Engineering Applications of Artificial Intelligence 164 (2026) 113352

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Research paper

# A knowledge region selection enhanced quality-diversity algorithm for real-world flexible job shop scheduling with Automated Guided Vehicles transportation

Haoxiang Qin [a], Yi Xiang [a,*], Yuyan Han [b], Yuting Wang [b], Junqing Li [c], Quanke Pan [d]

[a] School of Software Engineering, South China University of Technology, Guangzhou 510006, China
[b] School of Computer Science, Liaocheng University, Liaocheng 252059, China
[c] Department of Mathematics, Yunnan Normal University, Kunming 650500, China
[d] School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China

ARTICLE INFO

*Keywords:*
Quality-diversity
Q-learning
Flexible job-shop scheduling
Automated Guided Vehicle

ABSTRACT

The Flexible Job Shop Scheduling problem with Automated Guided Vehicles (FJSP-AGVs) is commonly encountered in modern manufacturing systems, with complexity increasing exponentially as the problem scales, making traditional methods inadequate. So far, a limitation of meta-heuristic algorithms incorporating machine learning for solving FJSP-AGVs is that machine learning is primarily used for heuristic selection (e.g., dispatching rules or local search operators) rather than actively guiding solution evolution based on the diverse features of solutions, leading to incomplete exploration and premature convergence. To tackle these issues, this paper first presents a Q-Learning Region Selection Enhanced Quality-Diversity (Q-QD) algorithm, which integrates a pairwise selection scheme to identify potential collaborative solutions within the feature space for two practical Chinese manufacturing scenarios: a coal machinery structural parts production workshop and a machining workshop in Nanjing. Unlike traditional approaches that focus solely on optimizing the objective, Q-QD leverages the customized feature space incorporating behaviors such as job transportation times and machine idle times. By combining Q-learning-based region selection with heuristic rule-based local search for AGV transportation, the proposed method effectively discovers high-performing and diverse solutions. The results obtained from two real-world cases and 21 benchmark instances indicate that the proposed Q-QD algorithm surpasses five advanced counterparts, achieving a makespan reduction of 12.2–30.8% relative to the strongest baseline. These substantial performance gains highlight the robustness and efficiency of QD-based optimization for solving FJSP-AGVs and suggest a valuable avenue for future research in intelligent job shop scheduling.

## 1. Introduction

The flexible job shop scheduling problem (FJSP) is extensively present in discrete manufacturing industries such as machinery and chemical engineering (Yuan and Xu, 2015; Li and Gao, 2016; Chen et al., 2025). In FJSP, each operation can be flexibly processed on multiple machines (Li et al., 2022a). However, traditional FJSP ignores the issue of material transportation between machines, or simply introduces time delays, while assuming an unlimited number of transportation resources (Xu et al., 2023). It is obviously not consistent with actual production.

In practice, transportation has become an increasingly critical factor in manufacturing systems, especially in the post-pandemic era (Abbasi et al., 2025). Global disruptions in logistics networks, resource shortages, and rising transportation costs have highlighted the vulnerability of production systems that rely on timely material transfer. These issues emphasize the need for smarter and resource-aware scheduling mechanisms that can adapt to limited and fluctuating transport capacity. As a commonly used transportation resource in shop scheduling, Automated Guided Vehicles (AGVs) are assigned the task of transporting products between machines. Variations in transportation time, caused by different AGV selections and transportation routes, directly influence the start and completion time of the operations (Pan et al., 2022b). This transportation time cannot be simply replaced by machine setup time or maintenance. However, due to limited AGV resources, many products

* Corresponding author.
*E-mail addresses:* 987352978@qq.com (H. Qin), xiangyi@scut.edu.cn (Y. Xiang), hanyuyan@lcu-cs.com (Y. Han), wangyuting@lcu-cs.com (Y. Wang), lijunqing@ynnu.edu.cn (J. Li), panquanke@shu.edu.cn (Q. Pan).

https://doi.org/10.1016/j.engappai.2025.113352
Received 17 March 2025; Received in revised form 26 October 2025; Accepted 24 November 2025

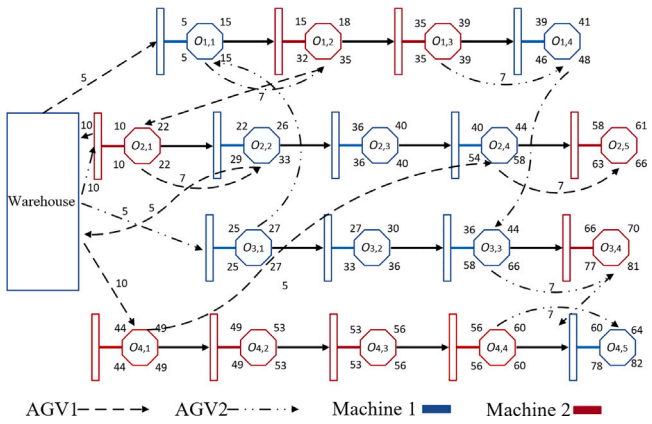



**Fig. 1.** The schedule illustration of the FJSP-AGVs.

are unable to be processed on the next machine without any delay, thereby disrupting the original scheduling scheme (Pan et al., 2024). Therefore, it is essential to design efficient scheduling optimization methods for FJSP that incorporate limited AGV resources (FJSP-AGVs). A brief illustration of a FJSP-AGVs scheduling graph is shown in Fig. 1. A notched square is used to represent an operation $O_{i,j}$. The upper pair indicates the planned start and completion time, while the lower pair indicates the actual start and completion time, respectively.

Various approaches have been proposed to address the FJSP and FJSP-AGVs. The most popular approaches are mathematical/exact methods (Meng et al., 2020; Yao et al., 2024; Li et al., 2024b) and meta-heuristic algorithms (Dauzère-Pérès et al., 2023; Wang et al., 2022; Lin et al., 2022; Lei et al., 2019; Ding et al., 2019; Zhang and Wang, 2018; Cao et al., 2021; An et al., 2023; Li et al., 2021; Pan et al., 2022a; Luo et al., 2020; Du et al., 2021; Zhang et al., 2023; Marzouki et al., 2022; Li et al., 2023a; Du et al., 2023; Meng et al., 2023). Exact methods are adept at finding optimal solutions for small-scale problems, but they struggle to identify the best solutions within a reasonable time or even fail to find a solution for large-scale problems (He et al., 2024; Huang et al., 2024). Meta-heuristic algorithms can efficiently solve large-scale FJSP-AGVs in finite time (Gao et al., 2019). So far, there have been a number of studies integrating meta-heuristic algorithms with machine learning techniques, demonstrating the superiority of their effectiveness. For example, Zhang et al. (2023) proposed a Q-Learning method to assist the evolutionary algorithm in selecting heuristics. Qin et al. (2023) utilized the Q-learning to help the iterative greedy (IG) algorithm choose local search operators. A hybrid deep Q-network (DQN) was designed to decide the utilization of the dispatching rules (Li et al., 2022b). In Zhang et al. (2024), a DQN was utilized to select local search operators for the memetic algorithm. In addition, recent advances in machine learning have shown great potential across a wide range of application domains, such as intelligent decision-making (Alazaidah et al., 2024), medical diagnosis (Elhag, 2024; Padre et al., 2025), energy and financial forecasting (Hamid et al., 2025; Aripov et al., 2025), and software testing (Girgis et al., 2025). These studies further highlight the rapid development of machine learning techniques and their adaptability to complex data-driven problems, providing inspiration for the integration into scheduling optimization.

From the above studies, it is evident that machine learning techniques can help meta-heuristic algorithms to intelligently choose heuristics, dispatching rules, or local search operators (Du et al., 2024). A common limitation among these approaches lies in their focus on selection strategies alone. Although such strategies enhance search efficiency, they often lack effective guidance for directing the evolutionary process (Mouret and Clune, 2015; Su et al., 2023). This leads to the first challenge: is it possible to assist in the selection of promising solutions by using machine learning techniques, thereby prompting solutions to evolve more efficiently and diversely towards the optimal solution? Furthermore, the search strategies presented in the above studies are mostly based on sorting perturbations or several heuristic rules. They do not take into account the problem characteristics, i.e., the impact of AGV transportation and machine idleness on the total sequence completion time. Evidence shows that studying the characteristics helps greatly in problem solving (Chatzilygeroudis et al., 2020; Cully and Demiris, 2018). This leads to the second challenge: can local search be further improved by integrating FJSP-AGVs domain knowledge, or by searching for high-performing and more diverse solutions based on the intrinsic features of the problems?

Motivated by the aforementioned challenges, we design a Q-Learning-driven Quality-Diversity (Q-QD) approach for FJSP-AGVs. Unlike conventional EAs—such as global optimization and multimodal approaches that focus solely on optimizing the objective function (Chatzilygeroudis et al., 2020), QD aims to generate a diverse set of high-performing solutions in a customized feature space (Pugh et al., 2016). It achieves this by maintaining a structured archive of solutions, where each entry represents a distinct behavioral characteristic while still striving for high quality. This fundamental difference allows QD to be particularly effective in handling diverse requirements, novel designs, or multiple behavior solutions. Furthermore, a particularly important point is that QDs have been theoretically proven to better avoid local optima and are, in most cases, superior to traditional EAs (Qian et al., 2024). This is also the main reason for choosing the QD framework rather than other EAs in this paper. So far, QD has achieved a handful of achievements in the fields of robotics (Cully et al., 2015; Grillotti and Cully, 2022), games (Ecoffet et al., 2021; Alvarez et al., 2022), neural networks (Colas et al., 2020; Flageat et al., 2023), and software testing (Xiang et al., 2022a,b).

A crucial consideration in designing Q-QD is selecting an appropriate machine learning technique to guide feature-space exploration. While deep reinforcement learning (DRL) methods such as DQN and Policy Gradient have shown potential in scheduling and combinatorial optimization (Zhang et al., 2024; Li et al., 2022b), they typically require large datasets, converge slowly, and generalize poorly to diverse scheduling problems. In contrast, tabular Q-Learning offers clear advantages: (i) it is model-free, requiring no prior knowledge of the environment; (ii) it efficiently learns from experience, making it suitable for adaptive decision-making; and (iii) it provides explicit, interpretable policies for flexible solution selection. Since feature-space exploration of QD focuses on adaptive region selection rather than high-dimensional state learning, Q-Learning serves as a computationally efficient and conceptually simple mechanism to balance exploration and exploitation in the Q-QD framework.

Building on these advantages, the proposed Q-QD algorithm adopts Q-Learning as a mechanism for adaptive region selection in the transportation-aware feature space, rather than for operator-level decision as in most RL-enhanced optimization methods. The tabular Q-Learning agent selects the region that has potential solutions, and the search operators act within that region. This region-level design yields three benefits: (i) sample-efficient learning in a low-dimensional grid setting, (ii) reliable reward feedback at the region level that remains stable, and (iii) explicit and interpretable policies that expose the trade-off between transportation frequency and machine idleness. The reward combines makespan improvement with archive coverage/diversity increments, thus encouraging the agent to prioritize regions that are both promising and under-explored. This contrasts with classical EAs (quality-only) and prior RL for operator selection, and directly targets the QD principle of high-quality and diverse solutions. Additionally, based on the region selection, we develop a pairwise selection scheme for generating potential collaborative solutions and design the AGV/critical-path-guided local search to perturb solutions, reducing ineffective moves and improving exploitation depth.

