Applied Soft Computing Journal 154 (2024) 111342

Contents lists available at ScienceDirect

Applied Soft Computing

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

An adaptive multi-objective multi-task scheduling method by hierarchical deep reinforcement learning

Jianxiong Zhang [a,b], Bing Guo [a,b], Xuefeng Ding [a,b], Dasha Hu [a,b], Jun Tang [a,b,c], Ke Du [c], Chao Tang [c], Yuming Jiang [a,b,*]

[a] College of Computer Science, Sichuan University, Chengdu, 610065, Sichuan, China
[b] Big Data Analysis and Fusion Application Technology Engineering Laboratory of Sichuan Province, Chengdu, 610065, Sichuan, China
[c] Changhong Central Research Institute, Sichuan Changhong Electronic (Group) Co., Ltd., Mianyang, 621000, Sichuan, China

ARTICLE INFO

Keywords:
Vaule net
Multi-task scheduling
Resource competition and conflict
Adaptive multi-objective scheduling
Hierarchical deep reinforcement learning

ABSTRACT

Actual manufacturing process scheduling in enterprise alliances are multi-task scheduling problems involving dynamic factors, and the competition and conflict for manufacturing resources also exist between multi-tasks. How to perform adaptive multi-objective scheduling of multi-tasks based on the real-time state of the manufacturing environment becomes critical. Therefore, this paper constructs an adaptive multi-task multi-objective scheduling considering resource competition and conflict among tasks (AMMS-RCCT) model based on the enterprise alliance value net, and adopts a hybrid strategy of "parallel+serial" to resolve conflicts while reducing the waiting time of tasks. With the objective of optimizing the total manufacturing time and total manufacturing cost, an adaptive multi-objective deep Q network (AMDQN) is proposed to solve the AMMS-RCCT model. AMDQN is based on a two-hierarchy deep reinforcement learning architecture containing a front controller deep Q network (C-DQN) and a back actuator deep Q network (A-DQN), which performs hierarchical decision-making on optimization objectives and scheduling rules to achieve compromise between multiple objectives while reducing the complexity for optimal selection scheduling rules. For the two optimization objectives of time and cost, two reward algorithms are proposed by introducing two metrics, the estimated tardiness rate and the estimated overspend rate, which guide the A-DQN to learn and adjust the scheduling rules according to the state changes. Besides, nine composite scheduling rules are designed to adapt to the dynamic manufacturing environment from multiple dimensions such as task urgency and completion rate as well as manufacturing resource utilization and cost. Finally, AMDQN is experimentally compared with the proposed nine composite scheduling rules, scheduling rules in existing research, and other scheduling methods based on reinforcement learning in simulated manufacturing environments with different numbers of tasks, subtasks, and manufacturing cells. The experimental results verify the effectiveness and superiority of AMDQN for multi-objective adaptive scheduling in multi-task scheduling problems.

Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.2019441.v1.

1. Introduction

In the actual manufacturing process, a dynamic alliance of manufacturing enterprises typically receives multiple orders for similar or heterogeneous manufacturing tasks within a certain time window [1,2]. The optimality of each individual task does not guarantee the overall optimality of the total order. In fact, most scheduling problems in low-volume, multi-variety discrete manufacturing systems can be viewed as multi-task optimal scheduling problems (multi-task problems that need to take dynamic factors into account). In multi-task dynamic optimal scheduling, the enterprise alliance is more concerned with maximizing the overall benefit, i.e., the global optimization of multi-tasks. Moreover, multi-tasks with similar requirements may compete and conflict for manufacturing resources. Therefore, with full consideration of the possible resource competition and conflict during multi-task scheduling, how to resolve the conflict and efficiently perform real-time

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: College of Computer Science, Sichuan University, Chengdu, 610065, Sichuan, China.
E-mail address: jiangym@scu.edu.cn (Y. Jiang).

https://doi.org/10.1016/j.asoc.2024.111342
Received 12 August 2023; Received in revised form 23 January 2024; Accepted 30 January 2024
Available online 3 February 2024

batch adaptive scheduling for multi-tasks becomes a key issue. Due to its high complexity and generality, the multi-task scheduling problem considering resource competition and conflict has been an important topic of research in both academia and industry [3].

Meta-heuristics and reinforcement learning (RL) algorithms are two techniques that are widely used for manufacturing scheduling. Meta-heuristic algorithms include genetic algorithms (GA) [4–7], ant colony optimization (ACO) algorithms [8,9], particle swarm optimization (PSO) algorithms [10–13] and so on. Li et al. [9] proposed an improved collaborative PSO algorithm with a bi-objective optimization model for batch task scheduling in cloud manufacturing. Yang et al. [14] proposed a GA-based scheduling system for supporting the analysis and planning of dynamic demand for multiple plants. Wang et al. [15] proposed an adaptive adjustment algorithm based on improved ACO algorithm for service composition exception handling for anomalous events in cloud manufacturing. Zhou et al. [16] proposed a hybrid teaching–learning-based optimization (HTLBO) algorithm combining ACO and cuckoo search algorithm for optimal allocation of resources in cloud manufacturing. The meta-heuristic algorithm decomposes the dynamic scheduling problem into a series of static scheduling problems and solves them separately according to the set objectives and is able to achieve better results. However, its iterative optimization process requires large computational overhead and suffers from weak generalization and poor adaptability. When the manufacturing environment conditions change, the scheduling plan needs to be re-solved. As a result, meta-heuristic algorithms are rarely used in multi-task real-time scheduling problems that consider dynamic factors.

In deep reinforcement learning (DRL) based algorithms, the scheduling problem is modeled as a markov decision process (MDP), where the decision maker sequentially adopts the correct actions (scheduling rule) to optimize a predefined goal based on the current production state. DRL algorithms learn from the environment and adapt their strategies based on the performance of agents in the environment, with better decision-making and adaptation capabilities. Thus, it is widely used to solve manufacturing scheduling problems. Dong et al. [17] proposed a task scheduling mechanism based on deep Q network (DQN) to train the neural network parameters by offline training for large-scale task scheduling. Chen et al. [18] combined the attention mechanism with DRL to model cloud–edge collaborative manufacturing task scheduling as a MDP to improve the efficiency of batch scheduling. For the problem of scheduling multiple combined tasks, Ping et al. [19] combined DRL with sequence generation for generating scheduling sequences for multiple tasks before scheduling. Park et al. [20] proposed a RL-based setup change scheduling method, which learns the appropriate policy by sharing parameters among agents to get the scheduling method with the shortest manufacturing time. Zhang et al. [21] constructed a multi-agent manufacturing system with online scheduling and scheduling strategy optimization capabilities for the interference problem of stochastic dynamic events.

Therefore, in this paper, the DRL algorithms with faster solving speed and stronger adaptive ability is used to solve the multi-task real-time scheduling problem considering dynamic factors, in order to overcome the slow response and weak adaptability of the traditional scheduling methods, such as accurate methods and meta-heuristic algorithms. However, for multi-objective scheduling problems, the single-agent structure usually combine all objectives for optimization. Since the optimization of different objectives corresponds to different reward functions and strategies, it is difficult for a single agent to achieve a good compromise among multiple optimization objectives, and thus it is difficult to design a single agent to optimize all objectives simultaneously. Hierarchical reinforcement learning (HRL) can address this problem by shifting the complexity of multi-objective optimization to collaboration between agents. HRL learns to operate on different levels of temporal abstraction, decomposing complex problems into multiple subproblems that are handled separately by agents at different levels. This enables more efficient responses to complex tasks and utilizes information transfer and collaboration between agents at different levels to improve learning efficiency.

With the motivation above, this paper constructs an adaptive multi-task multi-objective scheduling considering resource competition and conflict among tasks (AMMS-RCCT) model based on a hybrid strategy of “parallel+serial”. And an adaptive multi-objective deep Q network (AMDQN) based on a hierarchical deep reinforcement learning architecture is proposed to optimally solve AMMS-RCCT to achieve the overall optimization of total manufacturing time and total manufacturing cost. The contribution of this research are as follows.

(1) For the value net composed of multi-task parallel processing in enterprise alliance in a certain time window, this paper constructs the value net optimization model AMMS-RCCT from the perspective of overall benefit maximization. This model takes total manufacturing time and total manufacturing cost of multi-tasks as the optimization objectives, taking into account the resource utilization rate, and realizes multi-task real-time batch adaptive scheduling.
(2) To solve the resource competition and conflict among multi-tasks, an adaptive multi-objective deep Q network (AMDQN) is proposed by adopting a hybrid scheduling strategy of “parallel + serial”. AMDQN is based on a two-hierarchy deep reinforcement learning architecture containing a front controller DQN and a back actuator DQN, which reduces the complexity of scheduling rule selection while achieving multi-objective optimization compromise by decoupling the selection of optimization objectives and optimal scheduling rules.
(3) For the two optimization objectives of total manufacturing time and total manufacturing cost of the value net, two metrics, estimated tardiness rate and estimated overspend rate, as well as three levels of reward values are introduced into the reward algorithm. It guides the actuator DQN to learn the optimal scheduling rules for the objective selected by the controller DQN according to the real-time manufacturing state, realizing adaptive multi-objective scheduling under different manufacturing environments.

2. Literature review

Researchers have proposed various approaches to solve the manufacturing scheduling problem and this section briefly reviews the related work on manufacturing scheduling. It includes traditional manufacturing scheduling methods (accurate methods, game theory based methods and bio-inspired meta-heuristics), and RL-based manufacturing scheduling methods.

2.1. Traditional scheduling methods for manufacturing

Akbaripour et al. [22] proposed a mixed integer programming (MIP) model for solving service selection optimization and scheduling (SSOS) problems with complex combinatorial structures. For the manufacturing synchronization problem (MfgSync) in cloud manufacturing scheduling, Chen et al. [23] proposed an optimal solution algorithm based on dynamic programming (DP) and developed a cooperative game for customer coordination. Liu et al. [24] proposed a multi-task scheduling model based on Monte-Carlo (MC) methods and investigated the impact of task scheduling methods for different workloads on completion time and service utilization. Wang et al. [15] proposed an improved ACO algorithm to achieve adaptive adjustment of service composition exception in cloud manufacturing using processing quality, cost and service quality as optimization objectives. Wang et al. [25] proposed a multi-objective whale optimization algorithm for adaptive adjustment of service composition for multi-tasks. Liu et al. [26] proposed an architecture for platform-based smart manufacturing systems

(PSMSs) and based on this, proposed a PMS scheduling model for scheduling aggregated resources. Zhou et al. [27] proposed a dynamic scheduling method based on real-time simulation for solving the dynamic task scheduling problem in cloud manufacturing environments, but this method requires too much high-speed computing power and high-speed network transmission. Hu et al. [28] analyzed the factors affecting the scheduling of manufacturers, transformed the scheduling into a mathematical problem, and solved it using a chaotic optimization algorithm. Li et al. [29] designed multi-objective ACO (MACO) algorithm and multi-objective NSGA-II (MGA) algorithm to solve the scheduling problem, taking into account the dependency and independence of subtasks. Liu et al. [30] proposed a non-cooperative game model for 3D printing services and a multi-phase integrated solution based on GA. Laili et al. [31] used six multi-objective evolutionary algorithms to solve the multi-stage integrated scheduling problem for hybrid tasks containing different orders. Huang et al. [32] analyzed the relationship between key factors of QoS for different cloud services and proposed a chaotic control optimization algorithm (CCOA) for optimal selection of cloud service composition.

In summary, meta-heuristic algorithms are widely used to solve the manufacturing scheduling problem, which uses some rules to solve the scheduling problem with better results. However, its iterative search process is very time-consuming and prone to local optimization. Moreover, its solution process relies on long-term iterative optimization, lacks the ability to respond quickly to the dynamic environment, and suffers from insufficient adaptability.

In contrast, the proposed AMDQN uses a hierarchical deep reinforcement learning architecture to make hierarchical decisions on multi-objective optimization and optimal scheduling strategies, and designs different levels of agents to process them separately. This enables AMDQN to flexibly adjust its policies in dynamic environments, thus adapting to changes in the environment and realizing adaptive scheduling in different manufacturing environments. In addition, the proposed AMDQN is training-based and can save the trained models offline, thus obtaining fast response capability by reusing the pre-trained models.

2.2. RL-based scheduling methods

Due to its strong nonlinear adaptive and decision-making capabilities, RL shows obvious advantages in solving dynamic scheduling problems, and is widely applied to the fields of manufacturing service composition and shop job scheduling. Liang et al. [33] proposed a DQN algorithm with prioritized replay for solving cloud manufacturing service composition considering logistics. Liu et al. [34] investigated the DRL-based cloud manufacturing scheduling problem and proposed a DRL model for online single-task scheduling in cloud manufacturing. Zhou et al. [35] proposed a DRL-based solution method for the dynamic task scheduling problem with the optimization objective of minimizing the maximum completion time of all tasks. Zhu et al. [36] transformed a scheduling problem with multiple resources into a common learning objective and proposed a RL-based online learning approach to optimize it. To achieve intelligent decision-making for dynamic scheduling and reconfiguration, Yang et al. [37] proposed an actor-critic based DRL system architecture with the goal of minimizing the total delay of all jobs. Du et al. [38] constructed a collaborative optimization framework for robot service scheduling and proposed a DQN-based collaborative optimization method for service scheduling to achieve comprehensive performance improvement of the whole manufacturing system. Yin et al. [39] proposed a decentralized multi-task attention allocation (MTAA) framework in DRL, which is embedded with an A3C approach in order to solve the AGV task allocation and path planning problems. Dong et al. [40] proposed a workflow scheduling algorithm based on actor-critic architecture for task sequencing and task allocation with the goal of minimizing task execution time. For the large-scale scheduling problem in cloud manufacturing, Dong et al. [41] proposed a DQN-based task scheduling mechanism to achieve optimization of task execution time. Wei et al. [42] considered the service composition problem as a sequential decision-making problem and proposed a workflow application scheduling method based on Q-learning algorithms for optimal service decision-making. Liu et al. [43] proposed a scheduling algorithm based on DQN and Dueling DQN for the decentralized robotic manufacturing service scheduling problem, while optimizing both service quality and service performance objectives. Liu et al. [44] combined multi-agent reinforcement learning and graph convolutional networks to address the high real-time requirements of scheduling problems in cloud environments in order to solve dynamic scheduling problems. Tang et al. [45] proposed a DQN-based scheduling approach for the problem of scheduling policy decisions in reconfigurable manufacturing systems (RMS) to minimize reconfiguration behavior.

Due to the complexity and uncertainty of the manufacturing process, the state of the manufacturing tasks and resources is constantly changing. Therefore, it is critical to dynamically resolve resource competition and conflict among tasks according to the real-time state of the manufacturing environment to realize multi-task adaptive scheduling. DRL-based algorithms are able to learn from the environment with strong perception and have better decision-making capabilities by adjusting their strategies according to the performance of agents in the environment. In the dynamic multi-task scheduling problem, DRL-based algorithms can learn and adjust their behaviors according to changes in the state of the manufacturing environment in order to adapt to different manufacturing environments and task requirements. However, few research works have applied DRL to adaptive multi-task multi-objective scheduling with resource competition and conflict between tasks. Table 1 summarizes the differences between this paper and other mainstream RL-based manufacturing scheduling methods.

3. Problem definition and mathematical modeling

3.1. AMMS-RCCT description

Due to the continuous change of multi-tasks and their collaboration in the enterprise alliance, the multiple value chains driven by multi-tasks cross each other, and the structure of enterprise collaborative value activities changes from a simple linear relationship to a complex, dynamic net structure, forming a value net. The dynamic construction process of the value net is shown in Fig. 1. As can be seen in Fig. 1, multi-task may all require the same manufacturing cell when task matching is performed. However, since manufacturing cells can only handle one task at a time, this leads to competition and conflict between different tasks. Therefore, how to globally adaptively adjust and optimize the value net according to the real-time state of manufacturing resources, resolve the conflicts among tasks, and realize multi-task multi-objective adaptive scheduling is the focus of this paper.

