with the basic DQN and double DQN

To further verify the effectiveness of the two improvements, double DQN and improved ε-greedy exploration rate, we conducted a comparative analysis with both the basic DQN and double DQN. While double DQN addresses the overestimation bias in value estimation, it still faces the same exploration-exploitation challenges as the basic DQN. Specifically, a low exploration rate often results in insufficient exploration during the early stages, causing premature convergence to suboptimal solutions. Conversely, a high exploration rate can lead to excessive randomness during the later stages, preventing the model from fully leveraging learned strategies, thereby reducing both training efficiency and final performance. To ensure a fair and convincing comparison, we fixed the exploration rate of both the basic DQN and double DQN at 0.6. The performance of the basic DQN, double DQN and improved DQN were tested across multiple instances, each with 20 independent experiments. Table 8 presents the average makespan and training time for three methods. The superscript "*" denotes that among the 20 independent experiments in this instance, some of the results outperform the average of the solution by improved DQN.

The results indicate that the basic DQN and Double DQN perform well in small-scale test instances, with their results closely matching those of the improved DQN. Especially for Double DQN, in 20 independent experiments on the 10J2F5M2A, 20J3F5M2A and 30J3F5M2A, some of the results even outperform the average of the 20 independent runs of the improved DQN. This can be attributed to the small state space of the test cases, where the static exploration rate is sufficient for comprehensive exploration. However, as the problem size increases, the performance gap between the basic DQN, double DQN and the improved DQN becomes more significant. For example, in the 10J2F5M2A instance, the improved DQN reduces the makespan by about 20.18 % compared to basic DQN and by 4.39 % compared to double DQN. In the larger 300J4F10M8A instance, the reductions are approximately 93.96 % and 68.21 %, respectively.

While the solution results of double DQN are very close to those of the improved DQN in small-scale instances due to its value function correction, it still struggles with balancing exploration and exploitation.

Table 10
The processing information of the distributed manufacturing enterprise.