**Table 1**
The gap between the proposed approach and existing research.

| Research Gap | Limitations in Existing Studies | Proposed Approach |
|---|---|---|
| **Lack of Problem-Specific Exploration in ML-Assisted Metaheuristics** | ML is mainly used for heuristic selection (e.g., local search operators, dispatching rules) rather than actively guiding solution evolution (Qin et al., 2023; Li et al., 2022b; Zhang et al., 2024; Du et al., 2024). | Q-Learning Region Selection is introduced to dynamically guide QD toward promising feature-space regions, improving both efficiency and diversity. |
| **Inefficient Use of AGV Transportation Knowledge** | Most existing search strategies do not fully consider AGV scheduling rules, leading to suboptimal makespan (Yao et al., 2024; Li et al., 2024b). | A Critical Path-Based Local Search is designed to incorporate AGV transportation heuristics, effectively reducing idle time. |
| **Basic Selection Strategies in Existing QD Applications** | Prior scheduling studies using QD rely on random selection or simple perturbation without adaptive region-based exploration (Urquhart et al., 2020; Qin et al., 2024, 2025). | The proposed Pairwise Selection Scheme enhances collaborative search by refining the solution selection process. |
| **Limited QD Applications in Discrete Scheduling** | Most QD applications focus on continuous optimization (e.g., robotics, neural networks, games) (Cully et al., 2015; Grillotti and Cully, 2022; Ecoffet et al., 2021; Alvarez et al., 2022; Colas et al., 2020; Flageat et al., 2023), making direct adaptation to FJSP-AGVs infeasible (Qin et al., 2025). | The proposed QD framework is tailored for discrete scheduling, incorporating problem-specific heuristics to improve exploitation. |
| **Lack of Empirical Validation of Feature-Space Division and Selection Impact** | Few studies evaluate the effect of feature-space partitioning, solution selection bias, and QD behavior in scheduling problems (Cully and Demiris, 2018; Chatzilygeroudis et al., 2020). | A comprehensive experimental study examines feature-space division, evaluates the impact of different selection strategies, and explores the effectiveness of QD in discovering high-quality solutions. |

Comprehensive experiments on region division, region selection, pairwise selection, and evolutionary bias confirm that Q-QD achieves a 12.2–30.8% reduction in makespan over the best-performing existing algorithms. To better illustrate the differences, Table 1 further illustrates the limitations of prior studies and the advantages of our proposed approach.

The main contributions can be summarized as follows:

- The Q-Learning Region Selection Mechanism is developed for practical FJSP-AGVs cases to help QD optimization intelligently select potential regions for collaborative solutions in the feature space. The pairwise selection scheme further refines this process, and results show its superiority over randomization and mainstream selection methods.
- By integrating the domain knowledge of FJSP-AGVs, this paper proposes a critical path-based Local Search with AGV transportation heuristic rules. The neighborhood is defined by modifying one or two positions in the encoding vectors, and its effectiveness is evaluated against mainstream search methods. The proposed Local Search enhances QD by discovering high-performing and diverse solutions with varying behavioral feature values in the feature space.
- It is observed that Q-Learning primarily selects solutions from feature regions with fewer AGV transports, and the high-quality solutions found often align with these collaborative regions (see Section 5 for details).

The remainder of this work is shown as follows. Section 2 reviews related work of FJSPs and their variants with transportation constraints, machine learning methods, and QD algorithms. Section 3 illustrates the formulation of FJSP. In Section 4, the Q-QD is described in detail. In Section 5, this work conducts detailed experiments and analyzes the results. Then, we summarize this work in Section 6 and give an overview of possible future research directions.

## 2. Related work

Building upon the challenges and motivations discussed in the Introduction, this section reviews prior studies related to the FJSP and its extensions with transportation constraints, as well as the integration of machine learning and QD algorithms.

### 2.1. FJSP and its extension with transportation constraints

The FJSP has drawn extensive research interest from both academia and industry due to its wide range of real-world applications (Yuan and Xu, 2015; Wang et al., 2022; Lin et al., 2022). Nevertheless, for large-scale and highly interdependent FJSP instances, it remains extremely difficult for mathematical or exact optimization techniques to generate satisfactory solutions within a reasonable computational time (Meng et al., 2020; Yao et al., 2024; Li et al., 2024b). Consequently, numerous metaheuristic algorithms have been developed for smaller populations, such as the single-individual IG algorithm (Li et al., 2022a) and the master–apprentice evolutionary framework involving two individuals (Ding et al., 2019), both demonstrating competitive performance in FJSP optimization. Beyond these, many extended variants of the FJSP have been explored. For example, Zhang and Wang (2018) examined an assembly-constrained version of the problem, while Cao et al. (2021) introduced a knowledge-driven Cuckoo Search algorithm incorporating mating operations. To cope with distributed manufacturing systems and dynamic job arrivals, Huang et al. (2024) applied a Graph Neural Network (GNN) to extract intrinsic scheduling features. Moreover, An et al. (2023) presented a NSGA-III-based approach to handle five simultaneous optimization objectives in the FJSP. Although these studies proposed effective scheduling algorithms for different FJSP variants, none of them explicitly addressed transportation constraints within the scheduling process.

Up to now, research integrating FJSP with AGV transportation has been relatively scarce, particularly in scenarios where the interaction between scheduling decisions and AGV routing could be fully exploited. In Dai et al. (2019), an enhanced Genetic Algorithm (GA) was designed to jointly optimize makespan and energy usage under transportation limitations. Likewise, Li and Lei (2021) introduced a feedback-enhanced imperialist competitive method that simultaneously optimizes energy consumption, makespan, and total tardiness. From the perspective of green scheduling, Xu et al. (2023) developed an efficient heuristic approach to handle FJSP-AGV. Furthermore, studies employing both single- and dual-population GAs Li et al. (2024b), Han et al. (2024) have achieved comparable effectiveness in solving FJSP-AGVs. However, despite the success of these metaheuristic techniques, few have leveraged machine learning strategies to enhance algorithmic adaptability or incorporated AGV transportation knowledge explicitly within the optimization process.

### 2.2. Machine learning for FJSP-AGVs

Several studies have utilized reinforcement learning or deep reinforcement learning techniques to assist meta-heuristic algorithms in

obtaining good solutions. In Zhang et al. (2023), Chen et al. (2024), Q-Learning methods were utilized to choose heuristics for the method. Hu et al. (2020) proposed a DQN that selects the suitable dispatching rules and AGVs. In some other studies using DQNs to assist algorithms in solving FJSP-AGVs, the function of most DQNs is still to select operators and dispatch rules for the algorithms through learning (Li et al., 2022b; Zhang et al., 2024; Yuan et al., 2023). Upon investigation, machine learning methods used in a several related studies have similar selection functions (Qin et al., 2023; Tao et al., 2024; Li et al., 2023b). Inspired by these works, we propose the conjecture: is it possible to use machine learning methods for assisting algorithms in learning the evolutionary direction of solutions? We argue that perhaps changing the original operator selection or dispatching rules to the selection of regions, and thereby selecting potential collaborative solutions within that region, to provide more favorable knowledge for overall solution improvement. This might be more helpful for algorithm performance improvement. According to the above conjecture, this work applied Q-Learning for the Q-QD to intelligently choose regions within the feature space, and then select promising solutions. In the experimental chapter, we carry out a number of experiments to prove the validity of the conjecture.

### 2.3. Applications of quality-diversity algorithms

QD has achieved good results in various areas. In robotics, QD created a map containing diverse solutions for the robot. In the case of a damaged robot, the method guided the robot's learning process through trial and error so that the robot could continue to work effectively. This research has been successfully published in the renowned journal *Nature* (Cully et al., 2015). In other robotics research, QD autonomously learned how to define appropriate descriptors based on raw sensory data collected by robots (Grillotti and Cully, 2022). In games, QD algorithms were designed to generate different game scenarios and contents (Alvarez et al., 2022). In addition, QD had been used to solve 11 unsolved games in the Atari suite (Ecoffet et al., 2021). The article was also published in *Nature*. In neural networks (Colas et al., 2020), the combination of MAP-Elites and evolutionary strategies solved the problem of post-injury recovery in high-dimensional control tasks that cannot be accomplished by traditional methods. In software testing, QD was used to generate test cases applied to a software product line (Xiang et al., 2023). So far, only a few QDs have been applied to solve discrete combinatorial optimization issues (particularly in the area of scheduling). For example, in workforce scheduling and routing problems, QD was used to efficiently schedule and dispatch employees to accomplish special travel tasks (Urquhart and Hart, 2018; Urquhart et al., 2019, 2020). They needed to ensure a reasonable allocation of tasks while also optimizing the travel routes of employees. Apart from this, recent studies have applied QD optimization to solve FJSP-AGVs (Qin et al., 2024, 2025). However, this work has remained relatively basic and has not introduced meaningful improvements to the QD framework, such as enhancements in pairing selection strategies. In Bossens and Tarapore (2022), Tjanaka et al. (2023), evolutionary strategies were used to perturb solutions in the feature space, but they solved continuous rather than discrete problems. It obviously cannot be directly applied to the FJSP-AGVs. Therefore, we designed a Local Search based on domain knowledge of FJSP-AGVs and incorporated it into the QD algorithm to further enhance the exploitation.

## 3. Problem description

We then describe the mathematical formulation of the FJSP-AGV, which is developed with reference to Dai et al. (2019). The formulation covers the problem definition, symbols, and fundamental constraints.

The production system under study consists of a flexible job shop with $m$ machines, denoted as $M = \{1, \ldots, k, \ldots, m\}$. A total of $n$ jobs, represented by $I = \{1, \ldots, i, \ldots, n\}$, are to be processed on these machines. Each job $i$ is composed of $w_i$ operations, and its operation set is expressed as $J_i = \{1, \ldots, j, \ldots, w_i\}$. For each operation $O_{i,j}$, multiple candidate machines are available. Within the same job, different operations may be allocated to different machines rather than being restricted to a fixed one. The processing time of a given operation may vary depending on the selected machine. In addition, $v$ AGVs are responsible for transporting jobs between machines, e.g., from machine $k$ to $k'$. To ensure consistency of the model, the study assumes that, at time 0, both jobs and AGVs are available and positioned in the warehouse; once a job starts processing or transportation, it cannot be interrupted or preempted; when a job is transported for the first time, the AGV must return to the warehouse to pick it up; additional time factors, including loading and unloading, are implicitly considered within the transportation process.

### 3.1. Notations

For the sake of clarity and uniformity in the depiction of concepts and constraints, the paper employs the following notations throughout.