The adaptive multi-task multi-objective scheduling with resource competition and conflict between tasks considered in this paper can be defined as follows. There are n sequentially arriving manufacturing tasks $Task = \{Task_1, Task_2, \ldots, Task_n\}$ in a certain time window that are simultaneously scheduled to be processed on m manufacturing cells $MC = \{MC_1, MC_2, \ldots, MC_m\}$. Each manufacturing task $Task_i$ contains n_i subtasks, where $ST_{i,j}$ is the ith subtask of $Task_i$. Each subtask $ST_{i,j}$ has a corresponding manufacturing cell set $CMC_{i,j}$ $(CMC_{i,j} \subseteq MC)$, which consists of manufacturing cells that can accomplish that subtask $ST_{i,j}$. There is some intersection between the set of manufacturing cells corresponding to the subtasks of different tasks, which may lead to competition and conflict between multiple tasks. Therefore, the target of the problem is to minimize the total manufacturing time and manufacturing cost of all tasks in the batch while resolving conflicts between multiple tasks. The notations used for problem expression are listed in Table 2.

Table 1
Existing RL based methods for manufacturing scheduling problem.

Work	State space	Algorithm	Objective	Problem
Wang et al. [46] (2016)	Discrete	Q-learning	Service quality	Cloud manufacturing
Liu et al. [34] (2019)	Continuous	Deep Q-learning	Service quality; Performance	Cloud manufacturing
Hofmann et al. [47] (2020)	Continuous	Deep Q-learning	Throughput time	Reconfigurable manufacturing
Liang et al. [33] (2021)	Discrete	Deep Q-learning	Time; Cost; Reliability	Cloud manufacturing
Tang and Salonitis [45] (2021)	Continuous	Deep Q-learning	Reconfiguration actions	Reconfigurable manufacturing
Luo et al. [48] (2021)	Continuous	Hierarchical deep Q-learning	Total weighted tardiness; Machine utilization rate	Job shops
Wang et al. [44] (2022)	Continuous	Deep Q-learning	Makespan	Cloud manufacturing
Zhang et al. [49] (2022)	Continuous	Deep Q-learning	Weighted tardiness	Cloud manufacturing
Wang et al. [50] (2022)	Continuous	Deep Q-learning	Time; Cost; Reliability	Cloud manufacturing
Wang et al. [51] (2022)	Continuous	Deep Q-learning	Makespan; Machine utilization rate; Job delay rate	Job shops
Ours	Continuous	Hierarchical deep Q-learning	Makespan; Cost	Cloud manufacturing

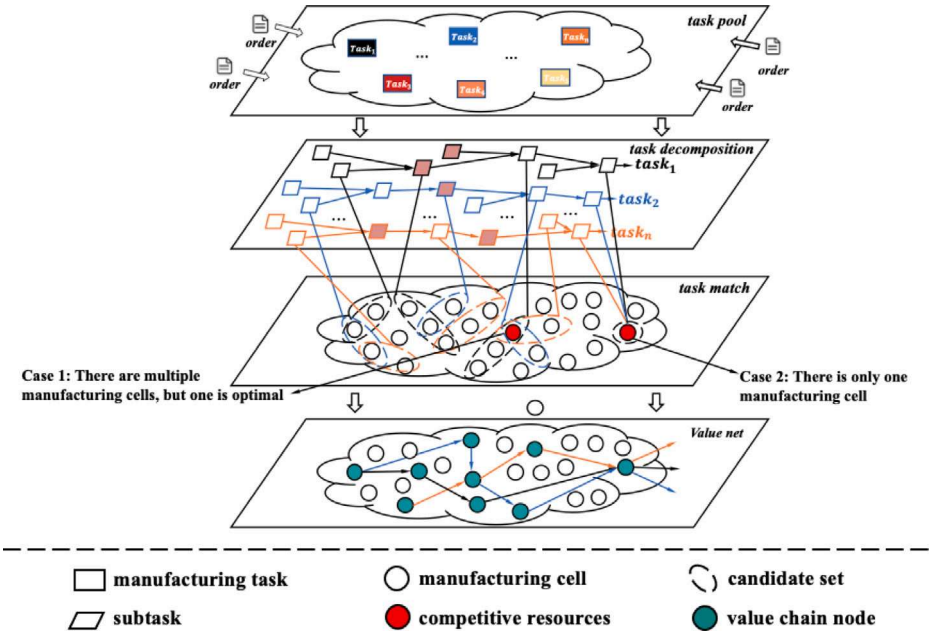


Fig. 1. The dynamic construction process of value nets.

3.2. Pre-conditions in AMMS-RCCT

Parallel execution. If a subtask does not have any dependent subtasks, it can be immediately assigned the corresponding manufacturing cell for processing. That is, most of the subtasks of each task are carried out in a parallel manner, rather than in a completely pipeline production manner.

$$\text{if } Pre(ST_{i,j}) = \emptyset, \text{ then } A_i \le (CT_{i,j} - t_{i,j,k})X_{i,j,k} \tag{1}$$

Subtask dependencies. If a subtask has dependent subtasks, the subtask should be executed until all dependent subtasks are completed. That is, the earliest start time of a subtask with multiple dependent subtasks depends on the latest completion time of all its dependent subtasks.

$$\text{if } Pre(ST_{i,j}) \neq \emptyset, \text{ then } (CT_{i,j} - t_{i,j,k})X_{i,j,k} \ge \max_{j \in Pre(ST_{i,j})} CT_{i,j} \tag{2}$$

Manufacturing cell task type. A manufacturing cell can only perform a certain type of subtask in the value chain. That is, a manufacturing cell will only appear once in a value chain.

$$CMC_{i,1} \cap CMC_{i,2} \cap \ldots \cap CMC_{i,j} = \emptyset \tag{3}$$

Table 2
Notation list.

Notation	Description
n	Number of manufacturing tasks
m	Number of manufacturing cells
$Task_i$	The ith task
n_i	Number of subtasks for manufacturing task
MC_k	The kth manufacturing cell
$ST_{i,j}$	The jth subtask of ith task
$CMC_{i,j}$	The set of manufacturing cells that can accomplishing the th subtask of $ST_{i,j}$
OC_k	Occupied time of manufacturing cell MC_k
$Pre_{i,j}$	The set of dependent subtasks of subtask $ST_{i,j}$
$t_{i,j,k}$	The manufacturing time of the subtask $ST_{i,j}$ on the manufacturing cell MC_k
$c_{i,j,k}$	The manufacturing cost of the subtask $ST_{i,j}$ on the manufacturing cell MC_k
A_i	The start processing time of the task $Task_i$
$CT_{i,j}$	The actual completion time of subtask $ST_{i,j}$
CTM_k	The completion time of the last subtask assigned to the manufacturing cell MC_k
$SN_i(t)$	The number of completed subtasks of the manufacturing task $Task_i$ at scheduling point t
D_i	The delivery deadline for the task $Task_i$
i, j, k	Index of manufacturing task $Task$, subtask ST and manufacturing cell MC
$X_{i,j,k}$	$X_{i,j,k}$ determines which manufacturing cell a subtask is assigned on. $X_{i,j,k}$ equals to 1 if subtask $ST_{i,j}$ is assigned to manufacturing cell MC_k, or 0 otherwise
$Y_{i,j,j+1}$	$Y_{i,j,j+1}$ determines the relative processing priority between any two subtasks. $Y_{i,j,j+1}$ equals to 1 if subtask $ST_{i,j}$ is a dependent subtask of $ST_{i,j+1}$, or 0 otherwise

3.3. Optimization objectives of AMMS-RCCT

According to the definition in 3.1 and the actual constraints considered in AMMS-RCCT, two optimization objectives are used in this research based on the actual manufacturing process. The first is to minimize the total manufacturing time of all tasks, and the second objective is to minimize the total manufacturing cost of all tasks.

$$\text{Minimize} \begin{cases} T = \max\limits_{i\in n} \max\limits_{j\in n_i} CT_{i,j} & \text{(a)} \\ C = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n_i} \sum\limits_{k=1}^{m} c_{i,j,k} X_{i,j,k} & \text{(b)} \end{cases} \tag{4}$$

$$\text{s.t.} \begin{cases} CT_{i,j} \le D_i & \text{(a)} \\ \sum\limits_{k\in MC_{i,j}} X_{i,j,k} = 1 & \text{(b)} \\ (STM_{i,j}, STM_{i,j} + t_{i,j,k}) \cap OC_k & \text{(c)} \end{cases} \tag{5}$$

Eq. (4)(a) is the total manufacturing time for all tasks, i.e., the maximum completion time. Considering the research scenario of this paper is a situation where most subtasks are parallel and a few tasks are serial. Therefore, the final completion time of each task depends on the completion time of its last subtask. Eq. (4)(b) is the total manufacturing cost for all tasks. Eq. (5)(a) is a delivery deadline constraint where each task has a delivery deadline that must be met. Eq. (5)(b) is the subtask assignment constraint, which ensures that each subtask is scheduled to be executed on the corresponding manufacturing cell and that a subtask can only be assigned to one manufacturing cell. Eq. (5)(c) is the occupancy time constraint. The existing scheduling plan will not be affected by the newly reached task, i.e., the execution time of the newly reached task cannot overlap with the occupancy time of the existing scheduling plan, where $STM_{i,j}$ is the start execution time of the subtask $ST_{i,j}$.

4. Proposed method for AMMS-RCCT

4.1. The hybrid optimization scheduling strategy

In order to solve the AMMS-RCCT, this paper proposes a hybrid optimal scheduling strategy of "parallel+serial", as shown in Fig. 2. Corresponding to the two different cases shown in Fig. 1 in 3.1, the optimization strategy can be divided into two cases: parallel execution and time-sharing (serial) execution.

(1) Parallel execution. For the case (Case1 in Fig. 1) where there are multiple manufacturing cells that can perform such subtasks, multiple subtasks are assigned to different manufacturing cells to be performed simultaneously, i.e., multiple tasks are performed in parallel. In this case, the problem is how to select the most suitable manufacturing cell for each subtask, which can make the two objectives of total manufacturing time and total manufacturing cost relatively optimal.

(2) Time-sharing (serial) execution: For the case (Case2 in Fig. 1) that there is only one manufacturing cell can execute this type of subtask, the time-sharing execution is realized by adjusting the execution time of each subtask. In this case, the problem is how to determine the order in which the tasks should be processed that makes the two objectives of total manufacturing time and total manufacturing cost relatively optimal.

From the above analysis, it can be concluded that AMMS-RCCT can actually be modeled as a MDP (shown in the dashed box in Fig. 2). It selects an appropriate manufacturing cell for each subtask and generates the corresponding scheduling plan until scheduling is completed for all subtasks.

The "parallel+serial" hybrid optimized strategy is essentially a fully parallel operation strategy. However, the actual environment cannot always be able to meet the conditions of fully parallel operation, only to take the "parallel + serial" strategy to solve the actual problem. Only in extreme special cases when none of the conditions for parallelism can be met does it degenerate to a fully serial approach.

4.2. Framework of the AMDQN

Due to the uncertainty and complexity of manufacturing systems, it is difficult to accurately model their state transfer functions. Therefore, this paper proposes an adaptive multi-objective deep Q network (AMDQN) to solve the AMMS-RCCT. As shown in Fig. 3, the proposed AMDQN consists of a two-hierarchy deep reinforcement learning architecture and two DQN (C-DQN and A-DQN). The two-hierarchy deep reinforcement learning architecture learns to operate at different levels of temporal abstraction, where C-DQN acts as a controller to learn strategies for different optimization objectives at a slower spatio-temporal scale and A-DQN acts as an actuator to learn the selection of different scheduling rules to optimize the selected objectives in a real-time manner. The optimization objectives (corresponding to different reward functions) are adaptively adjusted by the C-DQN, while the A-DQN selects the most suitable scheduling rules for real-time scheduling based on the selected optimization objectives, thus enabling a good compromise between different objectives in long-term scheduling.

AMDQN is based on two-hierarchy deep reinforcement learning, where the scheduling point t is defined as the moment of completion of each subtask. At each scheduling point t, the C-DQN takes as input the current state feature $s(t)$, and the output is the index $ri(t)$ of the reward function corresponding to the desired optimization objective in the current state. Subsequently, $s(t)$ and $ri(t)$ are merged and input into A-DQN, and the output is the optimal scheduling rule index $a(t)$ in the current state. According to the scheduling rule corresponding to $a(t)$, an unprocessed subtask is selected and assigned to a suitable manufacturing cell, thus completing the scheduling of a subtask. The system moves to the next scheduling point $t + 1$. The overall training procedure of the proposed AMDQN is provided

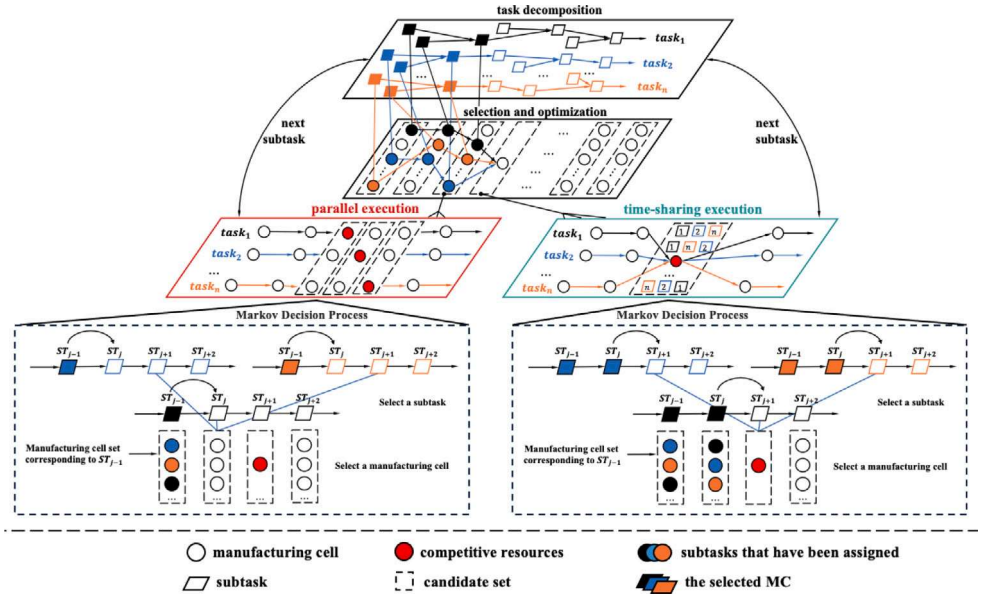


Fig. 2. The operation process of the proposed "parallel+serial" hybrid operation optimization scheduling strategy.

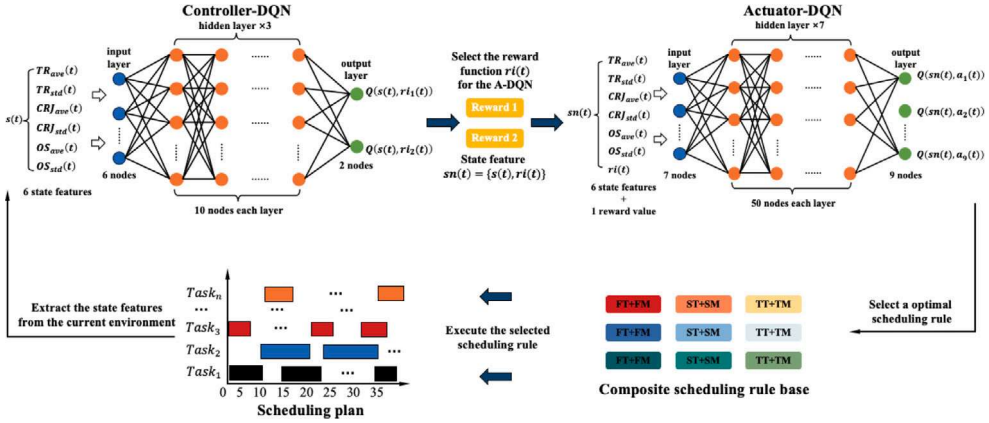


Fig. 3. The two-hierarchy deep reinforcement learning architecture of AMDQN.

in Algorithm 1. The detailed flow of algorithm training is shown in Fig. 4.