Job	Operation	F_1		F_2	
		Compatible machine	Processing time	Compatible machine	Processing time
J_1	O_{11}	1,2,3,4,5	13,10,8,10,11	1,2,3,4,5	6,10,9,7,15
	O_{12}	1,2,3	8,18,12	1,2,3	5,6,17
	O_{13}	1,2,3,5	12,13,16,19	1,2,3,5	15,10,14,14
	O_{14}	1,3,4,5	10,18,14,12	1,3,4,5	7,12,9,12
J_2	O_{21}	1,2,5	18,8,7	1,2,5	11,6,8
	O_{22}	2,3,4,5	16,17,14,19	2,3,4,5	8,18,12,14
	O_{23}	1,2,3,5	18,15,12,17	1,2,3,5	17,7,16,10
J_3	O_{31}	1,3,4	6,5,9	1,3,4	6,17,10
	O_{32}	1,4	6,19	1,4	15,9
	O_{33}	1,2,3,4,5	19,13,13,19,7	1,2,3,4,5	11,17,11,5,10
	O_{34}	1,2,3,4	12,11,15,17	1,2,3,4	10,7,14,10
J_4	O_{41}	1,3	7,8	1,3	6,15
	O_{42}	3,4,5	5,9,6	3,4,5	12,10,6
	O_{43}	1,3,4,5	17,6,12,8	1,3,4,5	13,17,19,13
	O_{44}	1,2,4,5	8,12,15,17	1,2,4,5	11,13,17,14
J_5	O_{51}	1,2,3,4,5	8,13,16,10,19	1,2,3,4,5	17,6,9,5,19
	O_{52}	1,4	19,11	1,4	12,9
	O_{53}	1,2,3,4,5	13,15,17,6,8	1,2,3,4,5	15,19,16,5,9
J_6	O_{61}	1,2,3,4,5	17,17,12,14,9	1,2,3,4,5	15,7,9,9,12
	O_{62}	1,2,4,5	14,13,15,18	1,2,4,5	14,18,14,13
	O_{63}	4,5	18,12	4,5	14,8
J_7	O_{71}	1,2,3,5	7,9,16,19	1,2,3,5	10,10,8,6
	O_{72}	1,2,4,5	11,16,15,14	1,2,4,5	9,12,6,19
	O_{73}	1,2,3,4	13,13,18,11	1,2,3,4	19,5,17,7
	O_{74}	1,2,3,4	7,11,14,18	1,2,3,4	10,12,5,14
J_8	O_{81}	1,2,4,5	11,19,16,5	1,2,4,5	9,7,8,18
	O_{82}	1,5	7,6	1,5	19,15
	O_{83}	3,5	16,9	3,5	19,10
J_9	O_{91}	1,3,4,5	10,13,5,14	1,3,4,5	16,8,19,19
	O_{92}	1,2,3,4,5	15,7,8,7,10	1,2,3,4,5	19,16,8,16,8
	O_{93}	1,4	13,9	1,4	13,9
J_{10}	O_{101}	4,5	19,16	4,5	6,18
	O_{102}	1,2,4,5	15,11,17,11	1,2,4,5	9,8,13,13
	O_{103}	1,2,3,4,5	12,9,15,15,14	1,2,3,4,5	19,10,18,18,19
J_{11}	O_{111}	1,2,3,4,5	12,12,6,15,19	1,2,3,4,5	12,15,10,15,13
	O_{112}	1,2,3,4,5	9,19,5,6,6	1,2,3,4,5	9,5,7,16,10
	O_{113}	1,2,3,4,5	7,18,8,7,19	1,2,3,4,5	9,7,10,10,18
	O_{114}	1,3,5	15,14,8	1,3,5	16,19,8
J_{12}	O_{121}	1,2,4,5	15,7,17,15	1,2,4,5	11,13,19,6
	O_{122}	1,2,3,4,5	17,10,6,17,13	1,2,3,4,5	10,15,16,14,11
	O_{123}	2,3,5	17,18,16	2,3,5	19,11,17
	O_{124}	3,4	12,16	3,4	10,6
J_{13}	O_{131}	1,2,3,4,5	12,16,15,12,9	1,2,3,4,5	18,19,8,18,16
	O_{132}	1,2,3,4,5	5,17,16,18,13	1,2,3,4,5	5,10,11,13,9
	O_{133}	1,2,3,4,5	18,10,6,18,16	1,2,3,4,5	5,16,16,7,10
	O_{134}	1,2,3,5	13,8,14,7	1,2,3,5	17,8,8,6
J_{14}	O_{141}	2,4	8,7	2,4	17,13
	O_{142}	1,2,3,4,5	19,15,5,13,11	1,2,3,4,5	14,10,8,6,17
	O_{143}	2,3,4,5	13,6,11,16	2,3,4,5	15,15,15,18
J_{15}	O_{151}	1,2,3,4,5	16,13,14,12,11	1,2,3,4,5	19,15,5,9,10
	O_{152}	2,4,5	19,13,18	2,4,5	5,13,5
	O_{153}	2,3	6,13	2,3	6,11
	O_{154}	1,2,3	18,9,9	1,2,3	11,12,11
J_{16}	O_{161}	1,3,4,5	10,8,10,18	1,3,4,5	12,19,13,17
	O_{162}	4,5	13,6	4,5	16,17
	O_{163}	2,4	14,7	2,4	15,10
	O_{164}	3,5	14,7	3,5	8,7
J_{17}	O_{171}	1,2,4,5	16,11,11,18	1,2,4,5	14,8,17,17
	O_{172}	1,2,3,4	9,8,12,13	1,2,3,4	5,15,12,15
	O_{173}	1,2,4,5	11,11,12,14	1,2,4,5	9,18,11,15
	O_{174}	2,3,4	5,12,15	2,3,4	10,12,9
J_{18}	O_{181}	1,2,3,4,5	16,14,16,17,18	1,2,3,4,5	10,18,19,16,7
	O_{182}	1,3,5	9,13,14	1,3,5	10,7,13
	O_{183}	1,2,3,4,5	19,6,16,7,12	1,2,3,4,5	13,8,8,13,13
J_{19}	O_{191}	1,2,3,4,5	11,5,14,11,18	1,2,3,4,5	17,15,5,8,9
	O_{192}	1,2	14,6	1,2	15,14
	O_{193}	4,5	14,5	4,5	17,17
	O_{194}	2,3,5	11,17,9	2,3,5	15,12,6
J_{20}	O_{201}	1,2,3,4	16,5,9,17	1,2,3,4	16,10,15,13
	O_{202}	1,2,3,4,5	6,9,14,5,8	1,2,3,4,5	10,14,11,14,16
	O_{203}	2,3,4,5	11,18,17,9	2,3,4,5	19,7,17,16
	O_{204}	2,5	12,8	2,5	18,11