$i$: Index of job
$j$: Index of operation
$k$: Index of machine
$t$: Index of position
$a$: Index of AGV
$n$: The total number of jobs
$w_i$: The number of operations of job $i$
$n_{max}$: The total number of operations of all jobs
$m$: The total number of machines
$v$: The total number of AGVs
$I$: The set of jobs, $I = \{1, \ldots, i, \ldots, n\}$
$A$: The set of AGVs, $A = \{1, \ldots, a, \ldots, v\}$
$J_i$: The operation set of job $i$, $J_i = \{1, \ldots, j, \ldots, w_i\}$
$M$: The collection of all machines, denoted as $M = \{1, \ldots, k, \ldots, m\}$
$H_k$: The set of available positions assigned to machine $k$
$O_{i,j}$: Operation $j$ that belongs to job $i$
$M_{i,j}$: The group of machines capable of executing $O_{i,j}$
$T_{i,j,k}$: The processing duration of $O_{i,j}$ when performed on machine $k$
$TR_{i,j,k,k'}$: The transportation time required to move $O_{i,j}$ from machine $k$ to $k'$
$C_{i,j}$: The completion time associated with $O_{i,j}$
$S_{i,j}$: The starting time of $O_{i,j}$
$B_{k,t}$: The starting time of position $t$ assigned to machine $k$
$P_k$: Available processing time set for machine $k$
$U_{i,j}$: Specific machine responsible for processing operation $O_{i,j}$
$x_{i,j,k}$: If $O_{i,j}$ is allocated to machine $k$, the value is set to 1; Otherwise, 0
$X_{i,j,k,t}$: A binary variable that equals 1 if $O_{i,j}$ is executed at position $t$ on machine $k$, and 0 otherwise

### 3.2. The objective function and constraints

The goal of FJSP-AGVs is to minimize the makespan, denoted by $f(x)$, which is defined as follows:

$$f(x) = \max_{i \in \{1,2,\ldots,n\}} C_{i,w_i} \tag{1}$$

where variable $C_{i,w_i}$ denotes the completion time of the final operation $w_i$ within job $i$. $\max_{i \in \{1,2,\ldots,n\}} C_{i,w_i}$ represents the makespan, which is the completion time of the last operation.

The constraints for FJSP-AGVs are as follows:

$$\sum_{i \in I} \sum_{j \in J_i} X_{i,j,k,t} \leqslant 1, \forall k \in M, t \in H_k \tag{2}$$

$$\sum_{i \in I} \sum_{j \in J_i} X_{i,j,k,t+1} \leqslant \sum_{i \in I} \sum_{j \in J_i} X_{i,j,k,t}, \forall k \in M, t \in H'_k \tag{3}$$

$$C_{i,n_i} \geqslant \sum_{k\in M}\sum_{t\in H_k}\left(T_{i,n_i,k}\cdot X_{i,n_i,k,t}\cdot x_{i,j,k}\right)+S_{i,n_i},\forall i\in I \tag{4}$$

$$S_{i,j} \geqslant S_{i,j-1}+\sum_{k,k'\in M}\sum_{t,t'\in H_k,H_{k'}}\left(T_{i,j,k}+TR_{i,j-1,k',k}\right)\cdot X_{i,j-1,k',t'}\cdot X_{i,j,k,t},\forall i\in I, j\in J_i \tag{5}$$

$$\sum_{i\in I}\sum_{j,j-1\in J_i}\left(T_{i,j,k}+TR_{i,j-1,k',k}\right)\cdot X_{i,j-1,k',t'}\cdot X_{i,j,k,t} \leqslant B_{k,t+1}-B_{k,t}, k\in M, t,t'\in H_k,H_{k'} \tag{6}$$

$$B_{k,t}\geqslant S_{i,j}-L\cdot\left(1-X_{i,j,k,t}\right),\quad \forall i\in I, j\in J_i, k\in M, t\in H_k \tag{7}$$

$$B_{k,t}\leqslant S_{i,j}-L\cdot\left(1-X_{i,j,k,t}\right),\quad \forall i\in I, j\in J_i, k\in M, t\in H_k \tag{8}$$

$$B_{k,t},S_{i,j}\geqslant 0,\quad \forall i\in I, j\in J_i, k\in M, t\in H_k \tag{9}$$

Constraint (2) specifies that each position $t$ on machine $k$ is allowed to process only one operation $O_{i,j}$ at any given time. Constraint (3) ensures that an operation $O_{i,j}$ can commence only after the completion of its immediate predecessor. Constraint (4) describes the linkage among the start time, processing duration, and completion time of each operation. Constraint (5) establishes the temporal dependency between consecutive operations by incorporating both processing and transportation time. Constraint (6) clarifies the relationship between a machine's start time, the processing time of operations, and any associated transportation time. Constraints (7)–(8) ensure that the start time of operation $O_{i,j}$ is synchronized with the start time of machine $k$. Constraint (9) ensures the permissible range for the start time of operations and machines.

## 4. Quality-diversity algorithm enhanced by Q-learning

First, we give the definition of the feature space. Then, the framework of the algorithm, as well as the Q-Learning technique, is elaborated. For the selection of collaborative solution regions and the pairwise selection scheme, we give a detailed explanation. Finally, for the sub-problems of FJSP-AGV, we further elaborate on the proposed local search operators based on AGV transportation.

### 4.1. Formulation of feature space

In this study, the feature space is constructed based on two quantitative attributes that capture essential behavioral patterns of scheduling solutions: the count of machine idles ($NI$) and job transfers ($NT$). These two integer-valued features jointly characterize the way production resources are allocated and utilized within the system. The indicator $NI$ represents the total number of idle observed across all machines. A higher value of $NI$ indicates lower machine utilization, which may prolong the overall makespan. The indicator $NT$ denotes the total number of job transfers among machines, including both forward and return trips handled by AGVs. Its range is defined as $0 \leq NT \leq 2\times n_{max}$, where $n_{max}$ is the maximum number of operations, while $NI$ ranges from 0 to $n_{max}$.

These two features were selected because they jointly capture the trade-off between machine utilization and job transfer efficiency, both of which have a direct impact on production time (Pan et al., 2022b). Excessive machine idleness leads to longer completion times, whereas frequent job transfers increase transportation delays and potential machine waiting. In an extreme case where no transfers occur and all operations are executed on a single machine, other machines remain idle, resulting in an impractically long makespan. Therefore, balancing $NI$ and $NT$ is essential for achieving efficient and realistic scheduling solutions.

**Algorithm 1** Q-QD Algorithm

**Input:** $N$, $batch$, $Max\_Eval$
**Output:** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)
**procedure** Main
1: $\mathcal{P}\leftarrow\emptyset$, $\mathcal{X}\leftarrow\emptyset$
*// Generate N dimensional grid with performances* $\mathcal{P}$ *and solutions* $\mathcal{X}$
2: **for** $i\leftarrow 1$ **to** *batch* **do**
3: $x\leftarrow$ random_solution ()
4: Add_to_grid $(\mathcal{P},\mathcal{X},x)$
5: **end for**
6: Initialize *Q-table*, $iter\leftarrow 1$, $s_1\leftarrow 1$
7: **while** $iter\leqslant Max\_Eval$ **do**
8: **for** $t\leftarrow 1$ **to** $N_S$ **do**
9: Select the $t$-th best solution $x$ from $\mathcal{X}$
10: $x_c\leftarrow$ Q-Learning($\mathcal{P}$, $\mathcal{X}$, $x$, $t$, *Q-table*) *// Alg.2*
11: $x_l\leftarrow$ LS-DK($x_c,\mathcal{P}$) *// Alg.3*
12: Add_to_grid $(\mathcal{P},\mathcal{X},x_l)$
13: **end for**
14: **end while**
**return** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)
**procedure** Add_to_Grid $(\mathcal{P},\mathcal{X},x)$
15: Calculate $f(x)$ and $\boldsymbol{b}_x$ using decoding schema
16: **if** $\mathcal{P}(\boldsymbol{b}_x)=\emptyset$ or $\mathcal{P}(\boldsymbol{b}_x)>f(x)$ **then**
17: $\mathcal{P}(\boldsymbol{b}_x)\leftarrow f(x)$
18: $\mathcal{X}(\boldsymbol{b}_x)\leftarrow x$
19: **end if**

### 4.2. Framework of Q-QD

The encoding contains three vectors: (1) Operation Sequence (OS); (2) Machine Assignment (MA); and (3) AGV Selection (AS). The length of all vectors is equal to the total number of operations $n_{max}$. The decoding schema is utilized to evaluate the solution. It contains the following steps: (1) each job is allocated to the selectable machine according to the MA vector; (2) the processing order of operations is determined according to the OS vector. (3) AGVs are selected for each job based on the AS vector. The MA, OS, and AS vectors collectively constitute the solution. Using this decoding schema, the makespan is returned along with the coordinates ($NT$, $NI$) of the solution in feature space.

The proposed algorithm is implemented based on the MAP-Elites (Mouret and Clune, 2015), and all generated solutions are archived in the feature space. The MAP-Elites framework can be described as follows: given an objective function $f(x)$: $\mathcal{X}\rightarrow\mathbb{R}$ to be minimized, the algorithm seeks to identify a diverse collection of high-quality solutions (e.g., those achieving minimal makespan) rather than focusing on a single optimal one. Users can flexibly choose $N$ dimensions to formulate the feature space (in this study, $NT$ and $NI$ are adopted as the two feature dimensions, i.e., $N=2$). MAP-Elites then establishes a correspondence between each cell of the feature space and a representative high-performing solution. The feature space is characterized by a behavioral descriptor function $\boldsymbol{b}$: $\mathcal{X}\rightarrow\mathbb{R}^N$, where for any solution $x\in\mathcal{X}$, $\boldsymbol{b}_x$ produces a vector summarizing its behavioral attributes.