C-DQN is a deep neural network (DNN) consisting of five fully connected layers, including one input layer with 6 nodes (corresponding to 6 state features), three hidden layers with 10 nodes each layer, and one output layer with 2 nodes (corresponding to 2 reward algorithms of optimization objectives). It takes the current manufacturing environment state feature s(t) as inputs and outputs the Q-value of the reward algorithm corresponding to each optimization objective. A-DQN is a DNN consisting of nine fully connected layers, including one input layer with 7 nodes (corresponding to 6 state features and 1 reward algorithm index), seven hidden layers with 50 nodes each layer, and an output layer with 9 nodes (corresponding to nine scheduling rules). It generates the Q-value for each scheduling rule using state features and reward function indexes as inputs. C-DQN and A-DQN both use ReLU as the activation function. Finally, after executing the scheduling rules, the new state feature are computed and the reward values are obtained using the reward algorithm. These reward values will be used to update the parameters of the two DQNs.

4.3. State features definition

State features reflect the current state of the manufacturing environment in deep Q-learning and have a very important impact on the next action to be taken. In order to accurately reflect the current state of the manufacturing environment, this paper extracts six state features that are closely related to manufacturing time and manufacturing cost to fully represent a decision point. It consists of the average estimated tardiness rate TR_{ave} for all tasks, the standard deviation TR_{std} of the estimated tardiness rate for all tasks, the average completion rate CRJ_{ave} for all tasks, the standard deviation CRJ_{std} of the completion rate for all tasks, the average estimated overspend rate OS_{ave} for all tasks, and the standard deviation OS_{std} of the estimated overspend rate for all tasks. The definitions are Eqs. (6)–(11) respectively.

$$TR_{ave} = \frac{\sum_{i=1}^{n} TR_i}{n} \tag{6}$$

$$TR_{std} = \sqrt{\frac{\sum_{i=1}^{n} (TR_i - TR_{ave})}{n}} \tag{7}$$

$$CRJ_{ave} = \frac{\sum_{i=1}^{n} CRJ_i}{n} \tag{8}$$

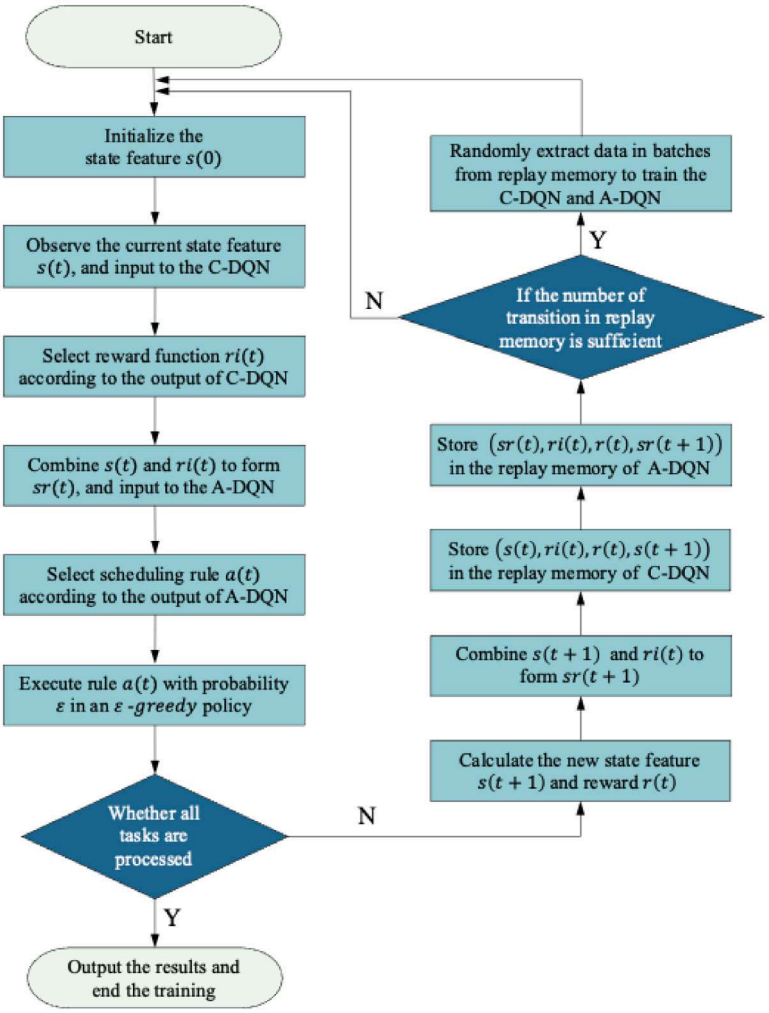


Fig. 4. The training process of the proposed AMDQN.

$$CRJ_{std} = \sqrt{\frac{\sum_{i=1}^{n}(CRJ_i - CRJ_{ave})}{n}} \tag{9}$$

$$OS_{ave} = \frac{\sum_{i=1}^{n} OS_i}{n} \tag{10}$$

$$OS_{std} = \sqrt{\frac{\sum_{i=1}^{n}(OS_i - OS_{ave})}{n}} \tag{11}$$

The above state feature take values in the range [0,1]. Training the model using the mean and standard deviation of these state features will facilitate the generalization of AMDQN and easily extend and generalize it to different untrained scenarios.

4.4. The composite scheduling rules

Due to the complexity and dynamics of the manufacturing process, no single scheduling rule is optimal in any manufacturing environment. Therefore, the most suitable scheduling rules should be used for different manufacturing resource conditions. In fact, there are two key issues when it comes to scheduling. How to first determine the next task that should be processed and then how to assign it to the appropriate manufacturing cell. For this reason, three scheduling rules are designed for the task selection problem and the manufacturing cell selection problem respectively. The combination of scheduling rules for tasks and manufacturing cells resulted in nine composite scheduling rules, the combination process of which is shown in Fig. 5.

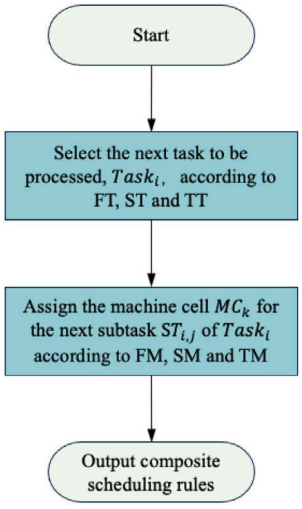


Fig. 5. The training process of the proposed AMDQN.

Algorithm 1: The overall training procedure of the proposed AMDQN

Input: number of tasks (n), number of subtasks per task (n_i), number of available manufacturing cell per subtask (e_{n_i})
Output: training results (IGD and HV)

Initialize parameters $\{\theta_1, \theta_2\}$ for the C-DQN and A-DQN respectively;
for *epoch = 1:L* **do**
  Generate a new training instance with random setting of n, n_i, and e_{n_i};
  Initialize the initial state feature $s(0)$;
  for $t = 0 : T$ *(T is the total number of subtasks to be scheduled)* **do**
    Observe the current state feature $s(t)$;
    input $s(t)$ into the C-DQN;
    Select the reward function $ri(t)$ corresponding to the objective to be optimized;
    Combine $s(t)$ and $ri(t)$ to form $sr(t)$, which is then input into the second DQN;
    Select scheduling rule $a(t)$ from the nine composite scheduling rules;
    With probability ϵ randomly execute scheduling rule $a(t)$ from the nine scheduling rules;
    Otherwise execute the scheduling rule $a(t)$;
    Calculate the new state feature value $s(t+1)$ and calculate the reward $r(t)$ by algorithm 8 or 9;
    Combine $s(t+1)$ and $ri(t)$ to form $sr(t+1)$;
    Store transition $(s(t), ri(t), r(t), s(t+1))$ in the replay memory of the C-DQN;
    Store transition $(sr(t), a(t), r(t), sr(t+1))$ in the replay memory of the A-DQN;
    if $t \geq$ *batchsize* **then**
      randomly extract data in batches from the replay memory to train the C-DQN and A-DQN respectively;
    end
    Every C steps update the parameters $\{\theta_1, \theta_2\}$ of the C-DQN and A-DQN;
  end
end

4.4.1. Scheduling rules for tasks

(1) First scheduling rule for tasks

The first scheduling rule for tasks (FT) first identifies the set of estimated delayed tasks $TR_{task}(t)$ and the set of all uncompleted tasks $UC_{task}(t)$ at the scheduling point t. If $TR_{task}(t)$ is empty, the tasks in $UC_{task}(t)$ are sorted according to the completion rate CRJ_i and the next subtask of the task $Task_i$ with the lowest completion rate $((SN_i(t))/n_i)$ is selected for execution. If $TR_{task}(t)$ is not empty, the next subtask of the task $Task_i \in TR_{task}(t)$ with the longest estimated overdue time is selected. Therefore, FT prefers to select tasks with high delivery overruns and relatively low completion rates, which help to shorten the manufacturing completion time and meet the final delivery period conditions. The process of FT is given in Algorithm 2.

(2) Second scheduling rule for tasks

The second scheduling rule for tasks (ST) first identifies the set of estimated delayed tasks $TR_{task}(t)$ and the set of all uncompleted tasks $UC_{task}(t)$ at the scheduling point t. If $TR_{task}(t)$ is not empty, the next subtask of task $Task_i \in TR_{task}(t)$ with the longest estimated overdue time is selected. Otherwise, the next subtask of task $Task_i \in TR_{task}(t)$, which is closest to the delivery deadline, is selected. Therefore, ST prefers tasks with tight delivery times, which help to shorten the manufacturing completion time and meet the final delivery period conditions. The process of ST is given in Algorithm 3.

Algorithm 2: The procedure of FT

Input: number of subtasks for each task (n_i), number of subtasks that have been processed for each task (SN_i), due date for each task (D_i), manufacturing time of $ST_{i,j}$ on MC_k ($t_{i,j,k}$))
Output: the next task to processed

$TR_{task}(t) \leftarrow \{i | SN_i(t) < n_i \&\& D_i < (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}})\}$;
$UC_{task}(t) \leftarrow \{i | SN_i(t) < n_i\}$;
if *isempty*$(TR_{task}(t))$ **then**
  $Task_i \leftarrow argmin_{i \in UC_{task}(t)} \frac{SN_i(t)}{n_i}$;
else
  $Task_i \leftarrow argmax_{i \in UC_{task}(t)} (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}} - D_i)$;
end
return $Task_i$;

Algorithm 3: The procedure of ST

Input: number of subtasks for each task (n_i), number of subtasks that have been processed for each task (SN_i), due date for each task (D_i), manufacturing time of $ST_{i,j}$ on MC_k ($t_{i,j,k}$))
Output: the next task to processed

$TR_{task}(t) \leftarrow \{i | SN_i(t) < n_i \&\& D_i < (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}})\}$;
if *isempty*$(TR_{task}(t))$ **then**
  $Task_i \leftarrow argmin_{i \in UC_{task}(t)} (D_i - CT_{i,SN_i(t)})$;
else
  $Task_i \leftarrow argmax_{i \in UC_{task}(t)} (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}} - D_i)$;
end
return $Task_i$;

(3) Third scheduling rule for tasks

The third scheduling rule for tasks (TT) first identifies the set of estimated delayed tasks $TR_{task}(t)$ and the set of all uncompleted tasks $UC_{task}(t)$ at the scheduling point t. If $TR_{task}(t)$ is not empty, the next subtask of task $Task_i \in TR_{task}(t)$ with the longest estimated overdue time is selected. Otherwise, the next subtask of task $Task_i \in TR_{task}(t)$ with the highest cost overspend is selected. TT prefers to control the manufacturing cost of each task, which helps to reduce the total manufacturing cost. The process of TT is given in Algorithm 4.

4.4.2. Scheduling rules for manufacturing cells

After selecting the subtasks $ST_{i,j}$ to be processed, the second problem is to select the appropriate manufacturing cell to process it.

(1) First scheduling rule for manufacturing cells

The first scheduling rule for manufacturing cells (FM) sorts the manufacturing cells according to their earliest available time and then selects the manufacturing cell MC_k with the earliest available time to perform the next subtask of task $Task_i$, where $k = argmin_{k \in AM_{MC}(t)} (\max(CTM_k(t), CT_{i,j-1}, A_i))$. This will reduce waiting time and minimize

Algorithm 4: The procedure of TT

Input: number of subtasks for each task (n_i), number of subtasks that have been processed for each task (SN_i), due date for each task (D_i), manufacturing time of $ST_{i,j}$ on MC_k ($t_{i,j,k}$)),manufacturing cost of $ST_{i,j}$ on MC_k ($c_{i,j,k}$))

Output: the next task to processed

$TR_{task}(t) \leftarrow \{i | SN_i(t) < n_i \&\& D_i < (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}})\}$;

if *isempty*$(TR_{task}(t))$ **then**

  $Task_i \leftarrow argmax_{i \in UC_{task}(t)} \frac{\sum_{j=1}^{SN_i(t)} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}}}{\sum_{j=1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}}}$;

else

  $Task_i \leftarrow argmax_{i \in UC_{task}(t)} (CT_{i,SN_i(t)} + \sum_{j=SN_i(t)+1}^{n_i} \frac{\sum_{k \in e_{n_i}} t_{i,j,k}}{e_{n_i}} - D_i)$;

end

return $Task_i$;

total manufacturing time while also improving manufacturing resource utilization. The process of FM is shown in Algorithm 5.

Algorithm 5: The procedure of FM

Input: the next subtask $ST_{i,j}$ of the select task $task_i$, manufacturing cost of $ST_{i,j}$ on MC_k ($c_{i,j,k}$)),the actual completion time of $ST_{i,j}$ ($CT_{i,j}$), the completion time of the last subtask assigned on MC_k (CTM_k)

Output: the manufacturing cell MC_k that performs $ST_{i,j}$

$AM_{MC}(t) \leftarrow \{k | t_{i,j,k} \neq 0 \&\& ((t, t + t_{i,j,k}) \cap OC_t = \emptyset)\}$;

$MC_k \leftarrow argmin_{k \in AM_{MC}(t)}(\max(CTM_k(t), CT_{i,j-1}, A_i))$;

return MC_k;

(2) Second scheduling rule for manufacturing cells

The second scheduling rule for manufacturing cells (SM) takes into account the manufacturing time of the manufacturing cell in addition to their earliest available time. The manufacturing cell MC_k with the earliest completion time is selected to perform the next subtask of task $Task_i$ based on the ordering of the completion time of the task by each manufacturing cell, where $k = argmin_{k \in AM_{MC}(t)}(\max(CTM_k(t), CT_{i,j-1}, A_i) + t_{i,j,k})$. This will reduce manufacturing time, increase efficiency and minimize total manufacturing time. The process of SM is given in Algorithm 6.

Algorithm 6: The procedure of SM

Input: the next subtask $ST_{i,j}$ of the select task $task_i$, manufacturing cost of $ST_{i,j}$ on MC_k ($c_{i,j,k}$)),the actual completion time of $ST_{i,j}$ ($CT_{i,j}$), the completion time of the last subtask assigned on MC_k (CTM_k)

Output: the manufacturing cell MC_k that performs $ST_{i,j}$

$AM_{MC}(t) \leftarrow \{k | t_{i,j,k} \neq 0 \&\& ((t, t + t_{i,j,k}) \cap OC_t = \emptyset)\}$;

$MC_k \leftarrow argmin_{k \in AM_{MC}(t)}(\max(CTM_k(t), CT_{i,j-1}, A_i) + t_{i,j,k})$;

return MC_k;

(3) Third scheduling rule for manufacturing cells

The third scheduling rule for manufacturing cells (TM) sorts the manufacturing cells based on their manufacturing costs and selects the manufacturing cell M_k with the lowest manufacturing cost to perform the next subtask of task $Task_i$, where $k = argmin_{k \in AM_{MC}(t)}(c_{i,j,k})$. This will help to reduce the total cost of task completion. The process of TM is given in Algorithm 7.