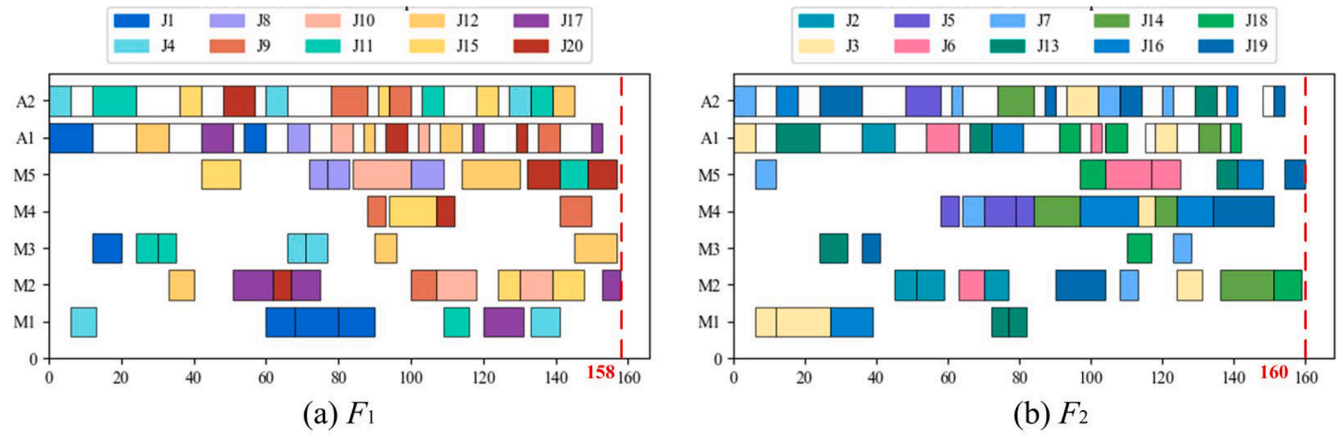


Fig. 11. Scheduling Gantt chart of the distributed manufacturing enterprise.

As the state space expands in large-scale examples, double DQN's performance falls significantly behind that of the improved DQN. This highlights the more crucial role of the improved ε-greedy exploration rate in handling larger-scale problems.

In terms of computational efficiency, the improved DQN exhibits slightly longer training times of each episode compared to the basic DQN. This increase is attributable to the separation of action selection from Q-value estimation and the higher exploration rate during early training stages, which elevates computational demands. Nevertheless, the additional time cost remains well within acceptable bounds, given the substantial improvements in solution quality.

Fig. 9 presents the boxplots of maximum completion time for both the basic DQN and the improved DQN across different problem instances, based on 10,000 solutions generated after 10,000 episodes. Outliers have been excluded for clarity. The results clearly demonstrate that the improved DQN consistently outperforms the basic DQN in terms of both average performance and solution stability. Across all instances, the improved DQN exhibits a significantly lower distribution range and mean value compared to the basic DQN. This indicates that the improved DQN not only achieves better solutions but also generates results with greater reliability. As the problem scale increases, the basic DQN struggles to effectively explore the larger state space, leading to a wider distribution of solutions and poor convergence behavior. In contrast, the improved DQN maintains a much narrower solution range and achieves superior results.

In summary, the improved DQN demonstrates greater robustness and competitiveness, especially in medium- and large-scale problems. Its ability to deliver superior solutions makes it a more suitable choice for tackling complex optimization problems like DHFJSP-AGV.

5.5. Example verification

In the previous section, AGV transportation times were assumed to follow a shared time matrix, demonstrating the algorithm's effectiveness under controlled conditions. However, in a real production workshop, AGV transportation times fluctuate based on dynamic, real-time conditions, making it impractical to rely solely on predefined timetables. Therefore, this section introduces a path planning algorithm that dynamically calculates AGV transportation times based on real-world factors [48]. These real-time inputs are then incorporated into the improved DQN for decision-making, further enhancing the model's practical applicability.