Algorithm 1 outlines the proposed approach. For each behavioral descriptor vector $\boldsymbol{b}(x)$, the archive $\mathcal{P}$ maintains a cell that records the best solution identified so far. Each cell can hold at most one candidate. It is worth noting that different solutions may correspond to identical behavioral descriptors. The details of the archiving procedure are presented in Lines 15–19: if a newly generated solution is mapped to an empty cell ($\mathcal{P}(\boldsymbol{b}_x)=\emptyset$), the solution is inserted into that cell; if the

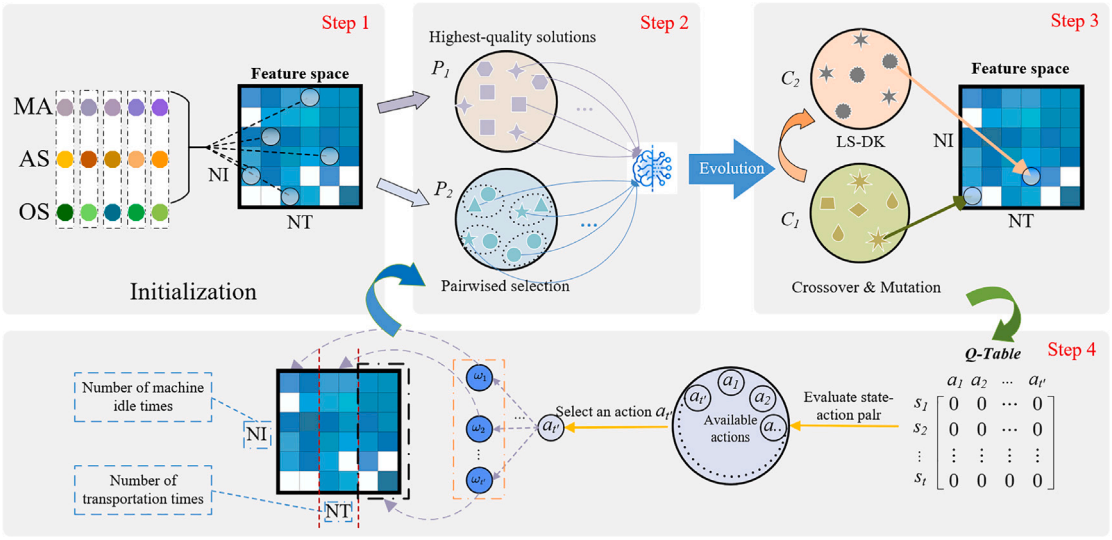



**Fig. 2.** Illustration of the Q-QD. The proposed algorithm can be divided into four steps: Step 1 represents the initialization, where a batch of solutions is randomly generated and archived into the feature space. Step 2 denotes the selection of the parent populations, where $P_1$ is the highest quality solution set selected from the feature space. $P_2$ is the set of collaborative solutions selected based on the Q-Learning region and pairwise selection schemes. In Step 3, the selected collaborative solutions are executed with Crossover_Mutation method, and after sending the reward values back to the Q-table and archiving the feature space, the Local Search continues to be performed on the offspring. The new solutions obtained continue to be used for updating the feature space. Step 4 represents the operation of the Q-Learning evaluation and selection actions. The selected action (region) will be used as the region for the pairwise selection scheme to choose collaborative solutions in Step 2.

corresponding cell is already occupied, the existing solution is replaced only when the new one yields a superior objective value ($\mathcal{P}(\boldsymbol{b}_x) > f(x)$). This mechanism guarantees that the grid consistently preserves the elite solution within each behavioral region.

The details of the Q-QD are as follows: Lines 2–5 indicate that a batch of solutions is randomly generated and archived in the feature space. Line 9 ranks the solutions in feature space and treats them as evolved solutions. Moreover, the selected solutions are also the states in the Q-table. Line 10 utilized the Q-Learning Region Selection and Pairwise Schemes to select collaborative solutions. The two selected solutions are operated on for crossover and mutation (details can be seen in Algorithm 2). Line 11 is the proposed local search based on domain knowledge (LS-DK). On the basis of the original method, we have added perturbations to the AS vector, the specific operation of which is to randomly select two operations and change the AGVs assigned to them. The details are elaborated in Section 4. D. To provide a more vivid depiction of the algorithm, we provide an illustration as shown in Fig. 2.

**Algorithm 2** Q-Learning Region Selection Mechanism

**Input:** $\mathcal{P}$, $\mathcal{X}$, $x$, $t$, *Q-table*, $\alpha$, $\gamma$, $\epsilon$
**Output:** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)
1: Execute $a_t$ using Eq.10
2: $\epsilon \leftarrow \epsilon \cdot 0.999$
3: Select the collaborative solution $x'$ based on the chosen region, following the pairwise selection scheme
4: $x_c, x_c' \leftarrow$ Crossover_Mutation $(x, x')$
5: **if** $t = N_S$ **then**
6: $\quad s_{t+1} \leftarrow 0$
7: **else**
8: $\quad s_{t+1} \leftarrow s_t + 1$
9: **end if**
10: Add_to_grid $(\mathcal{P}, \mathcal{X}, x_c(x_c'))$
11: Update $Q(s_t, a_t)$ using Eq.11
12: Update $\alpha$ using Eq.12
13: Find the higher quality solution between $x_c$ and $x_c'$, then assign it to $x_c$
**return** $x_c$

### 4.3. Region and pairwise selection schemes

$Q$-Learning identifies a region within the feature space where a collaborative solution is located, using the selection criterion of sorting by the $'NT'$ dimension in ascending order. The divided behavior feature set serves as a selectable region. The reason for choosing $'NT'$ as the division criterion and the number of divided regions will be verified in the experimental section. The details are shown in Algorithm 2. Line 1 represents the region where the collaborative solution is located by performing the action $a_t$. The selection of $a_t$ is indicated in Eq. (10).

$$a_t = \begin{cases} \underset{a \in A}{arg\max} Q\left(s_t, a\right), \ \tau \geqslant \varepsilon \\ a_{random}, \ \tau < \varepsilon \end{cases} \tag{10}$$

where $a_t$ represents the action, it determines which region in the feature space is chosen. Collaborative solutions will be selected in the chosen region. The details of region selection will be shown in the subsection. $\epsilon$ represents greedy parameter and $\tau$ means random seed from [0,1]. $Q\left(s_t, a\right)$ refers to a $Q$-value for state $s_t$. $s_t$ is defined as a single solution. $A$ encompasses the collection of all possible actions, with each distinct action corresponding to a different region in feature space. $\underset{a \in A}{arg\max} Q\left(s_t, a\right)$ is utilized to identify the action that possesses the highest $Q$-value among all possible actions for state $s_t$. $a_{random}$ denotes the selection of an action by a random seed.

Line 2 indicates that the value of $\epsilon$ is progressively decreasing. In the early stages of the iterative process, the $Q$-table gathers only a modest amount of valuable data, which may pose a risk of premature convergence in the evolutionary algorithm. Consequently, it is advisable to initialize $\epsilon$ at a higher value. Line 3 represents the selection of collaborative solutions. Inspired by Sfikas et al. (2021), the pairwise selection scheme uses the cell-based offspring improvement metric to select collaborative solution. The advantage of using this metric is that it outperforms the curiosity score metric. However, the effectiveness in solving FJSP-AGVs needs to be further verified in the experimental section. The details of the metric are as follows: Randomly select two unoccupied cells. (1) If one cell is newly discovered and the other has been successfully improved, choose the newly discovered one; (2) if

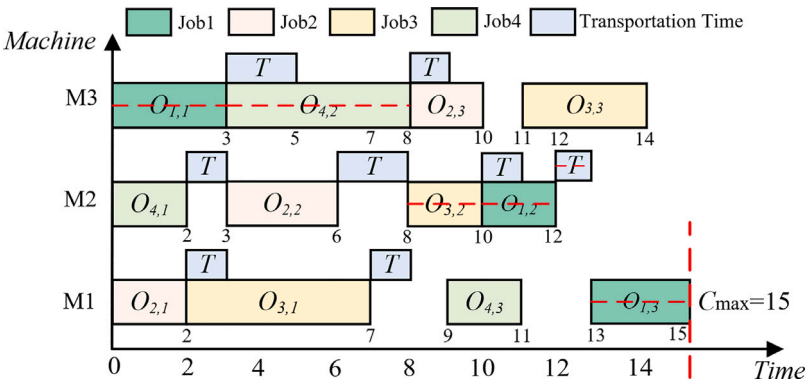



**Fig. 3.** A schematic of critical path.

both cells are newly discovered, select the cell with higher solution quality; (3) if both cells have been improved, choose the cell with more improvement counts. By using the region and pairwise selection schemes, solutions will evolve towards more promising regions for behavioral characteristics. The scheme will select parent solutions that are more likely to be improved, while ensuring diversity.

In Line 4, Crossover_Mutation method is implemented to perturb solutions $x$ and $x'$ (Li et al., 2023a), which adopted the precedence operation cross (POX) operator and the uniform crossover (UX) operator for all encoding vectors. Lines 5–9 represent the update of the state. Lines 11–12 indicate the updating of $Q(s_t, a_t)$ and $\alpha$ using Eqs. 11 and 12, respectively. The equations are described below:

$$Q\left(s_t, a_t\right) = Q\left(s_t, a_t\right) + \alpha \cdot \left[r_t + \gamma \cdot arg\max_{a \in A} Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right] \tag{11}$$

where $Q\left(s_t, a_t\right)$ denotes the $Q$ value associated with performing action $a_t$ at state $s_t$. $\alpha$ indicates the learning rate, while $\gamma$ refers to the discount factor. $r_t$ is the reward. Furthermore, $arg\max_{a \in A} Q\left(s_{t+1}, a\right)$ represents the action corresponding to the maximum $Q$-value under the next state $s_{t+1}$.

$$\alpha = \alpha - (\alpha - 0.01) \cdot iter / Max\_Iter \tag{12}$$

where $\alpha$ indicates the learning rate. $iter$ represents the current iteration count. $Max\_Eval$ denotes the total number of evaluations.

### 4.4. Local search based on domain knowledge heuristic rules

Since most of the operators proposed are based on the critical path, we first provide the definition of the critical path.

#### 4.4.1. Critical path

The critical path is characterized by a series of activities, including operations and AGVs, that culminate in the longest total duration within the schedule timeline (Lu et al., 2021). Operations that lie on this path are seen as critical operations. The motivation for incorporating the critical path concept lies in its ability to identify the most time-sensitive operations that directly determine the overall completion time of the schedule. By explicitly utilizing this path, the algorithm can focus local search efforts on operations and AGV movements that exert the greatest influence on makespan reduction. This targeted adjustment not only improves optimization efficiency but also enhances the ability of algorithm to coordinate machine utilization and transportation scheduling in a more integrated manner.

On the critical path, machines must operate continuously without idletime between successive operations and AGV transportation. The length of the critical path represents the makespan. Fig. 3 shows a schematic of the critical path (red dashed line).