Algorithm 7: The procedure of TM

Input: the next subtask $ST_{i,j}$ of the select task $task_i$, manufacturing cost of $ST_{i,j}$ on MC_k ($c_{i,j,k}$)),the actual completion time of $ST_{i,j}$ ($CT_{i,j}$), the completion time of the last subtask assigned on MC_k (CTM_k)

Output: the manufacturing cell MC_k that performs $ST_{i,j}$

$AM_{MC}(t) \leftarrow \{k | t_{i,j,k} \neq 0 \&\& ((t, t + t_{i,j,k}) \cap OC_t = \emptyset)\}$;

$MC_k \leftarrow argmin_{k \in AM_{MC}(t)}(c_{i,j,k})$;

return MC_k;

4.5. Reward algorithm

To achieve multi-objective optimization, two different reward algorithms are designed in this paper. These two reward algorithms optimize the total manufacturing time and total manufacturing cost by optimizing two metrics, estimated tardiness rate TR_{ave} and estimated overspend rate OS_{ave}, respectively. The pseudocodes of the two reward algorithms are shown in Algorithms 8 and 9. $TR_{ave}(t)$, $TR_{ave}(t+1)$, $OS_{ave}(t)$ and $OS_{ave}(t+1)$ can be calculated by Eqs. (6) and (10).

Algorithm 8: Reward algorithm for optimizing makespan

Input: $TR_{ave}(t)$, $TR_{ave}(t+1)$

Output: $r(t)$

if $TR_{ave}(t+1) < TR_{ave}(t)$ **then**

  $r(t) \leftarrow 1$;

else

  if $TR_{ave}(t+1) < TR_{ave}(t) * 1.1$ **then**

    $r(t) \leftarrow 0$;

  else

    $r(t) \leftarrow -1$;

  end

end

return $r(t)$;

Algorithm 9: Reward algorithm for optimizing manufacturing cost

Input: $OS_{ave}(t)$, $OS_{ave}(t+1)$

Output: $r(t)$

if $OS_{ave}(t+1) < OS_{ave}(t)$ **then**

  $r(t) \leftarrow 1$;

else

  if $OS_{ave}(t+1) < OS_{ave}(t) * 1.1$ **then**

    $r(t) \leftarrow 0$;

  else

    $r(t) \leftarrow -1$;

  end

end

return $r(t)$;

The reward $r(t)$ for each scheduling point t is assigned by determining whether the metric chosen at moment $t+1$ after the adoption of the scheduling rule is superior to its value at moment t. Two metrics, estimated tardiness rate $TR_{ave}(t)$ and estimated overspend rate $OS_{ave}(t)$, as well as three reward values of 1, 0, and −1 are designed to guide the A-DQN to learn and select the scheduling rule that has the optimal effect on shortening the total manufacturing time and reducing

Table 3
Parameter settings for training benchmarks.

Parameter	Range
Number of tasks (n)	[10,30]
Number of subtasks per task (n_i)	[3,10]
Number of available manufacturing cell per subtask (e_{n_i})	[10,30]
Manufacturing time of a subtask on an available manufacturing cell ($t_{i,j,k}$)	[20,50]
Manufacturing cost of a subtask on an available manufacturing cell ($c_{i,j,k}$)	70-$t_{i,j,k}$

Table 4
Parameter settings of the training algorithm.

Parameter	Range
Number of training epochs L	200
Replay memory size of C-DQN	16
Replay memory size of A-DQN	200
Minibatch size for gradient descent	16
Initial value of ϵ and the decrement after each scheduling point	0.6 and 0.0001
Discount factor γ	0.95
Optimizer	Adam

the total manufacturing cost according to the changing state of the manufacturing environment. By adaptively choosing different reward algorithms at different scheduling points, a good compromise between the two objectives of total manufacturing time and total manufacturing cost can be achieved in long-term scheduling.

In summary, a hybrid optimized scheduling strategy of "parallel + serial" is adopted to minimize the waiting time while eliminating conflicts by using serial and parallel operation for single and multiple optional manufacturing cell, respectively. AMDQN is constructed using a two-hierarchy deep reinforcement learning architecture containing C-DQN and A-DQN, to solve AMMS-RCCT by hierarchical decision-making of multiple objective optimization compromise with the selection of optimal scheduling rules. Meanwhile, two metrics, estimated tardiness rate and estimated overspend rate, are introduced into the optimization reward algorithm to guide the A-DQN to learn and adjust the scheduling rules according to the state changes of the manufacturing environment.

5. Numerical experiments

5.1. Parameter settings

Due to the large diversity and variability in the descriptions and definitions of manufacturing tasks and manufacturing cells, there is not yet a common standardized test set and benchmarks for it. Therefore, the training and testing benchmarks used in this paper are randomly generated. To improve the generality of AMDQN for different manufacturing environments, instances with different parameter settings are fed to it during each round of training. Each of these instances was randomly generated based on the parameters listed in Table 3. On the basis of the parameter settings in [48,51], combined with the characteristics of the problem in this paper, the parameters of the training algorithm are set in detail after preliminary experiments and adjustments, as shown in Table 4.

5.2. Performance metrics

In evaluating the performance of multi-objective optimization algorithms, there are two main aspects: convergence and diversity. To evaluate the effectiveness of the proposed AMDQN. In this paper, two different evaluation metrics, inverse generational distance (IGD) and hypervolume (HV), are used. These two metrics are comprehensive metrics, both reflecting both the convergence and diversity of the solution sets obtained, and are defined as follows.

(1) Inverted generational distance (IGD)

$$IGD(X,P)=\frac{\sum_{x\in P^*} d(x,X)}{|P|} \tag{12}$$

where $d(x,X)$ denotes the minimum Euclidean distance from a solution $x\in P$ to a solution in X, and $|P|$ denotes the number of solutions in P. The smaller value of IGD indicates that the non-dominated solution set is closer to the true pareto-optimal front and more uniformly distributed, and the convergence and diversity of the solution set is better.

(2) Hypervolume (HV)

$$HV(X,P)=\bigcup_{x\in X}^{x} v(x,P) \tag{13}$$

where $v(x,P)$ denotes the hypervolume of the space formed between the solution x and the reference point P in the set of non-dominated solutions X. The larger value of HV indicates that the non-dominated solution set is closer to the true pareto-optimal front and more uniformly distributed, and the convergence and diversity of the solution set is better.

5.3. Results analysis

In order to validate the effectiveness and generality of the proposed AMDQN, a variety of experimental comparisons were conducted in different manufacturing environments. In order to simulate realistic different manufacturing environments, several sets of instances with different parameter settings of n, n_i and e_{n_i} are set up. Where the number of initial tasks $n\in[10,20,30]$, the number of subtasks for each task $n_i\in[3,5,10]$, and the number of candidate manufacturing cells corresponding to each subtask $e_{n_i}\in[10,20,30]$. The final result was 27 test case combinations of different sizes and 10 runs of each.

5.3.1. Comparisons with the proposed composite scheduling rules

To verify the effectiveness and generality of AMDQN, the results of AMDQN with each composite scheduling rule on 27 instances are compared. Also, to further confirm whether AMDQN is capable of adaptively selecting the optimal scheduling rule, a comparison with a random selection method is added. The IGD and HV values obtained from 10 repeated runs of each method on different instances are shown in Tables 5 and 6, with the best results highlighted in bold. The average values of IGD and HV for 10 runs of AMDQN and each composite scheduling rule on 27 instances are shown in Fig. 6. Fig. 7 illustrates the Pareto-optimal fronts for some representative examples obtained by the AMDQN and each composite scheduling rule.

The comparison results in Tables 5 and 6, and Fig. 6 demonstrate that AMDQN performs better on almost all instances compared to using each scheduling rule alone and achieves a significant advantage over the average of all instances. This validates the effectiveness of AMDQN, i.e., the ability to adaptively select the optimal scheduling rule in various situations. It also reflects the fact that there is no single rule suitable for all production environments, but rather the most suitable scheduling rule should be selected in real time based on the state of the manufacturing process. Besides, AMDQN shows better results on all instances compared to the random selection method. This further confirms the effectiveness and generality of AMDQN for multi-task multi-objective adaptive scheduling in dynamic manufacturing environments based on the state of manufacturing tasks and manufacturing cells.

5.3.2. Comparisons with classical scheduling rules

To further confirm the advantages of AMDQN, it is compared with five classical scheduling rules that are widely used, including FIFO: First in First Out, EDD: Earliest Due Date first, MRT: Most Remaining Processing Time first, SPT: Shortest Processing Time first, LPT: Longest Processing Time first. Since these scheduling rules only select the

Table 5
Comparison of IGD values obtained by the AMDQN and each composite rule.

n	n_i	e_{n_i}	FT+FM	FT+SM	FT+TM	ST+FM	ST+SM	ST+TM	TT+FM	TT+SM	TT+TM	Random	AMDQN
10	3	10	1.46E+02	2.08E+02	1.49E+02	1.32E+02	1.57E+02	1.41E+02	1.14E+02	2.08E+02	1.33E+02	7.63E+01	**4.34E+01**
		20	1.86E+02	1.77E+02	1.45E+02	1.87E+02	1.70E+02	1.77E+02	1.81E+02	1.77E+02	1.91E+02	1.18E+02	**3.50E+01**
		30	1.77E+02	2.06E+02	1.91E+02	1.76E+02	2.21E+02	1.72E+02	1.90E+02	2.06E+02	1.93E+02	1.23E+02	**6.77E+01**
	5	10	4.34E+02	1.56E+02	**6.37E+01**	4.28E+02	1.53E+02	8.02E+01	2.92E+02	1.56E+02	6.82E+01	6.99E+01	6.51E+01
		20	3.79E+02	3.34E+02	2.83E+02	3.70E+02	3.44E+02	2.45E+02	3.48E+02	3.34E+02	2.74E+02	2.39E+02	**2.57E+01**
		30	2.16E+02	4.27E+02	4.15E+02	2.15E+02	4.21E+02	3.23E+02	2.87E+02	4.27E+02	3.70E+02	3.03E+02	**2.09E+02**
	10	10	4.37E+02	6.33E+02	5.24E+02	3.96E+02	6.05E+02	5.71E+02	4.81E+02	6.33E+02	6.06E+02	4.09E+02	**3.20E+02**
		20	7.42E+02	3.19E+02	2.69E+02	7.22E+02	3.38E+02	2.75E+02	6.07E+02	3.19E+02	2.76E+02	3.23E+02	**2.24E+02**
		30	5.38E+02	7.83E+02	7.26E+02	5.41E+02	7.45E+02	8.35E+02	6.01E+02	7.83E+02	7.59E+02	4.99E+02	**2.84E+02**
20	3	10	3.14E+02	3.01E+02	3.01E+02	3.20E+02	3.04E+02	3.14E+02	3.30E+02	3.01E+02	3.00E+02	2.48E+02	**4.19E+01**
		20	4.10E+02	3.75E+02	3.44E+02	4.12E+02	3.75E+02	3.22E+02	3.97E+02	3.75E+02	3.38E+02	2.53E+02	**1.07E+02**
		30	5.12E+02	3.51E+02	3.39E+02	5.18E+02	3.49E+02	3.36E+02	4.82E+02	3.51E+02	3.22E+02	3.14E+02	**1.19E+02**
	5	10	6.58E+02	2.73E+02	2.20E+02	6.51E+02	2.64E+02	5.56E+02	5.10E+02	2.73E+02	2.68E+02	2.39E+02	**9.55E+01**
		20	3.30E+02	8.02E+02	6.84E+02	2.96E+02	7.94E+02	7.41E+02	4.16E+02	8.02E+02	6.69E+02	5.05E+02	**2.97E+02**
		30	9.54E+02	4.34E+02	3.43E+02	9.32E+02	4.82E+02	3.32E+02	8.27E+02	4.34E+02	3.26E+02	4.13E+02	**2.47E+02**
	10	10	9.63E+02	1.19E+03	1.03E+03	9.67E+02	1.15E+03	1.12E+03	9.64E+02	1.19E+03	1.11E+03	8.50E+02	**4.82E+02**
		20	2.46E+03	1.98E+03	1.83E+03	2.44E+03	1.97E+03	2.04E+03	2.26E+03	1.98E+03	2.50E+03	1.80E+03	**1.41E+03**
		30	1.46E+03	1.17E+03	1.10E+03	1.41E+03	1.36E+03	1.34E+03	1.33E+03	1.17E+03	1.18E+03	1.00E+03	**4.05E+02**
30	3	10	9.15E+02	3.93E+02	3.58E+02	9.27E+02	4.13E+02	8.33E+02	6.89E+02	3.93E+02	1.03E+03	3.45E+02	**2.46E+02**
		20	6.59E+02	3.77E+02	4.01E+02	6.25E+02	5.24E+02	7.25E+02	5.87E+02	3.77E+02	4.01E+02	3.22E+02	**1.51E+02**
		30	4.89E+02	3.70E+02	4.03E+02	4.61E+02	4.04E+02	4.00E+02	4.08E+02	3.70E+02	3.81E+02	2.61E+02	**1.65E+02**
	5	10	1.30E+03	6.74E+02	6.56E+02	1.30E+03	6.89E+02	1.06E+03	1.11E+03	6.74E+02	1.52E+03	6.09E+02	**4.62E+02**
		20	1.03E+03	9.29E+02	8.07E+02	9.97E+02	9.45E+02	7.44E+02	9.21E+02	9.29E+02	8.04E+02	7.52E+02	**3.31E+02**
		30	6.09E+02	1.17E+03	9.85E+02	6.27E+02	1.19E+03	1.00E+03	6.97E+02	1.17E+03	1.04E+03	7.46E+02	**3.66E+02**
	10	10	1.96E+03	1.12E+03	1.05E+03	1.78E+03	1.12E+03	1.01E+03	1.61E+03	1.12E+03	1.01E+03	1.06E+03	**5.44E+02**
		20	4.37E+03	1.28E+03	1.12E+03	4.38E+03	1.32E+03	1.63E+03	3.41E+03	1.28E+03	2.91E+03	1.43E+03	**1.07E+03**
		30	3.08E+03	6.80E+02	4.39E+02	3.11E+03	7.27E+02	4.36E+02	2.24E+03	6.80E+02	4.68E+02	6.11E+02	**2.09E+02**

Table 6
Comparison of HV values obtained by the AMDQN and each composite rule.