The test data come from a manufacturing enterprise with two heterogeneous factories that share identical equipment layouts but differ in production capacities. A grid-based method, shown in Fig. 10, is used to map the workshop layout. The overall workshop area is 20 m × 16 m, with each grid cell representing a 2 m × 2 m space. The specific coordinates of the device are provided in Table 9. Initially, both AGVs start at nearly the same position, coordinates (2, 0), due to their close initial distances.

Order information from the distributed manufacturing enterprise have been collected, with the processing details shown in Table 10. This table presents the processing times for 20 different jobs across 5 machines in two factories. Each factory is equipped with 2 AGVs responsible for transferring jobs to various machines. Compatible machines refer to a set of machines that can perform a specific operation.

The model trained in Section 5.2 is applied to a distributed manufacturing enterprise case, with real-time transportation times calculated through a path planning algorithm. To ensure high relevance to practical applications, we selected the model trained on the 20J2F2A instance, which has a high degree of similarity to the actual case, for verification purposes. However, the state space of the designed DHFJSP-AGV model is generally applicable and not dependent on the specific instance size, making any model trained under similar conditions to be effectively adapted to other distributed manufacturing situations. The scheduling process utilizing the trained model begins with inputting general state features that encompass factories, jobs, machines, and AGVs. Based on the current state, the model employs a greedy policy to select the optimal action, determining which factory, job, machine, and AGV will be involved in the processing. The processing time for each job is derived from job-machine processing data, while the transportation time for the AGVs is calculated in real time using the path planning algorithm. After executing the chosen action, the state space is updated to reflect the completed job and the movement of the AGVs. This iterative process continues until all jobs are processed. The scheduling results, presented in Fig. 11, show that the two factories achieved completion times of 158 and 160, with each factory evenly managing 10 jobs. Throughout this process, the AGVs remain actively engaged, with minimal idle time for the machines, demonstrating the algorithm's generalization capabilities in the new distributed manufacturing environment.

6. Conclusion

This paper proposes an improved deep Q network (DQN) for solving the distributed heterogeneous flexible job-shop scheduling problem considering automated guided vehicle transportation (DHFJSP-AGV). A linear integer programming model for DHFJSP-AGV is developed with the objective of makespan and formulated as a Markov decision process. Eight elaborately-designed state features normalized in the range of [0, 1] are utilized to represent the production state at each scheduling point. The factory, job, machine and AGV dispatching rules are selected from the set of pre-designed dispatching rules to form combination dispatching rules to represent the action space. DQN enhanced by the

improved ε-greedy exploration and double DQN is trained to select the appropriate combination dispatching rules. The improved DQN and six well-known combination dispatching rules are applied to randomly generated 42 instances to compare with six well-known combination dispatching rules, basic DQN and double DQN, which proves the effectiveness of the improved DQN. Finally, the improved DQN is combined with the path planning algorithm and applied to an actual distributed manufacturing enterprise, which verifies the superiority and generalization ability of the algorithm in the actual production scenario.

This study still has room for improvement. While rule-based scheduling methods offer fast computation and strong generalization, their scheduling quality has yet to reach optimal levels. Future work could explore more advanced state representation techniques to enhance decision-making accuracy. As the problem scale increases, the expanded state-action space may demand higher computational resources. Approaches such as distributed training, hierarchical modeling, and attention mechanisms could be employed to address large-scale environments more efficiently. Incorporating multi-objective optimization in future research could further enhance the practical applicability of the proposed method.

CRediT authorship contribution statement

Minghai Yuan: Writing – original draft. **Songwei Lu:** Writing – review & editing. **Liang Zheng:** Validation. **Qi Yu:** Visualization. **Fengque Pei:** Investigation. **Wenbin Gu:** Supervision.

Declaration of competing interest

All authors disclosed no relevant relationships.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Jiangsu Province of China (BK20241780), Changzhou Science and Technology Program Project (CM20223014 and CJ20220207) and Changzhou Science and Technology Support Plan (Social Development) Project (CE20205045).

Data availability

Data will be made available on request.