Perturbation of the three vectors primarily focuses on modifying the index positions. Following the idea in Li et al. (2023a), the local search

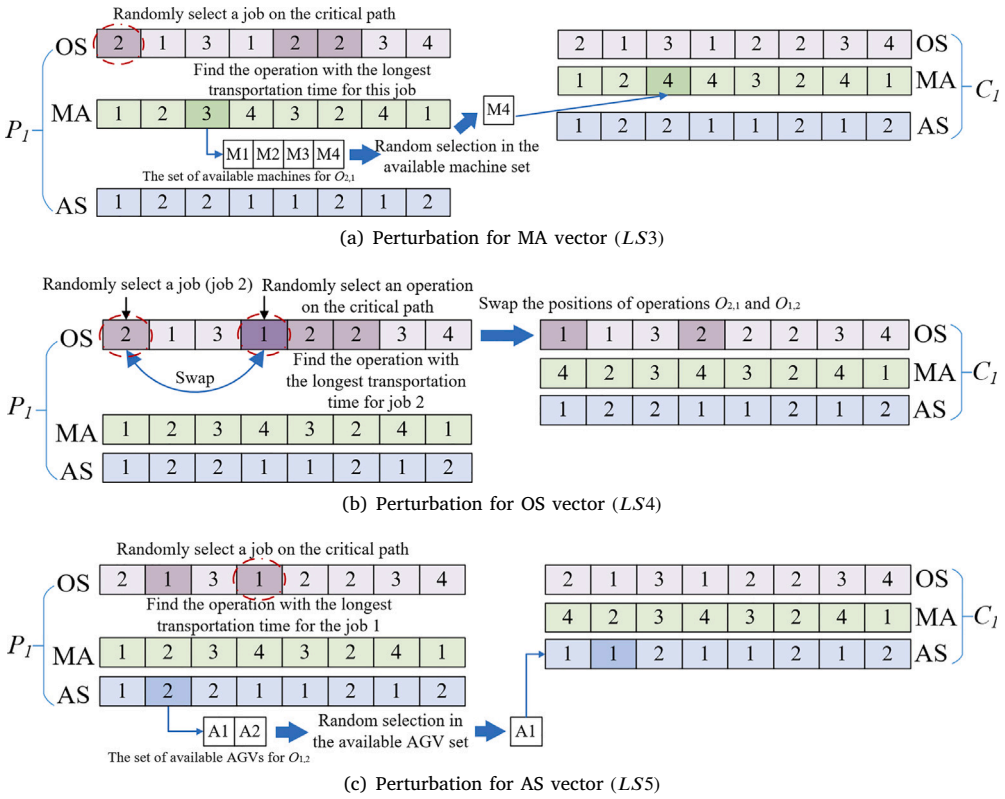
**Algorithm 3** LS-DK $(x, \mathcal{P})$

**Input:** $x, \mathcal{P}$
**Output:** $x'$
1: $x$ is encoded as MA, OS, *and* AS
2: Calculate the critical path *ope_path*
3: Randomly select one job $i$ from *ope_path*
4: Sort all operations within this job in descending order of AGV transportation time
5: Find $O_{i,j}$ with the longest AGV transportation time
6: **if** the ordering of MA vector is adjusted **then**
7: Change the machine for $O_{i,j}$ // $LS3$
8: **else if** the ordering of OS vector is adjusted **then**
9: Select a operation and swap its position with $O_{i,j}$ // $LS4$
10: **else if** the ordering of AS vector is adjusted **then**
11: Change the AGV for $O_{i,j}$ // $LS5$
12: **end if**
13: A new vector $x'$ is generated
**return** $x'$

operator $LS1$ is constructed with reference to the critical path mechanism. In this procedure, a critical operation along the critical path is randomly chosen, and its assigned machine is adjusted among the feasible options. To further improve the efficiency of $LS$, we propose the following hypothesis: can the effectiveness of the proposed algorithm be enhanced by incorporating heuristic rules derived from AGV transportation time into $LS$? Thus, we designed $LS$ that incorporates domain knowledge heuristic rules. The essential difference between it and other studies is that the ordering of the adjustments is based on the AGV transportation time rather than random selection. For example, in $LS2$, a job is selected and the transportation time required for all of its operations is sorted in descending order. Choose the operation that takes the longest transportation time by the AGV and change its selected machine. $LS3$-5 can be integrated into a unified description which integrates perturbations of OS, MA, and AS vectors. The details are given in the Algorithm 3.

From the Algorithm 3, it can be seen that the adjustments of MA, OS, and AS vectors exhibit a convergent behavior. In each iteration, a job $i$ is selected from the critical path *ope_path*, and the operation associated with the maximum AGV transportation time is identified. The corresponding element in the vector is then repositioned accordingly. In Line 13, if any of the MA, OS, or AS vectors is updated, it will be concatenated with the other two vectors into a new solution $x'$.

For ease of description, we give an example of the operation of $LS3$-5 as shown in Fig. 4. The three vectors are altered separately by the AGV transportation heuristic rules, which in turn generate the offspring solution. $LS3$ starts by randomly choosing a job located on the critical path from the $OS$ vector. It then determines the operation of this job that corresponds to the maximum transportation time and retrieves its associated $MA$ value. Subsequently, $LS3$ selects a new




(a) Perturbation for MA vector ($LS3$)

(b) Perturbation for OS vector ($LS4$)

(c) Perturbation for AS vector ($LS5$)

**Fig. 4.** The illustration of LS-DK.

machine from the available machine set for that operation (for example, M4 is chosen) and substitutes the original $MA$ value in the vector. The modified $MA$ vector, combined with the unchanged $OS$ and $AS$ vectors, forms a new individual to complete the perturbation stage. $LS4$ initiates the process by randomly picking an operation from the $OS$ vector, followed by exchanging its position with another operation belonging to a different job that has the longest transportation duration. After the swap, the $OS$ vector is updated, while the $MA$ and $AS$ vectors remain fixed, thereby generating a new individual and concluding the perturbation step. $LS5$ begins by randomly selecting a job along the critical path within the $OS$ vector. It then identifies the operation of that job with the highest transportation time and determines its current AGV allocation from the $AS$ vector. The OS and MA vectors remain unchanged, and the modified AS vector, together with them, forms a newly perturbed individual.

### 4.5. Computational complexity of Q-QD

The complexity of Q-QD consists of an initialization phase and an iterative improvement phase. The initialization phase generates $batch$ random solutions and inserts them into the feature-performance grid, leading to the complexity of $O$(batch). The iterative phase runs for at most $Max\_Eval$ iterations, where each iteration performs $NS$ updates, including selecting the t-th best solution ($O$(logN)), Q-learning and crossover operations ($O$(N)), and LS-DK, which involves sorting operations within a job ($O$(N)). Given that these steps are repeated $NS$ times per iteration, the overall computational burden of the iterative process grows asymptotically as $O(Max_Eval \times NS \times N)$. As this stage constitutes the primary computational component, the total time complexity of the Q-QD is also estimated to be $O(Max\_Eval \times NS \times N)$.

## 5. Experiments

This section performs the essential parameter tests, verifies the effectiveness of the proposed strategy, and compares it with the existing state-of-the-art algorithms. We also test the selection of which behavioral feature to divide, the difference between Q-Learning selection and random selection, and the relationship between the Q-Learning selection region and the final best solution. Additionally, we compare the pairwise selection scheme with the most widely used selection schemes, thereby verifying its superiority. Moreover, two real-world cases are tested: (1) a Chinese coal machine structural parts production workshop; (2) a Chinese machining workshop, Nanjing. Finally, we discuss the proposed algorithm.

### 5.1. Evaluation instances

Following Yao et al. (2024), Dai et al. (2019), two real-world manufacturing cases are employed for experimental validation: (1) a production workshop for structural components of Chinese coal machinery, and (2) a machining workshop located in Nanjing, China. Furthermore, 21 extended benchmark instances are generated based on the practical problems described in Dai et al. (2019). For each instance, 20 independent runs are performed. The number of operations varies within $\{20, 30, 40, 50, 80, 100, 120\}$, while the number of machines is chosen from $\{5, 8, 10\}$. For every job $i$, the number of operations $w_i$ is uniformly distributed in $[1, 5]$. The processing time $T_{i,j,k}$ of all operations ranges from $[5, 40]$, and the AGV transportation time between two distinct machines lies in $[1, 5]$. It is assumed that the round-trip time between any pair of machines remains identical. The maximum number of evaluations for all experiments is set as $Max\_Iter = 20 \cdot \sum_1^n w_i \cdot m$.

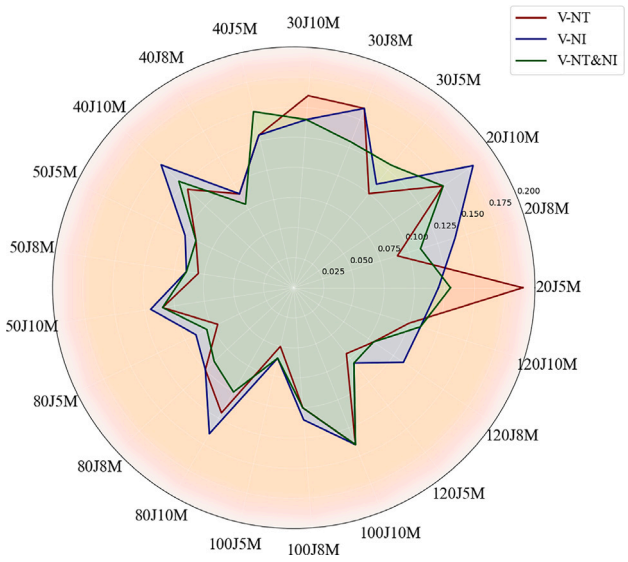



**Fig. 5.** Effect of different region division ways on RPI values.

### 5.2. Configuration and evaluation metric

All experiments were executed on a 64-bit Windows 11 platform equipped with an Intel Core i7-13790F processor (2.10 GHz) and 16 GB of RAM. The algorithms were implemented in the Python environment using the PyCharm 2023 IDE.

For performance assessment, this study adopts the Relative Percentage Increment (RPI) (Qin et al., 2022; Li et al., 2024a), a well-established indicator frequently applied in FJSP-AGV research for single-objective optimization. The RPI quantifies the relative deviation between makespan achieved by a specific approach and the best-known solution, as defined below:

$$RPI_a = \left(c_a - c_{best}\right) / c_{best} \times 100\% \tag{13}$$

where $c_a$ denotes the objective value (makespan) obtained by algorithm $a$, and $c_{best}$ corresponds to the smallest objective value among all algorithms. A smaller $RPI_a$ indicates better performance of algorithm $a$.

### 5.3. The way of dividing regions

Before conducting parameter tests, we first validated and analyzed three different strategies for dividing regions within the feature space: (1) division based on the number of job transfers ($NT$) (V-NT), (2) division based on the number of machine idle times ($NI$) (V-NI), and (3) joint division based on both $NT$ and $NI$ (V-NT&NI). To determine which strategy yields the best performance, we conducted experiments across instances of all sizes. As shown in Fig. 5, V-NT achieved 71.4% (15/21) of the best RPI values, outperforming V-NI with 19.0% (4/21) and V-NT&NI with 47.6% (10/21). This indicates that using the behavioral feature $NT$ alone for feature-space division, thereby defining the "actions" available to Q-Learning, produces the most effective results.

The reason lies in the characteristics of FJSP-AGV: transportation by AGVs is a dominant factor influencing job completion time. Therefore, defining regions based on $NT$ allows the algorithm to more accurately capture the underlying dynamics of the scheduling problem. Moreover, focusing on a single feature simplifies the learning process and enables Q-Learning to concentrate on the most critical source of variation in solution behavior. This focused design enhances learning stability and convergence efficiency. Based on these findings, all subsequent experiments employ $NT$ as the criterion for region division.