n	n_i	e_{n_i}	FT+FM	FT+SM	FT+TM	ST+FM	ST+SM	ST+TM	TT+FM	TT+SM	TT+TM	Random	AMDQN
10	3	10	4.01E−02	5.88E−02	6.08E−02	4.47E−02	8.94E−02	8.26E−02	5.93E−02	5.88E−02	7.81E−02	1.02E−01	**1.03E−01**
		20	3.49E−02	3.56E−02	3.27E−02	3.44E−02	5.41E−02	3.01E−02	1.81E−02	3.56E−02	5.10E−02	5.97E−02	**6.78E−02**
		30	5.53E−02	1.20E−01	1.10E−01	5.56E−02	1.12E−01	1.32E−01	5.31E−02	1.20E−01	1.27E−01	1.35E−01	**1.54E−01**
	5	10	2.33E−02	7.93E−02	1.16E−01	2.12E−02	9.56E−02	1.22E−01	5.19E−02	7.93E−02	1.29E−01	1.24E−01	**1.24E−01**
		20	4.14E−02	6.11E−02	9.03E−02	4.99E−02	1.06E−01	6.30E−02	4.50E−02	6.11E−02	9.45E−02	1.28E−01	**1.40E−01**
		30	3.95E−02	1.55E−02	6.17E−02	4.01E−02	3.98E−02	6.86E−02	3.71E−02	1.55E−02	9.55E−02	1.19E−01	**1.19E−01**
	10	10	2.25E−02	0.00E+00	2.87E−02	2.03E−02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.77E−02	3.49E−02	**4.20E−02**
		20	3.79E−02	1.07E−01	9.59E−02	3.98E−02	7.79E−02	1.02E−01	3.27E−02	1.07E−01	**1.16E−01**	9.44E−02	1.14E−01
		30	4.80E−02	4.22E−02	7.24E−02	4.75E−02	4.23E−02	0.00E+00	3.40E−03	4.22E−02	1.16E−01	1.12E−01	**1.29E−01**
20	3	10	4.35E−02	1.13E−01	5.42E−02	4.28E−02	8.98E−02	6.10E−03	2.65E−02	**1.13E−01**	6.83E−02	9.50E−02	1.11E−01
		20	4.39E−02	1.16E−01	3.37E−02	4.96E−02	1.09E−01	4.81E−02	4.39E−02	1.16E−01	2.41E−02	9.32E−02	**1.23E−01**
		30	4.69E−02	1.26E−01	6.11E−02	4.47E−02	1.36E−01	5.24E−02	4.55E−02	1.26E−01	7.92E−02	1.08E−01	**1.40E−01**
	5	10	3.59E−02	7.34E−02	5.61E−02	3.79E−02	7.51E−02	0.00E+00	2.49E−02	7.34E−02	6.72E−02	7.57E−02	**8.79E−02**
		20	5.44E−02	1.27E−01	1.35E−01	5.87E−02	1.19E−01	1.13E−01	5.41E−02	1.27E−01	1.30E−01	1.29E−01	**1.55E−01**
		30	3.84E−02	5.32E−02	6.84E−02	3.29E−02	6.77E−03	4.55E−02	4.40E−02	5.32E−02	9.70E−02	9.63E−02	**9.97E−02**
	10	10	2.74E−02	3.85E−02	6.13E−02	2.56E−02	3.79E−02	5.63E−02	2.93E−02	3.85E−02	6.45E−02	7.34E−02	**7.46E−02**
		20	4.36E−02	5.41E−02	1.19E−01	4.68E−02	9.24E−02	0.00E+00	5.40E−02	5.41E−02	5.20E−02	1.33E−01	**1.34E−01**
		30	4.58E−02	4.07E−02	**1.34E−01**	4.20E−02	2.92E−02	0.00E+00	4.13E−02	4.07E−02	1.03E−01	1.07E−01	1.04E−01
30	3	10	5.45E−02	1.50E−01	1.35E−01	5.03E−02	1.53E−01	0.00E+00	8.36E−02	1.50E−01	4.81E−02	1.53E−01	**1.81E−01**
		20	7.30E−02	2.02E−01	2.19E−01	7.72E−02	2.26E−01	3.12E−02	8.84E−02	2.02E−01	2.11E−01	2.15E−01	**2.41E−01**
		30	4.21E−02	9.60E−02	5.94E−02	4.39E−02	9.88E−02	3.53E−02	5.11E−02	9.60E−02	6.25E−02	9.94E−02	**1.03E−01**
	5	10	5.31E−02	1.27E−01	9.39E−02	5.04E−02	1.23E−01	0.00E+00	5.37E−02	1.27E−01	4.35E−02	1.26E−01	**1.29E−01**
		20	3.35E−02	2.67E−02	0.00E+00	3.77E−02	3.10E−02	0.00E+00	5.81E−04	2.67E−02	1.56E−02	4.72E−02	**8.88E−02**
		30	3.66E−02	8.91E−02	1.00E−01	3.72E−02	8.51E−02	8.33E−02	4.34E−02	8.91E−02	9.60E−02	9.37E−02	**1.11E−01**
	10	10	3.33E−02	4.92E−02	6.25E−02	3.72E−02	4.63E−02	4.46E−02	2.92E−02	4.92E−02	6.17E−02	5.97E−02	**6.97E−02**
		20	6.56E−05	8.08E−02	9.55E−02	0.00E+00	8.03E−02	0.00E+00	2.42E−02	8.08E−02	4.73E−02	7.99E−02	**1.02E−01**
		30	0.00E+00	0.00E+00	3.20E−02	0.00E+00	0.00E+00	3.66E−02	1.15E−03	0.00E+00	3.38E−02	2.92E−02	**3.60E−02**

subtasks to be processed in the next step, they do not specify the manufacturing cell to be used to execute the selected subtasks. For a fair comparison, they are uniformly assigned to the manufacturing cell with the earliest available time. The results obtained after 10 repeated runs of each method on each instance are shown in Tables 7 and 8, with the best results in bold. The average values of IGD and HV for 10 repeated runs of AMDQN and classical scheduling rules on 27 instances are shown in Fig. 8. Fig. 9 illustrates the Pareto-optimal fronts for some representative examples obtained by the AMDQN and classical scheduling rules.

The comparison results in Tables 7 and 8, and Fig. 8 demonstrate that in most instances, AMDQN showed the best results on all metrics and performs optimally on the average of all instances. This is because these classical scheduling rules only consider the selection of subtasks,

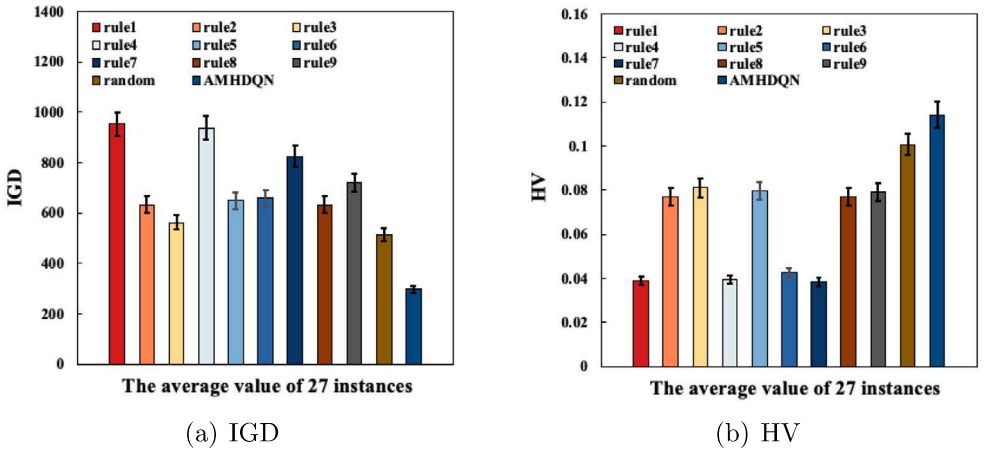


Fig. 6. Average values of IGD and HV obtained by the AMDQN and each composite rule.

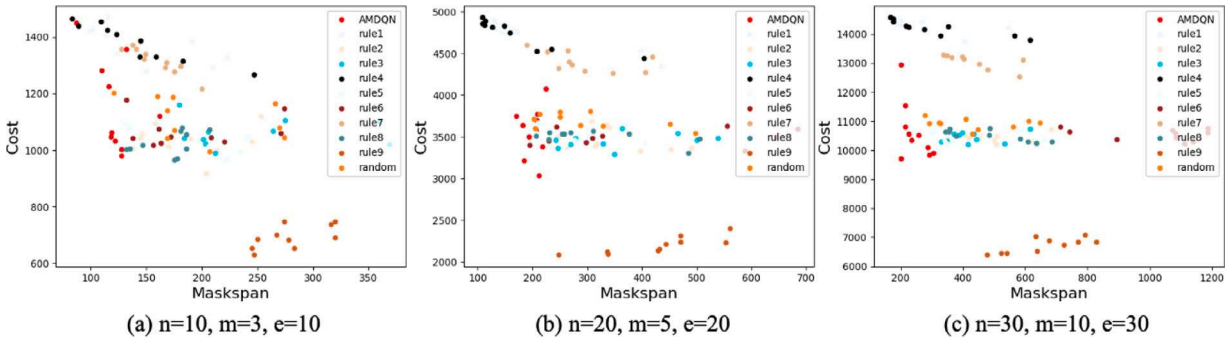


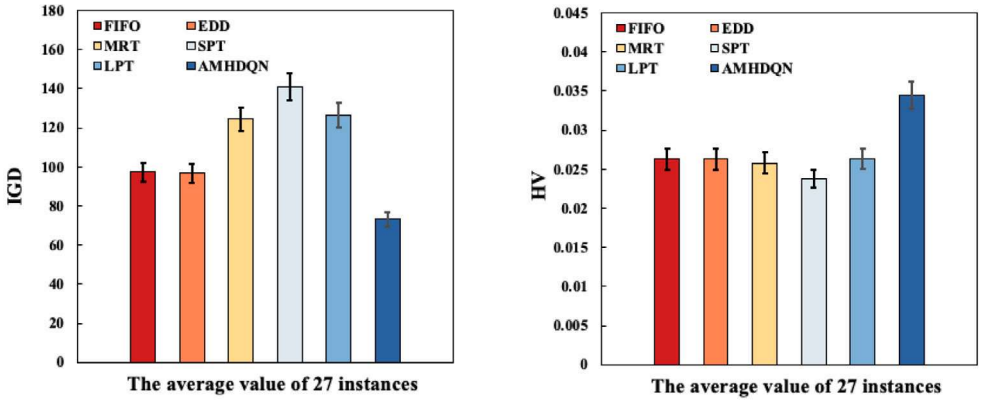
Fig. 7. Pareto-optimal fronts obtained by the AMDQN and each composite rule for some representative instances.

Table 7
Comparison of IGD values obtained by the AMDQN and classical scheduling rules.

n	n_i	e_{n_i}	FIFO	EDD	MRT	SPT	LPT	AMDQN
10	3	10	4.45E+01	4.45E+01	5.43E+01	5.67E+01	4.02E+01	**2.56E+01**
		20	2.98E+01	2.98E+01	6.14E+01	5.03E+01	3.16E+01	**2.47E+01**
		30	1.57E+02	1.57E+02	4.79E+01	4.05E+01	4.76E+01	**2.42E+01**
	5	10	9.75E+00	9.75E+00	9.47E+01	**9.74E+00**	2.22E+01	3.00E+01
		20	3.92E+01	3.92E+01	6.65E+01	1.04E+02	8.10E+01	**2.73E+01**
		30	5.88E+01	5.88E+01	7.03E+01	9.20E+01	1.26E+02	**3.13E+01**
	10	10	5.13E+01	5.13E+01	8.29E+01	5.80E+01	1.29E+02	**3.95E+01**
		20	9.81E+01	9.81E+01	7.79E+01	7.76E+01	7.57E+01	**5.07E+01**
		30	1.11E+02	1.11E+02	8.78E+01	8.70E+01	7.60E+01	**4.23E+01**
20	3	10	1.85E+01	**1.85E+01**	6.45E+01	5.49E+01	9.16E+01	2.94E+01
		20	5.87E+01	5.87E+01	1.37E+02	9.21E+01	6.21E+01	**5.74E+01**
		30	7.19E+01	7.19E+01	8.68E+01	8.72E+01	8.68E+01	**6.45E+01**
	5	10	1.49E+02	1.49E+02	1.18E+02	5.26E+01	1.09E+02	**4.64E+01**
		20	3.81E+01	2.83E+01	2.68E+02	2.48E+02	1.14E+02	**1.87E+01**
		30	8.52E+01	8.52E+01	8.89E+01	8.13E+01	1.07E+02	**2.75E+01**
	10	10	1.79E+02	1.79E+02	4.18E+00	2.85E+02	1.97E+02	**4.18E+01**
		20	3.48E+02	3.48E+02	3.66E+02	3.79E+02	**3.35E+02**	6.78E+02
		30	1.75E+02	1.75E+02	3.09E+02	2.44E+02	2.56E+02	**1.70E+02**
30	3	10	7.79E+01	7.79E+01	7.83E+01	1.21E+02	1.02E+02	**5.67E+01**
		20	8.60E+01	8.60E+01	**5.16E+01**	1.60E+02	1.12E+02	6.78E+01
		30	8.45E+01	8.45E+01	7.61E+01	7.68E+01	7.09E+01	**2.20E+01**
	5	10	1.16E+02	1.16E+02	2.30E+02	2.30E+02	3.03E+02	**3.72E+01**
		20	8.35E+01	**8.35E+01**	1.67E+02	1.13E+02	1.25E+02	1.37E+02
		30	1.12E+02	1.12E+02	2.24E+02	2.99E+02	1.47E+02	**9.65E+01**
	10	10	8.68E+01	8.68E+01	8.95E+01	1.76E+02	1.53E+02	**4.69E+01**
		20	7.20E+01	7.20E+01	5.27E+01	3.17E+02	8.81E+01	**2.60E+01**
		30	1.83E+02	1.83E+02	3.01E+02	2.06E+02	3.17E+02	**5.77E+01**

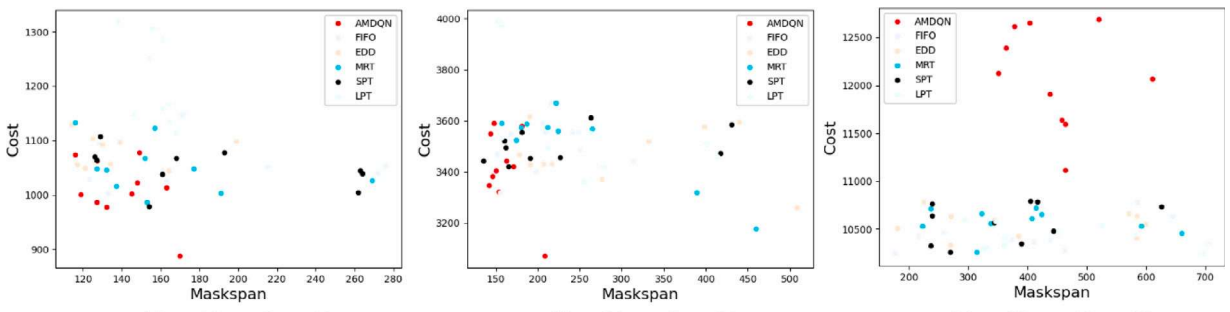
Table 8
Comparison of HV values obtained by the AMDQN and classical scheduling rules.

n	n_i	e_{n_i}	FIFO	EDD	MRT	SPT	LPT	AMDQN
10	3	10	5.53E−02	5.53E−02	5.01E−02	5.83E−02	6.92E−02	**8.07E−02**
		20	4.41E−02	4.41E−02	3.46E−02	3.98E−02	4.28E−02	**4.97E−02**
		30	1.24E−02	1.24E−02	2.20E−02	2.35E−02	1.73E−02	**3.92E−02**
	5	10	2.41E−02	2.41E−02	1.61E−02	2.44E−02	2.73E−02	**3.27E−02**
		20	3.59E−02	**3.59E−02**	1.40E−02	7.96E−03	2.23E−02	1.75E−02
		30	3.08E−02	3.08E−02	2.50E−02	2.41E−02	2.76E−02	**4.01E−02**
	10	10	1.89E−02	1.89E−02	2.89E−02	3.01E−02	3.55E−02	**4.10E−02**
		20	3.88E−02	3.88E−02	2.59E−02	3.77E−02	3.82E−02	**4.15E−02**
		30	3.49E−02	3.49E−02	4.10E−02	3.45E−02	4.14E−02	**4.51E−02**
20	3	10	1.95E−02	1.95E−02	1.31E−02	1.41E−02	7.49E−03	**3.04E−02**
		20	1.95E−02	1.95E−02	2.65E−02	2.62E−02	1.70E−02	**3.67E−02**
		30	2.01E−02	2.01E−02	2.81E−02	3.00E−02	4.79E−02	**4.96E−02**
	5	10	2.83E−02	2.83E−02	1.84E−02	1.49E−02	2.03E−02	**3.19E−02**
		20	1.77E−02	1.77E−02	1.93E−02	1.50E−02	1.99E−02	**2.01E−02**
		30	4.84E−02	4.84E−02	4.76E−02	4.83E−02	4.87E−02	**6.40E−02**
	10	10	3.84E−03	3.84E−03	8.28E−03	2.03E−03	0.00E+00	**1.69E−02**
		20	2.40E−02	2.40E−02	2.53E−02	2.80E−02	**4.89E−02**	4.04E−02
		30	4.03E−02	4.03E−02	5.07E−02	3.47E−02	4.58E−02	**5.58E−02**
30	3	10	1.72E−02	1.72E−02	2.09E−02	1.35E−02	3.20E−03	**2.09E−02**
		20	2.08E−02	2.08E−02	2.73E−02	1.70E−02	2.06E−02	**2.83E−02**
		30	2.97E−02	2.97E−02	2.97E−02	2.50E−02	2.92E−02	**3.76E−02**
	5	10	1.74E−02	1.74E−02	2.08E−02	1.87E−02	9.95E−03	**2.22E−02**
		20	1.68E−02	1.68E−02	1.86E−02	2.01E−02	1.91E−02	**2.32E−02**
		30	2.78E−02	2.78E−02	2.77E−02	2.08E−02	**3.20E−02**	1.83E−02
	10	10	2.39E−02	2.39E−02	**2.41E−02**	2.09E−03	9.92E−03	4.23E−03
		20	1.32E−02	1.32E−02	1.49E−02	8.05E−03	9.84E−03	**1.64E−02**
		30	2.57E−02	2.57E−02	1.77E−02	2.28E−02	0.00E+00	**2.59E−02**



(a) IGD (b) HV

Fig. 8. Average values of IGD and HV obtained by the AMDQN and classical scheduling rules.