References

[1] N.B. Rad, J. Behnamian, Recent trends in distributed production network scheduling problem, Artif. Intell. Rev. 55 (4) (2022) 2945–2995, https://doi.org/10.1007/s10462-021-10081-5.
[2] S. Nosratabadi, R. Hooshmand, E. Gholipour, A comprehensive review on microgrid and virtual power plant concepts employed for distributed energy resources scheduling in power systems, Renew. Sust. Energ. Rev. 67 (2017) 341–363, https://doi.org/10.1016/j.rser.2016.09.025.
[3] Y. Li, X. Li, L. Gao, L. Meng, An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times, Comput. Ind. Eng. 147 (2020) 106638, https://doi.org/10.1016/j.cie.2020.106638.
[4] R. Wang, G. Wang, J. Sun, F. Deng, J. Chen, Flexible job shop scheduling via dual attention network-based reinforcement learning, IEEE Trans. Neural Netw. Learning Syst. 35 (3) (2024) 3091–3102, https://doi.org/10.1109/TNNLS.2023.3306421.
[5] Y. Yao, Q. Liu, L. Fu, X. Li, Y. Yu, L. Gao, W. Zhou, A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles, IEEE Transact. Automat. Sci. Eng. (2024) 1–14, https://doi.org/10.1109/TASE.2024.3356255.
[6] L. De Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, Eur. J. Oper. Res. 200 (2) (2010) 395–408, https://doi.org/10.1016/j.ejor.2009.01.008.
[7] P. Lu, M. Wu, H. Tan, Y. Peng, C. Chen, A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems, J. Intell. Manuf. 29 (1) (2018) 19–34, https://doi.org/10.1007/s10845-015-1083-z.
[8] Q. Luo, Q. Deng, G. Gong, X. Guo, X. Liu, A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm, Expert Syst. Appl. 239 (2024) 120161, https://doi.org/10.1016/j.eswa.2023.120161.
[9] J. Xie, X. Li, L. Gao, L. Gui, A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems, J. Manuf. Syst. 71 (2023) 82–94, https://doi.org/10.1016/j.jmsy.2023.09.002.
[10] L. Meng, C. Zhang, Y. Ren, B. Zhang, C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, Comput. Ind. Eng. 142 (2020) 106347, https://doi.org/10.1016/j.cie.2020.106347.
[11] X. Chen, Y. Du, Hybrid artificial immune algorithm for energy-efficient distributed flexible job shop in semiconductor manufacturing, Clust. Comput. 27 (3) (2024) 3075–3098, https://doi.org/10.1007/s10586-023-04127-2.
[12] L. Meng, Y. Ren, B. Zhang, J. Li, H. Sang, C. Zhang, MILP modeling and optimization of energy- efficient distributed flexible job shop scheduling problem, IEEE Access 8 (2020) 191191–191203, https://doi.org/10.1109/ACCESS.2020.3032548.
[13] X. Li, J. Xie, Q. Ma, L. Gao, P. Li, Improved gray wolf optimizer for distributed flexible job shop scheduling problem, Sci. China-Technol. Sci. 65 (9) (2022) 2105–2115, https://doi.org/10.1007/s11431-022-2096-6.
[14] K. Zhu, G. Gong, N. Peng, L. Zhang, D. Huang, Q. Liang, X. Li, Dynamic distributed flexible job-shop scheduling problem considering operation inspection, Expert Syst. Appl. 224 (2023) 119840, https://doi.org/10.1016/j.eswa.2023.119840.
[15] Q. Luo, Q. Deng, G. Gong, L. Zhang, W. Han, K. Li, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, Expert Syst. Appl. 160 (2020) 113721, https://doi.org/10.1016/j.eswa.2020.113721.
[16] R. Li, W. Gong, L. Wang, C. Lu, C. Dong, Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling, IEEE Trans. Syst. Man Cybern. -Syst. 54 (1) (2024) 201–211, https://doi.org/10.1109/TSMC.2023.3305541.