### 5.4. Parameter testing

The Taguchi method described in Li et al. (2023a) is employed to analyze the sensitivity of five control parameters: the number of selected elite solutions (N-S), the number of selected regions (N-R), the learning rate $\alpha$, the discount factor $\gamma$, and the greedy coefficient $\epsilon$. Each parameter is examined under three experimental levels. Their corresponding ranges are defined as follows: N-S $\in \{5, 10, 20\}$, N-R $\in \{2, 4, 6\}$, $\alpha \in \{0.4, 0.6, 0.8\}$, $\gamma \in \{0.4, 0.6, 0.8\}$, and $\epsilon \in \{0.4, 0.6, 0.8\}$. The experimental design adopts an orthogonal array $L_{18}(3^5)$, producing 18 distinct parameter combinations. Different instance sizes are evaluated based on these configurations. Fig. 6 presents the factor-level trend plots of the five parameters, demonstrating how variations in parameter settings influence the performance of the Q-QD algorithm. N-S=5 performed best, indicating that using fewer elite individuals as parents at a time ensures the quality of the offspring. The reason for this may be that too many regions cause the algorithm to overfit on specific behavioral features, while fewer regions can help the algorithm to keep exploring a wide range of behavioral features. Both $\alpha$ and $\gamma$ are best when they are 0.8. The reason for this may be that lower learning rates slow the learning process and reduce the likelihood of falling into local optimum. Higher discount rates help the algorithm to take into account long-term rather than short-term rewards. $\epsilon$ works best when it takes the middle value. High $\epsilon$ promotes exploration but might destabilize learning, while low $\epsilon$ favors exploitation, risking local optima. In summary, the parameters of Q-QD are set as follows: N-S = 5, N-R = 2, $\alpha = 0.8$, $\gamma = 0.8$, $\epsilon = 0.6$.

### 5.5. Validation of the necessity for Q-learning region selection

Following the parameter analysis, we observed that the algorithm performs best when the number of regions (N-R) is set to 2. To further evaluate the necessity and effectiveness of using Q-Learning for region selection, we conducted a comparison experiment against a variant using random region selection (denoted as V-RR), in which the feature space is divided by $NT$ but regions are chosen randomly instead of being guided by Q-Learning. As shown in Fig. 7, Q-QD achieved the best RPI values in 76.2% (16/21) of all instances, while V-RR achieved only 42.9% (9/21). These results clearly demonstrate that Q-Learning significantly enhances the effectiveness of region selection.

Unlike random selection, Q-Learning adaptively updates its policy based on accumulated rewards, historical performance, and long-term outcomes. This learning-driven mechanism allows Q-QD to identify and exploit promising regions more effectively, reducing randomness in exploration and improving the overall convergence trend. The Q-Learning strategy not only stabilizes the search process but also accelerates convergence toward high-quality solutions by prioritizing regions with greater potential for improvement. Hence, Q-Learning serves as a crucial component that enables intelligent, data-driven feature-space exploration within the Q-QD framework. The impact of removing Q-Learning will be further examined in the Ablation Experiments subsection.

### 5.6. Ablation experiment

To evaluate the strategies in Q-QD, we conducted ablation experiments and compared the proposed algorithm with four variants: (1) V-QD, which removes the QD framework; (2) V-QL, which removes the Q-Learning component; (3) V-I, which removes the pairwise selection scheme; and (4) V-LS, which removes the designed local search. The comparative outcomes are presented in Table 2. Besides the RPI metric, both the MEAN and BEST results across 20 independent runs are reported for all instances. The final row (AVG) indicates the overall average of each performance indicator over all instances. For clarity, the best-performing results of each algorithm are emphasized in boldface. It is also noted that "J" and "M" in the instance labels correspond to the number of jobs and machines, respectively.

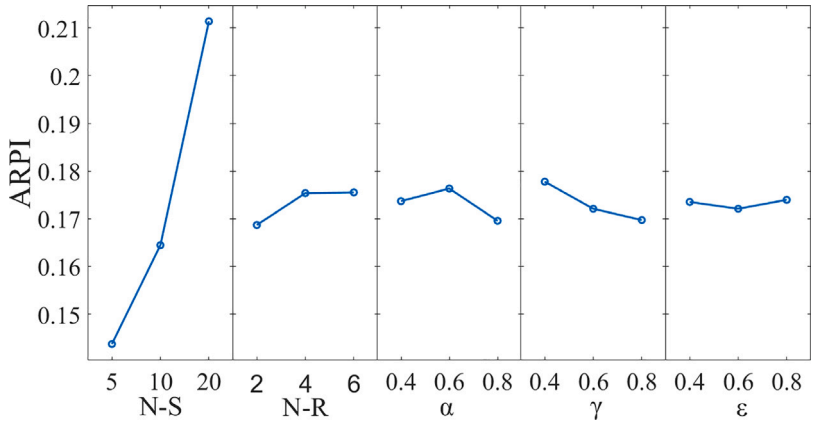



**Fig. 6.** Effect of different parameters on average RPI values.

**Table 2**
Comprehensive numerical results of all variants.

| Instance | ARPI | | | | | MEAN | | | | | BEST | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | V-QD | V-QL | V-I | V-LS | Q-QD | V-QD | V-QL | V-I | V-LS | Q-QD | V-QD | V-QL | V-I | V-LS | Q-QD |
| 20J5M | 0.34 | 0.56 | 0.10 | 0.15 | **0.08** | 356.9 | 415.3 | 294.4 | 306.7 | **289.4** | 316 | 383 | 271 | 283 | **267** |
| 20J8M | 0.37 | 0.73 | 0.12 | 0.17 | **0.09** | 339.9 | 429.0 | 277.5 | 289.0 | **270.7** | 303 | 408 | 249 | 263 | **248** |
| 20J10M | 0.49 | 0.86 | 0.16 | 0.26 | **0.10** | 304.9 | 380.2 | 237.4 | 257.4 | **224.9** | 255 | 358 | 216 | 233 | **204** |
| 30J5M | 0.28 | 0.56 | **0.10** | 0.11 | 0.11 | 604.6 | 736.9 | **518.5** | 523.5 | 524.9 | 529 | 695 | **471** | 493 | **471** |
| 30J8M | 0.32 | 0.67 | 0.10 | 0.13 | **0.09** | 429.0 | 541.2 | 355.6 | 366.5 | **354.1** | 386 | 488 | 325 | 341 | **324** |
| 30J10M | 0.42 | 0.84 | 0.13 | 0.23 | **0.12** | 333.8 | 433.2 | 266.6 | 288.1 | **264.3** | 307 | 404 | 245 | 269 | **235** |
| 40J5M | 0.22 | 0.47 | 0.10 | 0.11 | **0.10** | 777.3 | 937.5 | 701.3 | 707.8 | **699.8** | 724 | 883 | **639** | 640 | 664 |
| 40J8M | 0.25 | 0.67 | **0.08** | 0.13 | 0.09 | 578.0 | 769.3 | **499.8** | 519.9 | 505.2 | 551 | 698 | 464 | 492 | **462** |
| 40J10M | 0.32 | 0.74 | 0.12 | 0.18 | **0.09** | 529.0 | 698.0 | 449.7 | 474.1 | **437.1** | 499 | 644 | 428 | 445 | **402** |
| 50J5M | 0.14 | 0.42 | 0.07 | 0.08 | **0.07** | 967.3 | 1203.1 | 908.8 | 917.3 | **903.7** | 887 | 1133 | 849 | 849 | **847** |
| 50J8M | 0.21 | 0.63 | 0.09 | 0.11 | **0.08** | 728.8 | 976.9 | 655.4 | 668.4 | **650.8** | 680 | 926 | 619 | 621 | **601** |
| 50J10M | 0.28 | 0.72 | 0.11 | 0.15 | **0.11** | 617.6 | 828.7 | 536.1 | 553.2 | **532.9** | 556 | 786 | **482** | 509 | 495 |
| 80J5M | 0.11 | 0.39 | **0.10** | 0.10 | 0.10 | 1482.7 | 1859.8 | **1472.4** | 1475.1 | 1473.5 | 1423 | 1772 | 1392 | **1341** | 1393 |
| 80J8M | 0.14 | 0.55 | **0.09** | 0.10 | 0.10 | 1063.5 | 1440.8 | **1011.4** | 1023.4 | 1019.6 | 1004 | 1379 | 950 | **931** | 957 |
| 80J10M | 0.17 | 0.60 | 0.09 | 0.12 | **0.07** | 975.3 | 1335.3 | 908.0 | 936.8 | **891.9** | 922 | 1238 | 848 | 865 | **835** |
| 100J5M | 0.07 | 0.35 | **0.07** | 0.10 | 0.08 | 1908.4 | 2410.4 | **1898.3** | 1958.8 | 1923.8 | 1842 | 2277 | 1793 | 1861 | **1781** |
| 100J8M | 0.08 | 0.47 | 0.06 | 0.06 | **0.05** | 1253.8 | 1700.1 | 1224.3 | 1232.8 | **1217.2** | 1163 | 1639 | **1160** | 1188 | 1176 |
| 100J10M | 0.16 | 0.65 | **0.12** | 0.15 | 0.13 | 1160.6 | 1650.4 | **1121.8** | 1146.5 | 1124.2 | 1112 | 1583 | **999** | 1045 | 1068 |
| 120J5M | 0.05 | 0.33 | 0.07 | 0.09 | **0.06** | **2470.3** | 3120.3 | 2507.1 | 2547.8 | 2488.7 | 2403 | 3069 | 2413 | 2427 | **2344** |
| 120J8M | 0.07 | 0.49 | **0.06** | 0.08 | 0.10 | 1569.2 | 2169.3 | **1542.3** | 1577.6 | 1600.5 | 1496 | 2118 | 1463 | **1460** | 1480 |
| 120J10M | 0.12 | 0.58 | 0.09 | 0.10 | **0.08** | 1472.0 | 2079.9 | 1434.0 | 1440.4 | **1418.6** | 1432 | 2015 | 1357 | 1362 | **1313** |
| AVG | 0.22 | 0.58 | 0.10 | 0.13 | **0.09** | 948.7 | 1243.6 | 896.2 | 914.8 | **896.0** | 894.8 | 1185.5 | 839.7 | 853.2 | **836.5** |

As can be seen from Table 2, Q-QD achieves the best RPI values on 66.7% (14/21) of the instances, while V-I achieves 33.3% (7/21), and all other variants achieve none. For the MEAN and BEST values, Q-QD also outperforms the other variants on most instances. According to the average results, Q-QD achieves the best RPI (0.09), MEAN (896), and BEST (836.5) among all algorithms. Further comparison results are shown in the box plots and win-rate charts in this section. As illustrated in Fig. 8, Q-QD achieves the smallest mean and median values. The results of V-I show that the pairwise selection scheme has the least effect on overall performance. In contrast, the win-rate analysis shows that Q-QD achieves the highest number of wins after incorporating the pairwise selection scheme, indicating its contribution to improving performance. Among all strategies, the QD framework and Q-Learning have the most significant impact on the effectiveness of algorithms, followed by the local search.