(a) n=10, m=3, e=10 (b) n=20, m=5, e=20 (c) n=30, m=10, e=30

Fig. 9. Pareto-optimal fronts obtained by the AMDQN and classical scheduling rules for some representative instances.

Table 9
Comparison of IGD values obtained by the AMDQN and scheduling rules in existing research.

n	n_i	e_{n_i}	Scheduling rules in [51]	Scheduling rules in [48]	AMDQN
10	3	10	1.25E+02	1.11E+02	**3.16E+01**
		20	6.84E+01	2.79E+02	**2.22E+01**
		30	**2.56E+01**	3.26E+02	3.79E+01
	5	10	**4.76E+01**	3.35E+02	5.70E+01
		20	6.58E+01	1.82E+02	**5.01E+01**
		30	2.61E+02	8.13E+01	**5.90E+01**
	10	10	2.87E+02	1.95E+02	**5.26E+01**
		20	2.40E+02	1.50E+02	**1.10E+02**
		30	5.88E+02	5.23E+02	**1.15E+02**
20	3	10	1.55E+02	**9.35E+01**	4.10E+02
		20	2.62E+02	8.74E+01	**6.30E+01**
		30	2.59E+02	8.69E+01	**8.23E+01**
	5	10	3.32E+02	1.07E+02	**9.74E+01**
		20	3.66E+02	8.38E+02	**9.24E+01**
		30	2.20E+02	6.55E+02	**9.60E+01**
	10	10	6.50E+02	8.76E+02	**2.72E+01**
		20	4.00E+02	1.54E+03	**7.60E+01**
		30	5.29E+02	1.30E+03	**3.26E+02**
30	3	10	1.68E+02	4.33E+02	**9.19E+01**
		20	1.87E+02	3.54E+02	**7.73E+01**
		30	1.40E+02	9.79E+02	**1.08E+02**
	5	10	**3.69E+01**	7.30E+02	1.18E+02
		20	**9.95E+01**	8.41E+02	1.58E+02
		30	5.70E+02	6.04E+02	**1.58E+02**
	10	10	**3.09E+02**	1.57E+03	4.90E+02
		20	**3.91E+02**	2.49E+03	7.91E+02
		30	7.50E+02	1.42E+03	**2.94E+02**

Table 10
Comparison of HV values obtained by the AMDQN and scheduling rules in existing research.

n	n_i	e_{n_i}	Scheduling rules in [51]	Scheduling rules in [48]	AMDQN
10	3	10	6.97E−02	9.87E−02	**1.12E−01**
		20	4.12E−02	0.00E+00	**7.68E−02**
		30	**5.83E−02**	0.00E+00	4.78E−02
	5	10	5.02E−02	2.03E−02	**6.22E−02**
		20	**6.97E−02**	4.35E−02	5.87E−02
		30	0.00E+00	**1.12E−01**	5.92E−02
	10	10	0.00E+00	7.35E−02	**1.47E−01**
		20	0.00E+00	1.06E−02	**9.06E−02**
		30	0.00E+00	**5.34E−02**	2.63E−02
20	3	10	0.00E+00	0.00E+00	**8.70E−02**
		20	0.00E+00	**8.21E−02**	0.00E+00
		30	4.86E−03	**1.04E−01**	7.55E−02
	5	10	0.00E+00	0.00E+00	**9.34E−02**
		20	0.00E+00	**4.22E−02**	0.00E+00
		30	1.01E−01	4.75E−02	**1.39E−01**
	10	10	8.12E−03	1.53E−02	**5.99E−02**
		20	5.92E−02	2.26E−02	**9.65E−02**
		30	1.06E−01	4.58E−02	**1.08E−01**
30	3	10	6.22E−02	1.33E−02	**8.03E−02**
		20	5.89E−02	1.18E−02	**7.30E−02**
		30	**3.37E−02**	0.00E+00	2.45E−02
	5	10	**2.29E−02**	0.00E+00	2.02E−02
		20	3.26E−02	9.61E−04	**4.05E−02**
		30	8.03E−02	5.71E−02	**1.03E−01**
	10	10	3.68E−02	2.98E−02	**5.64E−02**
		20	0.00E+00	0.00E+00	**4.77E−02**
		30	0.00E+00	1.78E−02	**3.00E−02**

whereas the selection of a suitable manufacturing cell to perform the selected subtask is equally important as it determines the manufacturing time and manufacturing cost of that subtask. This demonstrates the superiority of the nine scheduling rules compared to the classical scheduling rules for scheduling in complex manufacturing environments, further validating the effectiveness of AMDQN in optimizing the total manufacturing time and total manufacturing cost.

5.3.3. Comparisons with scheduling rules in existing research

To verify the advantages of AMDQN over scheduling rules in existing research, it is compared with two recently proposed scheduling rules for the dynamic flexible job shop scheduling problem [48,51]. Each method was repeated 10 times on each instance and the experimental results are shown in Tables 9 and 10, with the best results highlighted in bold. The average values of IGD and HV for 10 runs of AMDQN and the scheduling rules in [48,51] on 27 instances are shown in Fig. 10. Fig. 11 illustrates the Pareto-optimal fronts for some representative examples obtained by the AMDQN and the scheduling rules in [48,51].

The comparison results in Tables 9 and 10, and Fig. 10 demonstrate that AMDQN achieves the best results in most of the cases and achieves a significant advantage over the average of all the instances compared to the scheduling rules of [48,51]. This is because these two scheduling rules do not consider manufacturing cost as an optimization objective. This verifies that AMDQN is able to achieve a better compromise between the two optimization objectives of total manufacturing time and total manufacturing cost. It should be noted that the scheduling rules of [48,51] obtain optimal results on some instances due to the fact that they focus more on the optimization objective of the manufacturing time, which leads to optimal results on the total manufacturing time.

5.3.4. Comparisons with meta-heuristics and exact algorithms

To further validate the adaptive capability of AMDQN in dynamic manufacturing environments, AMDQN is compared with traditional scheduling methods on 27 instances of different scales, including three

Table 11
Comparison of IGD nad HV values obtained by the AMDQN and MIP.

n	n_i	e_{n_i}	IGD		HV	
			MIP	**AMDQN**	MIP	**AMDQN**
3	3	3	**1.82E+01**	**1.82E+01**	**1.64E−02**	1.62E−02
		5	**2.80E+01**	**2.80E+01**	**2.99E−02**	1.97E−02
	5	3	**3.47E+01**	**3.47E+01**	1.68E−02	**2.62E−02**
		5	**3.82E+01**	**3.82E+01**	3.38E−02	**5.76E−02**
5	3	3	**4.20E+01**	**4.20E+01**	2.19E−02	**3.72E−02**
		5	**5.95E+01**	**5.95E+01**	3.31E−02	**4.15E−02**
	5	3	**7.73E+01**	**7.73E+01**	2.75E−02	**4.36E−02**
		5	**5.86E+01**	**5.86E+01**	3.80E−02	**4.62E−02**

latest meta-heuristics and an exact algorithm: hybrid adaptive differential evolution (HADE) algorithm [52], multi-objective evolutionary algorithm based on decomposition (MOEAD) algorithm [53], non-dominated sorting genetic algorithm II (NSGA-II) algorithm [54] and mixed integer programming (MIP) [55]. It is worth noting that even for the smallest scale of the 27 instances, MIP fails to solve in finite time. Therefore, the proposed AMDQN is compared with MIP on eight smaller instances. Each method was run 10 times on each instance and the comparison results are shown in Tables 11–13 with the best results highlighted in bold. The average values of IGD and HV for AMDQN and the three meta-heuristic algorithms for 10 repeated runs on 27 instances are shown in Fig. 12. Fig. 13 illustrates the Pareto-optimal fronts for some representative instances obtained by the AMDQN and three meta-heuristics.

The comparison of results in Table 11 shows that the proposed AMDQN is able to obtain the same performance as the MIP algorithm on most instances. However, the proposed AMDQN is also capable of scheduling on large-scale instances, which is more relevant to the needs of real-world environments. The comparison results from Tables 12 and 13 show that AMDQN outperforms the other three metaheuristics on most instances. We found that MOEAD shows the worst results on almost all instances, outperforming NSGA-II and HADE on only a few

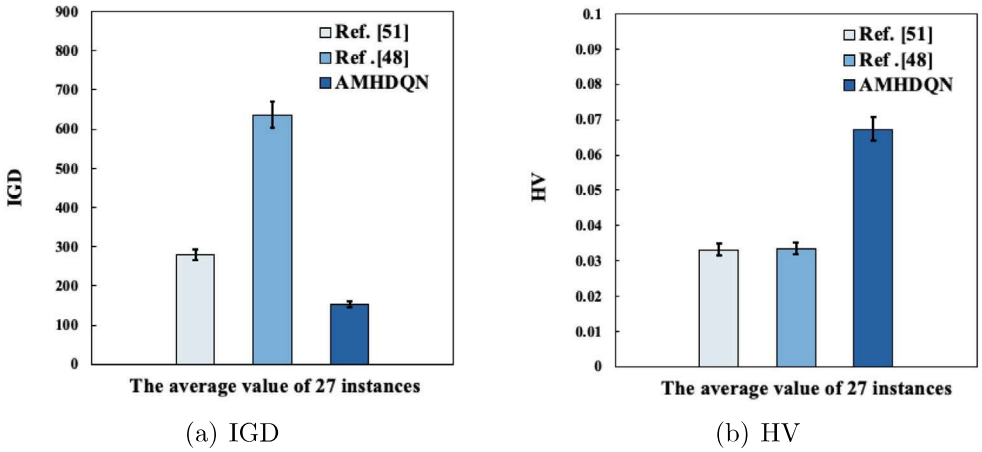


Fig. 10. Average values of IGD and HV obtained by the AMDQN and scheduling rules in existing research.

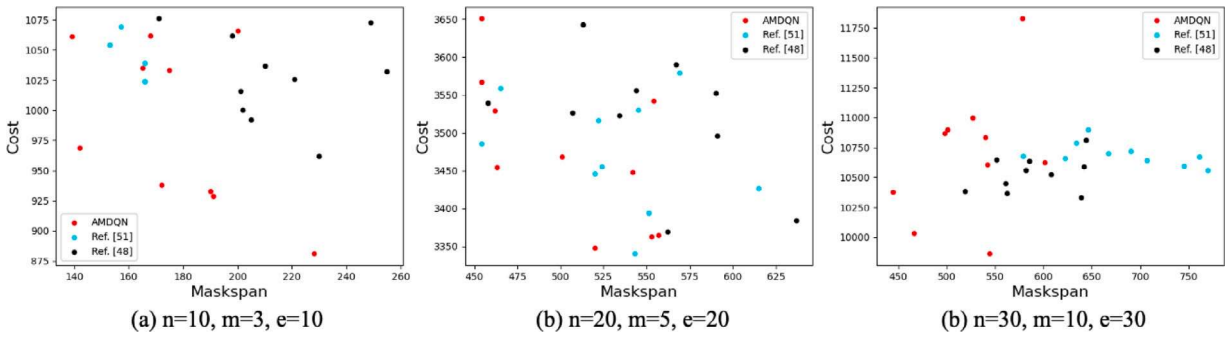


Fig. 11. Pareto-optimal fronts obtained by the AMDQN and scheduling rules in existing research for some representative instances.

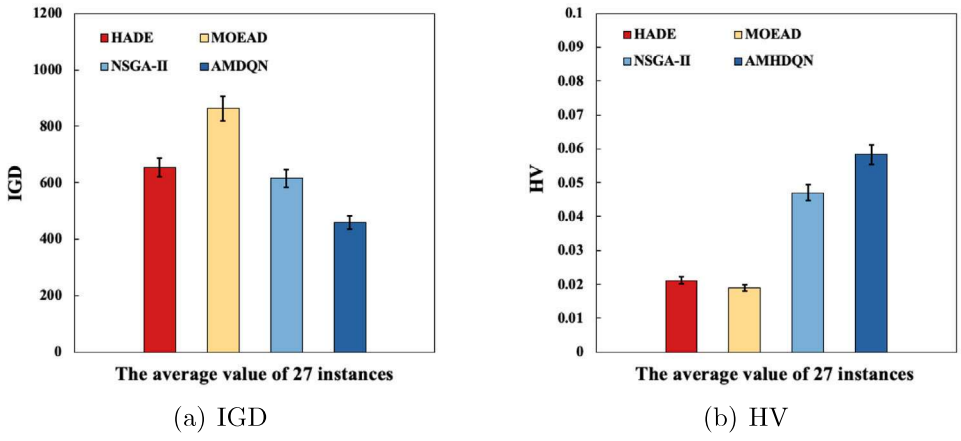


Fig. 12. Average values of IGD and HV obtained by the AMDQN and three meta-heuristics.

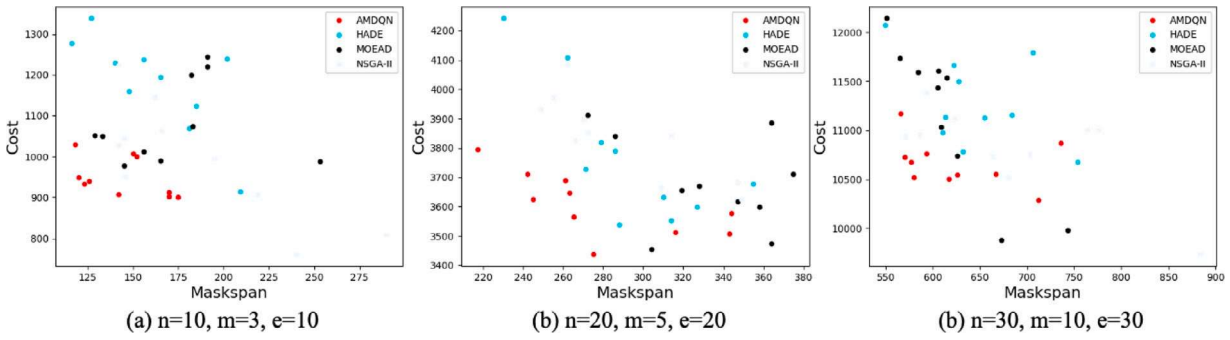


Fig. 13. Pareto-optimal fronts obtained by the AMDQN and three meta-heuristics for some representative instances.