[17] Q. Zhang, W. Shao, Z. Shao, D. Pi, J. Gao, Deep reinforcement learning driven trajectory-based meta-heuristic for distributed heterogeneous flexible job shop scheduling problem, Swarm Evol. Comput. 91 (2024) 101753, https://doi.org/10.1016/j.swevo.2024.101753.
[18] Y. Chen, X. Liao, G. Chen, Y. Hou, Dynamic intelligent scheduling in low-carbon heterogeneous distributed flexible job shops with job insertions and transfers, Sensors 24 (7) (2024) 2251, https://doi.org/10.3390/s24072251.
[19] H. Zhang, Y. Chen, Y. Zhang, G. Xu, Energy-saving distributed flexible job shop scheduling optimization with dual resource constraints based on integrated Q-learning multi-objective Grey Wolf optimizer, CMES-Comp. Model. Eng. Sci. 140 (2) (2024) 1459–1483, https://doi.org/10.32604/cmes.2024.049756.
[20] J. Sun, Z. Xu, Z. Yan, L. Liu, Y. Zhang, An approach to integrated scheduling of flexible job-shop considering conflict-free routing problems, Sensors 23 (9) (2023) 4526, https://doi.org/10.3390/s23094526.
[21] A. Goli, E.B. Tirkolaee, N.S. Aydin, Fuzzy integrated cell formation and production scheduling considering automated guided vehicles and Human factors, IEEE Trans. Fuzzy Syst. 29 (12) (2021) 3686–3695, https://doi.org/10.1109/TFUZZ.2021.3053838.
[22] W. Tan, X. Yuan, G. Huang, Z. Liu, Low-carbon joint scheduling in flexible open-shop environment with constrained automatic guided vehicle by multi-objective particle swarm optimization, Appl. Soft. Comput. 111 (2021) 107695, https://doi.org/10.1016/j.asoc.2021.107695.
[23] X. Wen, Y. Fu, W. Yang, H. Wang, Y. Zhang, C. Sun, An effective hybrid algorithm for joint scheduling of machines and AGVs in flexible job shop, Meas. Control 56 (9–10) (2023) 1582–1598, https://doi.org/10.1177/00202940231173750.
[24] X. Han, W. Cheng, L. Meng, B. Zhang, K. Gao, C. Zhang, P. Duan, A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV, Swarm Evol. Comput. 86 (2024) 101538, https://doi.org/10.1016/j.swevo.2024.101538.
[25] L. Meng, W. Cheng, B. Zhang, W. Zou, W. Fang, P. Duan, An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem, Sensors 23 (8) (2023) 3815, https://doi.org/10.3390/s23083815.
[26] B. Xin, S. Lu, Q. Wang, F. Deng, X. Shi, J. Cheng, Y. Kang, Simultaneous scheduling of processing machines and automated guided vehicles via a multi-view modeling-based hybrid algorithm, IEEE Trans. Autom. Sci. Eng. (2023), https://doi.org/10.1109/TASE.2023.3301656.
[27] X. Zhou, F. Wang, N. Shen, W. Zheng, A green flexible job-shop scheduling model for multiple AGVs considering carbon footprint, System.-Basel 11 (8) (2023) 427, https://doi.org/10.3390/systems11080427.
[28] C. Lei, N. Zhao, S. Ye, X. Wu, Memetic algorithm for solving flexible flow-shop scheduling problems with dynamic transport waiting times, Comput. Ind. Eng. 139 (2020) 105984, https://doi.org/10.1016/j.cie.2019.07.041.
[29] Z. Liu, S. Guo, L. Wang, Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption, J. Clean Prod. 211 (2019) 765–786, https://doi.org/10.1016/j.jclepro.2018.11.231.
[30] B. Zhou, X. Liao, Particle filter and Levy flight-based decomposed multi-objective evolution hybridized particle swarm for flexible job shop greening scheduling with crane transportation, Appl. Soft. Comput. 91 (2020) 106217, https://doi.org/10.1016/j.asoc.2020.106217.
[31] Y. Du, J. Li, C. Luo, L. Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, Swarm Evol. Comput. 62 (2021) 100861, https://doi.org/10.1016/j.swevo.2021.100861.