### 5.7. Effects of pairwise selection scheme

Based on the box plot in Fig. 8, a question arises: How much advantage does the improvement metric pairwise selection scheme have over mainstream selection schemes? To verify its utility, this section compares it with the following widely used selection schemes: (1) Random seed selection (V-S), (2) Binary tournament selection and elite selection strategy (V-T) (Han et al., 2024), (3) Roulette wheel selection (V-R) (Li et al., 2024b), and (4) Curiosity score (V-C, a QD selection strategy) (Cully and Demiris, 2018). Fig. 9 gives an illustration of a box plot and win rate plots. Statistically, Q-QD achieved the smallest mean and median values, which were superior to the other variants in 95.2% (20/21) of the instances. V-S achieved the best RPI value in only 4.8% (1/21) of the instances. Followed by V-C, which achieved the best RPI values in 14.3% (3/21) of the instances, while the number of V-T and V-R achieving the best values was 0%. In the win rate plots, V-S and V-C performed the best when Q-QD was excluded. When Q-QD was included, it won over all variants. The reason may be that: The pairwise selection scheme guides the selection process through specific criteria (e.g., number of new discoveries, number of improvements) rather than relying on randomness or simple fitness scores, which helps to point more directly in the direction of potential improvements. By prioritizing newly discovered cells, the scheme encourages the exploration of new possibilities, which may not be prominent enough in roulette and tournament selection, as they focus more on confirmed fitness. Compared to the curiosity score, the pairwise selection scheme considers both the quality of the solution and the number of improvements during the selection process, rather than being based solely on curiosity or novelty, allowing the algorithm to focus on solutions that show improvement while maintaining diversity.

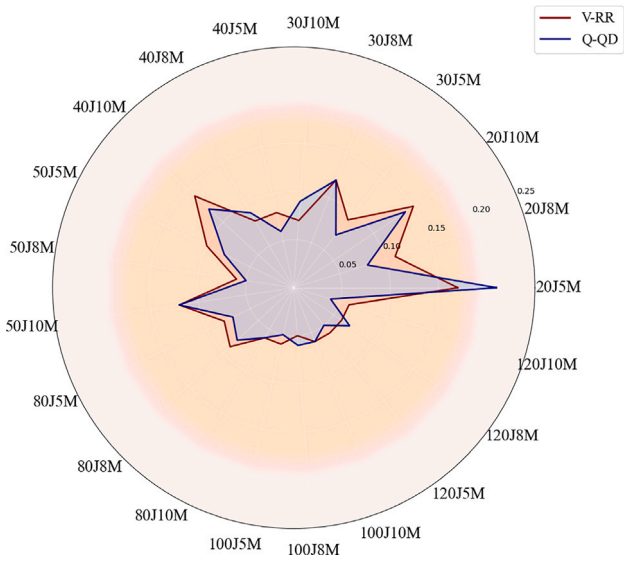



**Fig. 7.** Effect of random seed selection vs. Q-learning selection on RPI values.

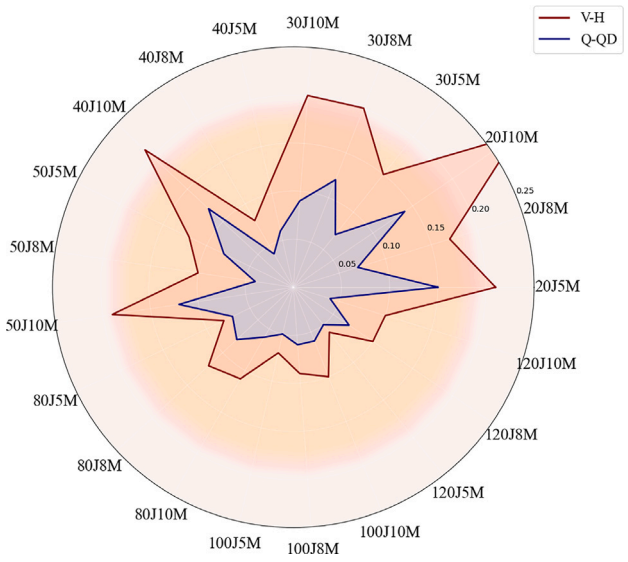



**Fig. 10.** Comparison between Q-QD and the variant without AGV transportation heuristic rules.

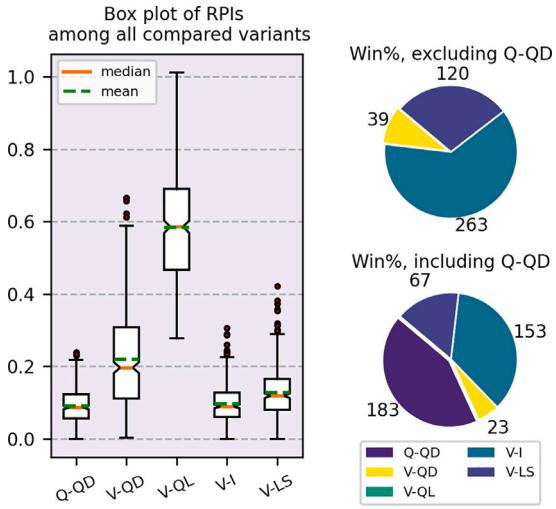



**Fig. 8.** Comparison among Q-QD and variants.

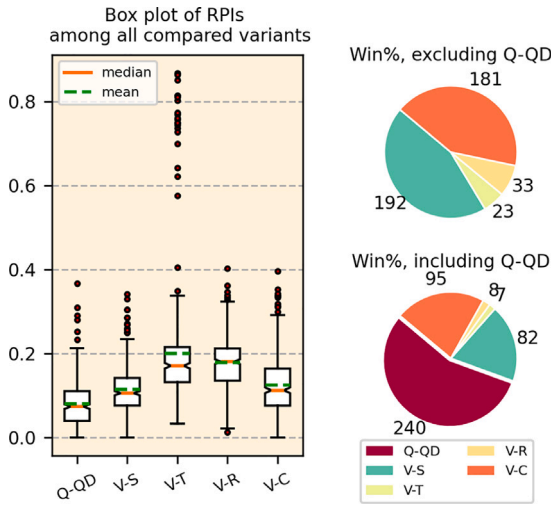



**Fig. 9.** Comparison among variants with different selection schemes.

### *5.8. Validation of effects for local search based on AGV transportation heuristic rules*

Compared with the mainstream Local Search, how much the effect of the proposed Local Search based on AGV transportation heuristic rule has been improved, which needs to be further verified. We conducted a design of experiments to eliminate all heuristic rules with AGV transportation and retain the critical path and the perturbation approach to vectors (named as V-H). As shown in Fig. 10, according to the tests on 21 instances, we obtained that: the Local Search designed in this paper obtained the best PRI values on all instances (21/21). Moreover, the final average RPI value was calculated to be about 38.5% higher than that of V-H. The reason for this may be that by prioritizing operations with long AGV transport times, the algorithm is able to identify and focus on those parts that may have the greatest impact on the overall scheduling efficiency. Heuristic rules help to reduce ineffective perturbations on unimportant or low-impact operations, making it more likely that each perturbation will have a positive effect. When dealing with scheduling problems with complex transportation demands and multiple AGVs, heuristic rules can better adapt to this complexity and provide more rational scheduling solutions.

### *5.9. Remarks on the effects of different variants*

The ablation experiments and comparison results provide clear insights into the contribution of different components in Q-QD. Among all the variants, V-QL proves to be the most crucial component, as its removal leads to the most significant performance drop. Without Q-Learning, the algorithm loses its ability to intelligently guide solution evolution, resulting in inefficient region selection, which significantly impacts the final scheduling performance. Compared to random region selection (V-RR), Q-Learning achieves 76.2% of the best RPI values, confirming its role in stabilizing and improving solution evolution.

The QD framework also plays a fundamental role, as removing it results in a complete loss of diversity maintenance and adaptive exploration. The results show that Q-QD consistently outperforms V-QD in both RPI and solution quality metrics, highlighting the crucial role of QD in sustaining an appropriate exploration. Q-Learning guides the search direction, whereas the QD framework is what enables the algorithm to retain high-quality solutions, making it another indispensable component.

**Table 3**
Performance of all algorithms.

| Instance | ARPI | | | | | | MEAN | | | | | | BEST | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QD | IGSA | IGA | DCGA | DQNMA | Q-QD | QD | IGSA | IGA | DCGA | DQNMA | Q-QD | QD | IGSA | IGA | DCGA | DQNMA | Q-QD |
| 20J5M | 0.75† | 0.64† | 0.51† | 0.49† | 0.62† | **0.15** | 450.60 | 423.25 | 390.50 | 384.05 | 416.85 | **297.45** | 414 | 378 | 341 | 347 | 375 | **258** |
| 20J8M | 0.95† | 0.83† | 0.47† | 0.48† | 0.76† | **0.07** | 472.8 | 444 | 358.4 | 359.95 | 428.8 | **260.25** | 441 | 414 | 324 | 321 | 384 | **243** |
| 20J10M | 1.15† | 0.96† | 0.60† | 0.56† | 0.87† | **0.14** | 444.75 | 406.4 | 330.2 | 322.35 | 387.25 | **236.7** | 427 | 369 | 292 | 289 | 360 | **207** |
| 30J5M | 0.65† | 0.63† | 0.39† | 0.32† | 0.51† | **0.07** | 786.3 | 775.3 | 661.35 | 629.95 | 717.35 | **507.5** | 754 | 724 | 590 | 585 | 684 | **476** |
| 30J8M | 0.96† | 0.86† | 0.54† | 0.51† | 0.67† | **0.12** | 610 | 579.8 | 480.8 | 470.4 | 519.95 | **347.9** | 573 | 545 | 426 | 424 | 490 | **312** |
| 30J10M | 1.00† | 0.88† | 0.52† | 0.52† | 0.73† | **0.09** | 487.75 | 459.35 | 370.7 | 369.75 | 422.05 | **266.95** | 462 | 419 | 327 | 326 | 372 | **244** |
| 40J5M | 0.55† | 0.50† | 0.29† | 0.28† | 0.40† | **0.06** | 998.9 | 969.65 | 835.4 | 827.15 | 906.6 | **684.5** | 981 | 902 | 772 | 766 | 859 | **644** |
| 40J8M | 0.70† | 0.60† | 0.27† | 0.28† | 0.49† | **0.04** | 832.3 | 781.45 | 618.85 | 625.25 | 729 | **508.15** | 781 | 741 | 551 | 584 | 684 | **489** |
| 40J10M | 0.99† | 0.88† | 0.36† | 0.49† | 0.74† | **0.12** | 774.45 | 734.2 | 532.1 | 582.55 | 679 | **436.3** | 721 | 676 | 486 | 537 | 636 | **390** |
| 50J5M | 0.53† | 0.50† | 0.25† | 0.26† | 0.38† | **0.08** | 1259.25 | 1239.55 | 1033.5 | 1034.75 | 1140.5 | **886.6** | 1196 | 1153 | 956 | 969 | 1084 | **824** |
| 50J8M | 0.70† | 0.64† | 0.26† | 0.29† | 0.49† | **0.04** | 1037.8 | 999.4 | 770.45 | 784.25 | 910.9 | **631.55** | 990 | 951 | 704 | 714 | 876 | **610** |
| 50J10M | 0.95† | 0.86† | 0.37† | 0.47† | 0.64† | **0.12** | 920.9 | 878.1 | 649.55 | 695.3 | 777.95 | **528.65** | 867 | 800 | 578 | 638 | 733 | **473** |
| 80J5M | 0.44† | 0.41† | 0.20† | 0.19† | 0.29† | **0.07** | 1934.6 | 1887 | 1605.25 | 1595.45 | 1730.95 | **1432.75** | 1866 | 1767 | 1492 | 1497 | 1662 | **1342** |
| 80J8M | 0.66† | 0.61† | 0.22† | 0.28† | 0.44† | **0.08** | 1515.75 | 1471.2 | 1113.1 | 1165.75 | 1317.05 | **989.6** | 1461 | 1416 | 976 | 1072 | 1233 | **914** |
| 80J10M | 0.77† | 0.69† | 0.21† | 0.32† | 0.52† | **0.06** | 1445.5 | 1384.1 | 992.95 | 1081 | 1243.7 | **866.05** | 1400 | 1296 | 899 | 960 | 1159 | **818** |
| 100J5M | 0.39† | 0.36† | 0.13† | 0.14† | 0.24† | **0.05** | 2494.15 | 2452.85 | 2030.15 | 2051.1 | 2226.75 | **1883.2** | 2412 | 2284 | 1887 | 1938 | 2125 | **1798** |
| 100J8M | 0.58† | 0.51† | 0.14† | 0.23† | 0.37† | **0.06** | 1783.1 | 1701.3 | 1282 | 1381.75 | 1539.75 | **1192.6** | 1703 | 1592 | 1186 | 1264 | 1448 | **1126** |
| 100J10M | 0.68† | 0.61† | 0.15† | 0.28† | 0.48† | **0.06** | 1740.15 | 1671.9 | 1196.95 | 1330.25 | 1533.25 | **1097** | 1670 | 1560 | 1117 | 1198 | 1409 | **1038** |
| 120J5M | 0.35† | 0.33† | 0.09† | 0.13† | 0.21† | **0.05** | 3174.4 | 3121.8 | 2574.3 | 2662.85 | 2851.85 | **2460.1** | 3035 | 3011 | 2422 | 2578 | 2749 | **2351** |
| 120J8M | 0.58† | 0.51† | 0.11† | 0.23† | 0.39† | **0.07** | 2255.55 | 2161.4 | 1580.8 | 1756.3 | 1990.4 | **1526.6** | 2196 | 2077 | 1457 | 1668 | 1884 | **1429** |
| 120J10M | 0.65† | 0.60† | 0.10† | 0.24† | 0.48† | **0.04** | 2188.1 | 2118.1 | 1451.1 | 1642.55 | 1954.05 | **1376.35** | 2114 | 1957 | 1324 | 1522 | 1850 | **1323** |
| AVG | 0.72 | 0.64 | 0.30 | 0.34 | 0.51 | **0.08** | 1270.95 | 1227.10 | 970.37 | 1005.51 | 1123.50 | **852.02** | 1217.50 | 1153.75 | 889.15 | 933.75 | 1060.30 | **799.30** |