Table 12
Comparison of IGD values obtained by the AMDQN and three meta-heuristics.

n	n_i	e_{n_i}	HADE	MOEAD	NSGA-II	**AMDQN**
10	3	10	1.98E+02	2.46E+02	1.91E+02	**1.31E+02**
		20	2.26E+02	6.18E+01	6.46E+01	**4.34E+01**
		30	1.94E+02	1.10E+02	**6.18E+01**	**6.18E+01**
	5	10	**9.66E+01**	3.97E+02	**9.66E+01**	1.36E+02
		20	2.91E+02	4.00E+02	2.32E+02	**2.44E+02**
		30	5.44E+02	5.29E+02	4.75E+02	**3.35E+02**
	10	10	3.27E+02	8.75E+02	2.74E+02	**2.01E+02**
		20	3.38E+02	4.02E+02	**0.00E+00**	6.62E+02
		30	4.12E+02	3.79E+02	5.94E+02	**3.25E+02**
20	3	10	3.89E+02	5.52E+02	4.89E+02	**3.06E+02**
		20	3.54E+02	4.60E+02	3.66E+02	**2.40E+02**
		30	4.25E+02	5.15E+02	4.33E+02	**2.86E+02**
	5	10	4.15E+02	1.00E+03	4.68E+02	**2.95E+02**
		20	5.86E+02	5.86E+02	**2.36E+02**	**2.36E+02**
		30	5.81E+02	9.08E+02	4.90E+02	**3.58E+02**
	10	10	1.28E+03	1.19E+03	**7.97E+02**	**7.97E+02**
		20	1.30E+03	1.53E+03	1.16E+03	**8.19E+02**
		30	1.12E+03	1.66E+03	8.31E+02	**6.52E+02**
30	3	10	3.27E+02	6.45E+02	5.39E+02	**2.90E+02**
		20	7.95E+02	3.16E+02	5.03E+02	**2.74E+02**
		30	4.87E+02	8.19E+02	4.70E+02	**3.19E+02**
	5	10	6.63E+02	1.20E+03	1.13E+03	**6.12E+02**
		20	6.60E+02	1.44E+03	9.70E+02	**5.44E+02**
		30	8.38E+02	1.39E+03	1.14E+03	**6.59E+02**
	10	10	**1.36E+03**	1.97E+03	**1.36E+03**	**1.36E+03**
		20	1.79E+03	1.12E+03	1.53E+03	**1.12E+03**
		30	1.63E+03	2.58E+03	1.72E+03	**1.12E+03**

Table 13
Comparison of HV values obtained by the AMDQN and three meta-heuristics.

n	n_i	e_{n_i}	HADE	MOEAD	NSGA-II	**AMDQN**
10	3	10	3.21E−02	2.51E−02	5.16E−02	**1.00E−01**
		20	0.00E+00	1.47E−02	4.42E−02	**4.92E−02**
		30	3.02E−02	0.00E+00	5.26E−02	**6.69E−02**
	5	10	0.00E+00	9.24E−03	**3.70E−02**	3.49E−04
		20	3.21E−02	4.47E−02	1.20E−01	**6.58E−02**
		30	3.15E−02	4.28E−02	4.37E−02	**6.18E−02**
	10	10	0.00E+00	1.22E−02	3.34E−02	**3.76E−02**
		20	1.51E−02	0.00E+00	**3.04E−02**	7.15E−03
		30	0.00E+00	2.42E−02	**3.03E−02**	1.45E−02
20	3	10	3.28E−02	4.14E−02	5.67E−02	**1.46E−01**
		20	2.96E−03	2.62E−03	4.38E−02	**6.27E−02**
		30	3.33E−02	1.86E−02	4.68E−02	**7.37E−02**
	5	10	2.56E−02	0.00E+00	4.72E−02	**6.68E−02**
		20	3.54E−03	4.48E−03	3.01E−02	**1.72E−02**
		30	2.85E−02	3.75E−04	4.92E−02	**5.23E−02**
	10	10	1.17E−02	2.14E−02	**4.06E−02**	2.31E−02
		20	3.07E−02	2.27E−02	4.73E−02	**5.94E−02**
		30	2.68E−02	2.14E−03	5.25E−02	**6.12E−02**
30	3	10	2.59E−02	2.74E−02	4.25E−02	**5.14E−02**
		20	7.02E−03	2.47E−02	4.22E−02	**6.40E−02**
		30	2.82E−02	2.56E−03	4.00E−02	**5.16E−02**
	5	10	2.90E−02	4.96E−02	5.82E−02	**1.07E−01**
		20	2.82E−02	7.70E−03	4.44E−02	**7.34E−02**
		30	3.14E−02	3.08E−02	4.95E−02	**1.04E−01**
	10	10	2.54E−02	1.66E−02	**4.16E−02**	1.22E−02
		20	2.85E−02	4.85E−02	4.41E−02	**7.14E−02**
		30	2.88E−02	1.61E−02	4.75E−02	**7.41E−02**

instances. This may be due to the fact that MOEAD uses weight vectors based on uniform distribution to guide the evolution, but for complex problems such as multi-task real-time scheduling, weight vectors that are uniformly distributed on the unit hyperplane do not guarantee the uniform distributability of the solution. NSGA-II and MOEAD, on the other hand, each achieve second best results at some fixed instances (NSGA-II: n = 10, MOEAD: n = 20), even NSGA-II is superior or equal to the proposed AMDQN algorithm on a few instances. However, the proposed AMDQN achieves optimal results on the remaining 23 instances, verifying its adaptability to different manufacturing environments.

5.3.5. Comparisons with other RL-based scheduling methods

Moreover, in order to verify the superiority of hierarchical structure of the AMDQN in multi-objective optimization, it is also compared with three other RL-based scheduling methods, including classical Q-learning, DQN (DQN), and double DQN (DDQN). Since these methods only use a single DQN to select the scheduling rule without considering the desired optimization objective. For comparison purposes, the manufacturing time and manufacturing cost reward functions (Algorithms 8 and 9 in Section 4.4) are combined using conditional judgments, the pseudocode is shown in Algorithm 10. Each method was repeated for 10 runs on each instance and the comparison results are shown in Tables 14 and 15, with the best results highlighted in bold. The average values of IGD and HV for AMDQN and the other three RL-based scheduling methods for 10 repeated runs on 27 instances are shown in Fig. 14. Fig. 15 illustrates the Pareto-optimal fronts for some representative examples obtained by the AMDQN and other RL-based methods.

Algorithm 10: Combined reward algorithm

Input: $TR_{ave}(t)$, $TR_{ave}(t+1)$, $OS_{ave}(t)$, $OS_{ave}(t+1)$
Output: $r(t)$

```
if TR_ave(t+1) < TR_ave(t) then
    r(t) ← 1;
else
    if TR_ave(t+1) < TR_ave(t) * 1.1 then
        r(t) ← 0;
    else
        if OS_ave(t+1) < OS_ave(t) then
            r(t) ← 1;
        else
            if OS_ave(t+1) < OS_ave(t) * 1.1 then
                r(t) ← 0;
            else
                r(t) ← −1;
            end
        end
    end
end
return r(t);
```

The comparison results from Tables 14 and 15 demonstrate that AMDQN outperforms other RL-based scheduling methods in most cases. It is worth noting that the Q-learning algorithm shows the worst results on almost all instances, due to the fact that the it needs to maintain a Q table, which can only store discrete and limited states and actions. However, most states in real manufacturing systems are continuous, and forcing them to be discretized would reduce the perception accuracy of the agents for the manufacturing environment. Compared to the other two DRL-based methods, AMDQN also achieves the best results on the vast majority of instances. This is due to the fact that a single DQN cannot specify the objective to be optimized, resulting in the inability of the agents to learn a scheduling rule that is optimal for each of the two optimization objectives. This verifies that the hierarchical structure of the AMDQN is more effective and superior.

5.4. Discussion and limitation

In this paper, the effectiveness of the proposed AMDQN is evaluated from both convergence and diversity perspectives. The above experimental results demonstrate that the solution set obtained by AMDQN

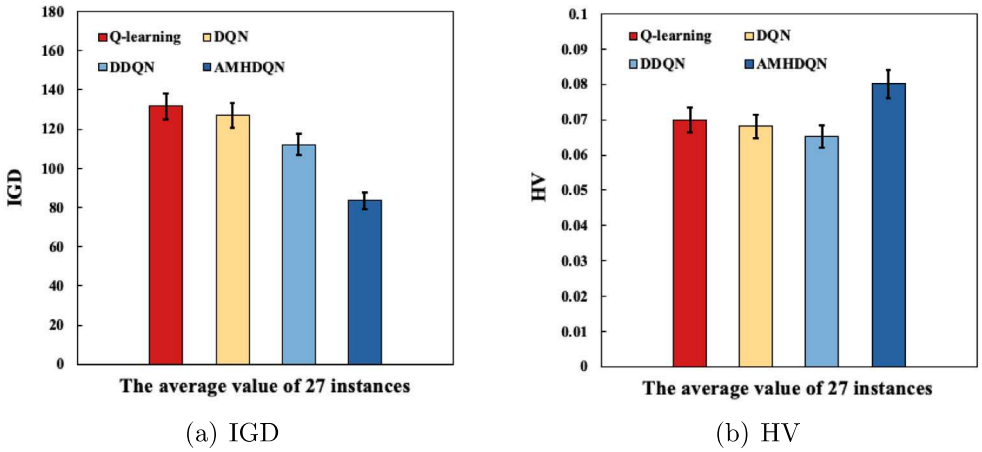


Fig. 14. Average values of IGD and HV obtained by the AMDQN and other RL-based scheduling methods.

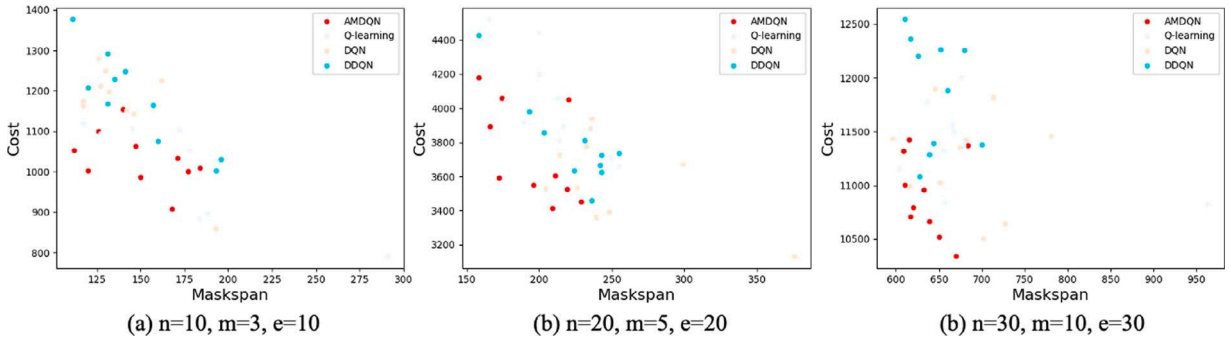


Fig. 15. Pareto-optimal fronts obtained by the AMDQN and RL-based scheduling methods for some representative instances.

Table 14
Comparison of IGD values obtained by the AMDQN and other RL-based scheduling methods.

n	n_i	e_{n_i}	Q-learning	DQN	DDQN	AMDQN
10	3	10	8.23E+01	4.08E+01	3.69E+01	**2.57E+01**
		20	1.38E+02	4.22E+01	1.58E+02	**1.43E+02**
		30	4.41E+01	4.60E+01	3.32E+01	**2.27E+01**
	5	10	7.37E+01	4.70E+01	6.04E+01	2.82E+01
		20	8.66E+01	5.44E+01	5.38E+01	**5.19E+01**
		30	8.80E+01	6.56E+01	7.36E+01	**6.13E+01**
	10	10	5.11E+01	4.93E+01	4.86E+01	**2.29E+01**
		20	1.96E+02	2.06E+02	2.31E+02	**4.21E+01**
		30	1.34E+02	4.64E+01	9.80E+01	**1.25E+02**
20	3	10	1.06E+02	1.17E+02	5.35E+01	**3.77E+01**
		20	5.31E+01	5.17E+01	6.77E+01	**2.65E+01**
		30	6.60E+01	**4.86E+01**	1.12E+02	5.70E+01
	5	10	5.74E+01	5.97E+01	3.28E+01	**1.55E+02**
		20	1.17E+02	1.55E+02	2.04E+02	**3.63E+01**
		30	9.70E+01	5.83E+01	**5.49E+01**	7.02E+01
	10	10	4.50E+01	1.90E+02	1.17E+02	**3.54E+01**
		20	2.58E+02	2.19E+02	2.71E+02	**8.02E+01**
		30	3.38E+02	1.14E+02	**3.11E+01**	1.19E+02
30	3	10	3.20E+01	8.37E+01	7.01E+01	**1.72E+01**
		20	6.00E+01	6.60E+01	**2.32E+01**	5.27E+01
		30	2.93E+01	1.34E+02	9.35E+01	**1.14E+02**
	5	10	1.36E+02	1.22E+02	5.75E+01	**1.15E+02**
		20	1.55E+02	1.46E+02	3.15E+01	**1.13E+02**
		30	1.31E+02	6.20E+01	1.40E+02	**1.34E+02**
	10	10	1.12E+02	1.55E+02	1.47E+02	**1.08E+02**
		20	6.65E+02	7.66E+02	4.85E+02	**2.83E+02**
		30	2.04E+02	2.83E+02	2.43E+02	**1.79E+02**

Table 15
Comparison of HV values obtained by the AMDQN and other RL-based scheduling methods.

n	n_i	e_{n_i}	Q-learning	DQN	DDQN	AMDQN
10	3	10	7.14E−02	7.66E−02	6.53E−02	**9.83E−02**
		20	1.49E−01	1.48E−01	1.28E−01	**1.73E−01**
		30	1.24E−01	1.11E−01	**1.31E−01**	1.29E−01
	5	10	1.01E−01	9.71E−02	8.14E−02	**1.01E−01**
		20	8.54E−02	7.40E−02	7.62E−02	**9.02E−02**
		30	4.49E−02	4.92E−02	**5.23E−02**	4.89E−02
	10	10	4.52E−02	4.56E−02	3.50E−02	**4.74E−02**
		20	6.24E−02	6.66E−02	5.72E−02	**9.78E−02**
		30	6.85E−02	8.15E−02	7.27E−02	**8.44E−02**
20	3	10	4.03E−02	3.52E−02	5.00E−02	**5.37E−02**
		20	7.15E−02	6.31E−02	5.45E−02	**7.46E−02**
		30	2.17E−01	**2.26E−01**	1.98E−01	2.14E−01
	5	10	3.54E−02	3.34E−02	3.97E−02	**5.08E−02**
		20	4.12E−02	2.95E−02	7.56E−03	**5.13E−02**
		30	3.73E−02	4.74E−02	3.73E−02	**5.11E−02**
	10	10	4.87E−02	3.61E−02	4.66E−02	**5.54E−02**
		20	9.20E−02	**1.07E−01**	8.86E−02	1.05E−01
		30	4.65E−02	4.65E−02	4.42E−02	**5.38E−02**
30	3	10	7.19E−02	6.28E−02	6.56E−02	**7.47E−02**
		20	5.33E−02	4.61E−02	**5.80E−02**	5.34E−02
		30	7.53E−02	7.39E−02	6.20E−02	**8.55E−02**
	5	10	2.63E−02	2.97E−02	**3.76E−02**	3.17E−02
		20	2.60E−02	1.28E−02	8.80E−03	**2.60E−02**
		30	9.23E−02	8.39E−02	8.39E−02	**1.01E−01**
	10	10	5.42E−02	5.10E−02	5.33E−02	**6.15E−02**
		20	6.21E−02	6.19E−02	8.16E−02	**9.98E−02**
		30	4.39E−02	4.42E−02	4.49E−02	**5.35E−02**

is comprehensively optimal among all the compared algorithms. First, compared to the other three RL-based scheduling methods, AMDQN is optimized by 16% and 13% over the second best DDQN, in terms of the mean values of IGD and HV, respectively. This is due to the two-hierarchy deep reinforcement learning framework of AMDQN which makes hierarchical decisions on the selection of the optimization objective and the optimal scheduling strategy respectively, thus achieving a better compromise between multiple optimization objectives. Secondly, AMDQN shows better results on almost all instances compared to the five classical scheduling rules and two scheduling rules in recent research. This is due to the fact that in this paper, nine composite scheduling rules are designed to adapt to different manufacturing environment states from two levels, namely, manufacturing tasks and manufacturing resources, as well as multiple dimensions, such as urgency and completion rate of tasks and utilization and cost of manufacturing resources. Finally, AMDQN achieves better performance on all instances compared to the proposed nine composite scheduling rules. This is made possible by the introduction of two metrics, estimated tardiness rate and estimated overspend rate, in the two designed reward algorithms, which guide the A-DQN to learn and adjust the scheduling rules according to the changing state of the manufacturing environment.