[32] Y. Du, J. Li, C. Li, P. Duan, A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times, IEEE Trans. Neural Netw. Learn. Syst. 35 (4) (2024).
[33] J. Li, Y. Du, K. Gao, P. Duan, D. Gong, Q. Pan, P. Suganthan, A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem, IEEE Trans. Autom. Sci. Eng. 19 (3) (2022) 2153–2170, https://doi.org/10.1109/TASE.2021.3062979.
[34] Y. An, X. Chen, J. Zhang, Y. Li, A hybrid multi-objective evolutionary algorithm to integrate optimization of the production scheduling and imperfect cutting tool maintenance considering total energy consumption, J. Clean Prod. 268 (2020) 121540, https://doi.org/10.1016/j.jclepro.2020.121540.
[35] Z. Zhou, L. Xu, X. Ling, B. Zhang, Digital-twin-based job shop multi-objective scheduling model and strategy, Int. J. Comput. Integrat. Manufact. 37 (1–2) (2024) 87–107, https://doi.org/10.1080/0951192X.2023.2204475.
[36] Y. Li, H. Minh, M. Cao, X. Qian, M.A. Wahab, An integrated surrogate model-driven and improved termite life cycle optimizer for damage identification in dams, Mech. Syst. Signal Process. 208 (2024) 110986, https://doi.org/10.1016/j.ymssp.2023.110986.
[37] B. Dang, H. Nguyen-Xuan, M.A. Wahab, An effective approach for VARANS-VOF modelling interactions of wave and perforated breakwater using gradient boosting decision tree algorithm, Ocean Eng. 268 (2023) 113398, https://doi.org/10.1016/j.oceaneng.2022.113398.
[38] L. Nguyen-Ngoc, Q. Nguyen-Huu, G. De Roeck, T. Bui-Tien, M. Abdel-Wahab, Deep neural network and evolved optimization algorithm for damage assessment in a truss bridge, Mathematics 12 (15) (2024) 15, https://doi.org/10.3390/math12152300.
[39] W. Song, X. Chen, Q. Li, Z. Cao, Flexible job-shop scheduling via graph neural network and Deep reinforcement learning, IEEE Trans. Ind. Inf. 19 (2) (2023) 1600–1610, https://doi.org/10.1109/TII.2022.3189725.
[40] V. Tran, T. Nguyen, H. Nguyen-Xuan, M.A. Wahab, Vibration and buckling optimization of functionally graded porous microplates using BCMO-ANN algorithm, Thin-Wall. Struct. 182 (2023) 110267, https://doi.org/10.1016/j.tws.2022.110267.
[41] M. Zhang, L. Wang, F. Qiu, X. Liu, Dynamic scheduling for flexible job shop with insufficient transportation resources via graph neural network and deep reinforcement learning, Comput. Ind. Eng. 186 (2023) 109718, https://doi.org/10.1016/j.cie.2023.109718.
[42] F. Zhang, R. Li, W. Gong, Deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling with multi-AGV, Comput. Ind. Eng. 189 (2024) 109917, https://doi.org/10.1016/j.cie.2024.109917.
[43] H. Wang, J. Cheng, C. Liu, Y. Zhang, S. Hu, L. Chen, Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events, Appl. Soft. Comput. 131 (2022) 109717, https://doi.org/10.1016/j.asoc.2022.109717.
[44] R. Chen, W. Li, H. Yang, A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem, IEEE Trans. Ind. Inf. 19 (2) (2023) 1322–1331, https://doi.org/10.1109/TII.2022.3167380.
[45] S.S. Panwalkar, W. Iskander, A survey of scheduling rules, Oper. Res. 25 (1) (1977) 45–61, https://doi.org/10.1287/opre.25.1.45.
[46] S. Luo, Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning, Appl. Soft. Comput. 91 (2020) 106208, https://doi.org/10.1016/j.asoc.2020.106208.
[47] J.C. Serrano-Ruiz, J. Mula, R. Poler, Job shop smart manufacturing scheduling by deep reinforcement learning, J. Ind. Inform. Integrat. 38 (2024) 100582, https://doi.org/10.1016/j.jii.2024.100582.
[48] X. Yin, P. Cai, K. Zhao, Y. Zhang, Q. Zhou, D. Yao, Dynamic path planning of AGV based on kinematical constraint A* algorithm and following DWA fusion algorithms, Sensors 23 (8) (2023) 4102, https://doi.org/10.3390/s23084102.