This paper proposes a new architecture and a new methodology for resolving conflicts and solving adaptive multi-task and multi-objective scheduling with resource competition and conflict among tasks. In this paper, state features containing three dimensions and six indicators are extracted. The coupling relationship of indicators within the same dimension may be misleading to the network, but it also provides a more accurate representation of the state of the manufacturing environment. Second, for external events such as urgent task insertion and requirement changes, this paper uses to include them into the next time window for processing.

6. Conclusion

In order to be closer to the actual manufacturing overall scheduling requirements of enterprise alliances, this paper proposes an AMMS-RCCT model for enterprise alliance value nets to simulate multi-task scheduling in enterprise alliances that contain resource competition and conflict among tasks. First, a hybrid optimization scheduling strategy of "parallel+serial" is adopted, where serial and parallel methods are used for single and multiple candidate manufacturing cells respectively to achieve the shortest possible total manufacturing time while resolving conflicts. Second, an AMDQN based on a two-hierarchy deep reinforcement learning architecture is proposed, which employs two independent agents (C-DQN and A-DQN) for hierarchical decision-making to solve the AMMS-RCCT model. This reduces the complexity of scheduling rule optimization while achieving multi-objective optimization compromise by decoupling the selection of optimization objective and the selection of the optimal scheduling rule. AMDQN uses a combination of front C-DQN and back A-DQN to adaptively select the appropriate optimization objective and its corresponding optimal scheduling rule according to the environment state at each scheduling point, and achieve a good compromise between the total manufacturing time and the total manufacturing cost while multi-task adaptive optimal scheduling. Based on this, two different reward algorithms are designed by introducing two metrics, the estimated trainees rate and the estimated overspend rate, and three different levels of reward values to guide the A-DQN to learn and adjust the scheduling rules according to the manufacturing state changes. Moreover, three scheduling rules aiming at reducing total manufacturing time and total manufacturing cost are designed from the manufacturing tasks and manufacturing resources, respectively. Nine composite scheduling rules are obtained by combining them to adapt to different manufacturing environment conditions and optimize the enterprise alliance value net as a whole.

Finally, this paper designs 27 instances with different numbers of tasks, subtasks, and manufacturing cells to simulate real manufacturing scenarios. AMDQN is experimentally compared with the proposed nine scheduling rules, five classical scheduling rules, scheduling rules in existing research and three other RL-based scheduling methods from both convergence and diversity perspectives. The experimental results demonstrate that the scheduling effectiveness of AMDQN is significantly better than other methods on the vast majority of instances. The average IGD and HV values of AMDQN are optimized by 16% and 13%, respectively, compared to the second best DRL-based algorithm DDQN.

In future research, methods such as feature selection will be used to optimize the state features of the manufacturing environment so as to improve the effectiveness of deep reinforcement learning networks in solving scheduling problems. Besides, real-time processing methods for manufacturing process uncertainties represented by external events will be investigated. Further consideration is given to more other objectives, including the utilization of manufacturing resources of the dynamic alliance of firms over a certain period of time, reliability and service quality of manufacturing resources.

CRediT authorship contribution statement

Jianxiong Zhang: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Bing Guo:** Writing – review & editing, Supervision. **Xuefeng Ding:** Writing – review & editing. **Dasha Hu:** Writing – review & editing. **Jun Tang:** Validation, Formal analysis, Data curation. **Ke Du:** Validation, Data curation. **Chao Tang:** Validation, Data curation. **Yuming Jiang:** Writing – review & editing, 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.

Data availability

Code metadata section with link to Code Ocean are available.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant No. 2020YFB1707900, 2023YFB3308300 and 2020YFB1711800; the National Natural Science Foundation of China under Grant No. 62262074, U2268204 and 62172061; the Science and Technology Project of Sichuan Province under Grant No. 2022YFG0159, 2022YFG0155 and 2022YFG0157.

References

[1] Y. Li, F. Tao, Y. Cheng, X. Zhang, A. Nee, Complex networks in advanced manufacturing systems, J. Manuf. Syst. 43 (2017) 409–421.
[2] Q. Zhang, M.F. Zhani, R. Boutaba, J.L. Hellerstein, Dynamic heterogeneity-aware resource provisioning in the cloud, IEEE Trans. Cloud Comput. 2 (1) (2014) 14–28.
[3] H. Yan, Q. Hua, Y. Wang, W. Wei, M. Imran, Cloud robotics in smart manufacturing environments: Challenges and countermeasures, Comput. Electr. Eng. 63 (2017) 56–65.
[4] W. Li, C. Zhu, L.T. Yang, L. Shu, E.C.-H. Ngai, Y. Ma, Subtask scheduling for distributed robots in cloud manufacturing, IEEE Syst. J. 11 (2) (2015) 941–950.
[5] Z. Cheng, D. Zhan, X. Zhao, H. Wan, et al., Multitask oriented virtual resource integration and optimal scheduling in cloud manufacturing, J. Appl. Math. 2014 (2014).
[6] F. Li, L. Zhang, Y. Liu, Y. Laili, F. Tao, A clustering network-based approach to service composition in cloud manufacturing, Int. J. Comput. Integr. Manuf. 30 (12) (2017) 1331–1342.

[7] H. Jiang, J. Yi, S. Chen, X. Zhu, A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly, J. Manuf. Syst. 41 (2016) 239–255.

[8] Y. Zhang, D. Xi, H. Yang, F. Tao, Z. Wang, Cloud manufacturing based service encapsulation and optimal configuration method for injection molding machine, J. Intell. Manuf. 30 (2019) 2681–2699.

[9] L. TAI, et al., Multi-objective dynamic scheduling of manufacturing resource to cloud manufacturing services, China Mech. Eng. 24 (12) (2013) 1616.

[10] C. Jian, Y. Wang, Batch task scheduling-oriented optimization modelling and simulation in cloud manufacturing, Int. J. Simul. Model. 13 (1) (2014) 93–101.

[11] Z. Wang, J. Zhang, J. Si, Application of particle swarm optimization with stochastic inertia weight strategy to resources scheduling and assignment problem in cloud manufacturing environment, in: Proceedings of the 33rd Chinese Control Conference, IEEE, 2014, pp. 7567–7572.

[12] M. Yuan, K. Deng, W.A. Chaovalitwongse, S. Cheng, Multi-objective optimal scheduling of reconfigurable assembly line for cloud manufacturing, Optim. Methods Softw. 32 (3) (2017) 581–593.

[13] Z. Wang, J. Zhang, Y. Qi, Job shop scheduling method with idle time in cloud manufacturing, Control Decis. 32 (5) (2017) 811–816.

[14] P. Helo, D. Phuong, Y. Hao, Cloud manufacturing–scheduling as a service for sheet metal manufacturing, Comput. Oper. Res. 110 (2019) 208–219.

[15] Y. Wang, S. Wang, B. Yang, B. Gao, S. Wang, An effective adaptive adjustment method for service composition exception handling in cloud manufacturing, J. Intell. Manuf. (2022) 1–17.

[16] J. Zhou, X. Yao, Hybrid teaching–learning-based optimization of correlation-aware service composition in cloud manufacturing, Int. J. Adv. Manuf. Technol. 91 (2017) 3515–3533.

[17] T. Dong, F. Xue, C. Xiao, J. Li, Task scheduling based on deep reinforcement learning in a cloud manufacturing environment, Concurr. Comput.: Pract. Exper. 32 (11) (2020) e5654.

[18] Z. Chen, L. Zhang, X. Wang, K. Wang, Cloud–edge collaboration task scheduling in cloud manufacturing: An attention-based deep reinforcement learning approach, Comput. Ind. Eng. 177 (2023) 109053.

[19] Y. Ping, Y. Liu, L. Zhang, L. Wang, X. Xu, Sequence generation for multi-task scheduling in cloud manufacturing with deep reinforcement learning, J. Manuf. Syst. 67 (2023) 315–337.

[20] I.-B. Park, J. Huh, J. Kim, J. Park, A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities, IEEE Trans. Autom. Sci. Eng. 17 (3) (2019) 1420–1431.

[21] Y. Zhang, H. Zhu, D. Tang, T. Zhou, Y. Gui, Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems, Robot. Comput.-Integr. Manuf. 78 (2022) 102412.

[22] H. Akbaripour, M. Houshmand, T. Van Woensel, N. Mutlu, Cloud manufacturing service selection optimization and scheduling with transportation considerations: mixed-integer programming models, Int. J. Adv. Manuf. Technol. 95 (2018) 43–70.

[23] J. Chen, G.Q. Huang, J.-Q. Wang, C. Yang, A cooperative approach to service booking and scheduling in cloud manufacturing, European J. Oper. Res. 273 (3) (2019) 861–873.

[24] Y. Liu, X. Xu, L. Zhang, L. Wang, R.Y. Zhong, Workload-based multi-task scheduling in cloud manufacturing, Robot. Comput.-Integr. Manuf. 45 (2017) 3–20.

[25] Y. Wang, S. Gao, S. Wang, R. Zimmermann, An adaptive multiobjective multitask service composition approach considering practical constraints in fog manufacturing, IEEE Trans. Ind. Inform. 18 (10) (2021) 6756–6766.

[26] Y.-k. Liu, X.-s. Zhang, L. Zhang, F. Tao, L.-h. Wang, A multi-agent architecture for scheduling in platform-based smart manufacturing systems, Front. Inf. Technol. Electron. Eng. 20 (11) (2019) 1465–1492.

[27] L. Zhou, L. Zhang, A dynamic task scheduling method based on simulation in cloud manufacturing, in: Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems: 16th Asia Simulation Conference and SCS Autumn Simulation Multi-Conference, AsiaSim/SCS AutumnSim 2016, Beijing, China, October 8-11, 2016, Proceedings, Part III 16, Springer, 2016, pp. 20–24.

[28] Y. Hu, F. Zhu, L. Zhang, Y. Lui, Z. Wang, Scheduling of manufacturers based on chaos optimization algorithm in cloud manufacturing, Robot. Comput.-Integr. Manuf. 58 (2019) 13–20.

[29] F. Li, L. Zhang, T.W. Liao, Y. Liu, Multi-objective optimisation of multi-task scheduling in cloud manufacturing, Int. J. Prod. Res. 57 (12) (2019) 3847–3863.

[30] S. Liu, L. Zhang, W. Zhang, W. Shen, Game theory based multi-task scheduling of decentralized 3D printing services in cloud manufacturing, Neurocomputing 446 (2021) 74–85.

[31] Y. Laili, S. Lin, D. Tang, Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment, Robot. Comput.-Integr. Manuf. 61 (2020) 101850.

[32] B. Huang, C. Li, F. Tao, A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system, Enterp. Inf. Syst. 8 (4) (2014) 445–463.

[33] H. Liang, X. Wen, Y. Liu, H. Zhang, L. Zhang, L. Wang, Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning, Robot. Comput.-Integr. Manuf. 67 (2021) 101991.

[34] Y. Liu, L. Zhang, L. Wang, Y. Xiao, X. Xu, M. Wang, A framework for scheduling in cloud manufacturing with deep reinforcement learning, in: 2019 IEEE 17th International Conference on Industrial Informatics, INDIN, Vol. 1, IEEE, 2019, pp. 1775–1780.

[35] L. Zhou, L. Zhang, B.K. Horn, Deep reinforcement learning-based dynamic scheduling in smart manufacturing, Procedia Cirp 93 (2020) 383–388.

[36] H. Zhu, M. Li, Y. Tang, Y. Sun, A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing, IEEE Access 8 (2020) 9987–9997.

[37] S. Yang, Z. Xu, Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing, Int. J. Prod. Res. 60 (16) (2022) 4936–4953.

[38] H. Du, W. Xu, B. Yao, Z. Zhou, Y. Hu, Collaborative optimization of service scheduling for industrial cloud robotics based on knowledge sharing, Procedia CIRP 83 (2019) 132–138.

[39] Z. Yin, J. Liu, D. Wang, Multi-AGV task allocation with attention based on deep reinforcement learning, Int. J. Pattern Recognit. Artif. Intell. 36 (09) (2022) 2252015.

[40] T. Dong, F. Xue, C. Xiao, J. Zhang, Workflow scheduling based on deep reinforcement learning in the cloud environment, J. Ambient Intell. Humaniz. Comput. (2021) 1–13.

[41] T. Dong, F. Xue, C. Xiao, J. Li, Task scheduling based on deep reinforcement learning in a cloud manufacturing environment, Concurr. Comput.: Pract. Exper. 32 (11) (2020) e5654.

[42] Y. Wei, D. Kudenko, S. Liu, L. Pan, L. Wu, X. Meng, A reinforcement learning based workflow application scheduling approach in dynamic cloud environment, in: Collaborative Computing: Networking, Applications and Worksharing: 13th International Conference, CollaborateCom 2017, Edinburgh, UK, December 11–13, 2017, Proceedings 13, Springer, 2018, pp. 120–131.

[43] Y. Liu, Y. Ping, L. Zhang, L. Wang, X. Xu, Scheduling of decentralized robot services in cloud manufacturing with deep reinforcement learning, Robot. Comput.-Integr. Manuf. 80 (2023) 102454.

[44] X. Wang, L. Zhang, Y. Liu, F. Li, Z. Chen, C. Zhao, T. Bai, Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning, J. Manuf. Syst. 65 (2022) 130–145.

[45] J. Tang, K. Salonitis, A deep reinforcement learning based scheduling policy for reconfigurable manufacturing systems, Procedia CIRP 103 (2021) 1–7.

[46] H. Wang, X. Wang, X. Hu, X. Zhang, M. Gu, A multi-agent reinforcement learning approach to dynamic service composition, Inform. Sci. 363 (2016) 96–119.

[47] C. Hofmann, C. Krahe, N. Stricker, G. Lanza, Autonomous production control for matrix production based on deep Q-learning, Procedia CIRP 88 (2020) 25–30.

[48] S. Luo, L. Zhang, Y. Fan, Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning, Comput. Ind. Eng. 159 (2021) 107489.

[49] L. Zhang, C. Yang, Y. Yan, Y. Hu, Distributed real-time scheduling in cloud manufacturing by deep reinforcement learning, IEEE Trans. Ind. Inform. 18 (12) (2022) 8999–9007.

[50] X. Wang, L. Zhang, Y. Liu, C. Zhao, K. Wang, Solving task scheduling problems in cloud manufacturing via attention mechanism and deep reinforcement learning, J. Manuf. Syst. 65 (2022) 452–468.

[51] 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.

[52] G.-G. Wang, D. Gao, W. Pedrycz, Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm, IEEE Trans. Ind. Inform. 18 (12) (2022) 8519–8528.

[53] C.-C. Chiu, C.-M. Lai, Multi-objective missile boat scheduling problem using an integrated approach of NSGA-II, MOEAD, and data envelopment analysis, Appl. Soft Comput. 127 (2022) 109353.

[54] I. Rahimi, A.H. Gandomi, K. Deb, F. Chen, M.R. Nikoo, Scheduling by NSGA-II: Review and bibliometric analysis, Processes 10 (1) (2022) 98.

[55] K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, L. Tang, A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem, Expert Syst. Appl. 205 (2022) 117796.