V-LS also proves to be highly effective, although its impact is slightly lower than that of Q-Learning and the QD framework. Without this mechanism, the algorithm's ability to refine solutions and reduce ineffective perturbations is weakened, leading to higher makespan values. The experiment comparing V-LS to V-H (without AGV heuristics) shows an average 38.5% improvement in RPI, indicating that AGV-aware search significantly enhances scheduling efficiency.

In contrast, V-I has the least impact on overall performance compared to the other key components. The findings indicate that although V-I surpasses the other ablated variants once Q-QD is excluded, the removal of Q-QD does not lead to a significant degradation in solution quality. The win-rate analysis confirms that V-I contributes to fine-tuning the search process but is not as critical as Q-Learning, the QD framework, or local search.

In summary, the performance ranking of different components is evident: Q-Learning > QD Framework > Local Search > Pairwise Selection.

### 5.10. Comparison experiment and analysis

To verify the effectiveness of the Q-QD approach, a comparative study was conducted against five representative state-of-the-art algorithms. For fairness, all algorithmic parameters were configured in accordance with their original settings. The compared algorithms are as follows: classical QD (Cazenille, 2018) based on MAP-Elites, the IG and Simulated Annealing Algorithm (IGSA) (Li et al., 2022a) for FJSP with transportation, Improved Genetic Algorithm (IGA) for Multi-AGV FJSP (Meng et al., 2023), Dual Population Collaborative GA (DCGA) for FJSP-AGVs (Han et al., 2024), and Memetic Algorithm based on DQN (DQNMA) for FJSP-AGVs (Zhang et al., 2024). The reason for choosing the above algorithms is that they are all solved FJSPs with transportation constraints and are the most up-to-date algorithms.

We extend the case of Dai et al. (2019) and continue with the comparison of algorithm performance. As shown in Table 3, compared to the best performing IGA, Q-QD improves the average completion time by roughly 12.2% on the MEAN values. We also performed the Wilcoxon test with a significance level of 0.05. † indicates that there is a significant difference between the comparison algorithm and Q-QD. As in Table 3, Q-QD significantly outperforms other compared approaches, further validating the superiority of the proposed algorithm. In addition, to further see how different algorithms are ranked in terms of overall performance, Friedman tests are conducted in this paper. This paper showed box plots and win rate charts for all the algorithm results (Fig. 11). From the box plot results, it was clear that Q-QD obtained the best median and average RPI values. According to the win rate charts, Q-QD won in most of the test cases (94.1%). Followed by IGA, which won in 5.9% of the instances. After eliminating Q-QD, IGA was winning even more in the majority of the instances. As shown in Fig. 12, the proposed Q-QD won the first rank and scored 1.31 points higher than the second-ranked IGA.

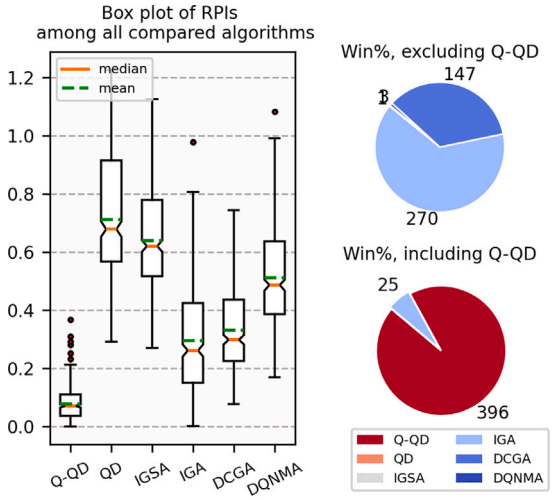



**Fig. 11.** Comparison results among Q-QD and algorithms.

All the above experiments confirm the superiority of the proposed Q-QD algorithm, which is mainly attributed to the effective integration of Q-Learning and the QD optimization framework. The Q-Learning-based region selection mechanism enables the algorithm to adaptively identify promising behavioral feature regions through a reward-driven learning process, rather than relying on random or static selection. By continuously updating its policy based on experience, Q-Learning guides the search toward regions that are more likely to contain high-quality solutions, thus achieving a better balance between exploration

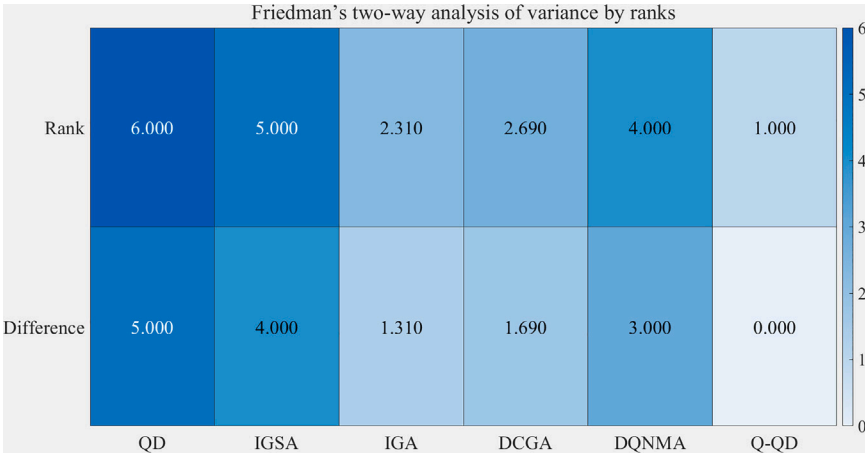



**Fig. 12.** Friedman Test of Q-QD and compared algorithms.

and exploitation. This adaptive selection significantly improves search efficiency and accelerates convergence, allowing Q-QD to focus computational resources on feature regions with higher optimization potential. The pairwise selection scheme further enhances this process by refining solution choices within the selected regions. Through comparative evaluation of candidate solutions based on improvement metrics, it ensures that superior individuals are retained while maintaining population diversity, preventing premature convergence, and sustaining exploration ability throughout the search process.

Furthermore, the local search designed with domain-specific knowledge of FJSP-AGVs plays a complementary role in improving solutions. By incorporating AGV transportation heuristics and critical-path reasoning, the local search perturbs encoding vectors in a more informed way, enabling the algorithm to explore scheduling sequences with higher precision. This knowledge-guided mechanism improves both local exploitation efficiency and the diversity of discovered solutions, enhancing the robustness of the overall optimization process. In essence, Q-Learning provides adaptive exploitation ability, while the QD framework ensures the maintenance of diversity and quality across the feature space. Their joint effect establishes a dynamic and self-adaptive optimization mechanism, which fundamentally explains the superior performance of Q-QD in achieving substantial improvements in makespan reduction across various benchmark instances.

To illustrate the convergence behavior of the compared algorithms, evolutionary curves are plotted. Three representative instances of varying scales are selected in this study, namely 20J8M (small), 50J5M (medium), and 100J10M (large). As depicted in Fig. 13, the $X$-axis denotes the evolutionary iteration, while the $Y$-axis indicates the corresponding completion time. It can be observed from the figure that Q-QD converges more rapidly than the other algorithms and yields superior-quality solutions. The algorithm maintains consistent convergence performance across different instance scales, primarily owing to the Q-Learning-based intelligent region selection strategy and the Local Search component enhanced by AGV transportation heuristics. This integrated search mechanism, which combines intelligent selection with domain-specific knowledge, not only accelerates the search process of the algorithm but also strengthens its exploitation capability when addressing FJSP-AGVs.

### 5.11. Real-world case studies

Tables 4 and 5 show the results of all algorithms for 20 tests in the two real-world cases. Their practical application scenarios are the Chinese coal machine structural parts production workshop (Yao et al., 2024) and the Chinese machining workshop, Nanjing (Dai et al., 2019), respectively. In the first real-world case, the workshop layout settings are as shown in Fig. 14. The workshop is equipped with 11 machines, comprising 6 types of processing equipment, with some operations being performed on parallel machines. Two AGVs are employed to handle the transportation of jobs among machines. In this case, a total of 20 jobs are scheduled. For the second real-world scenario, the workshop consists of 10 machines processing 10 distinct jobs (comprising 32 operations in total), and the number of AGVs is fixed at 2. Due to confidentiality constraints, the detailed layout of the production workshop cannot be disclosed. Nevertheless, the original literature provides datasets that are publicly available for testing, which are also adopted in this study. Owing to page limitations, their detailed descriptions are omitted here. To verify whether the performance differences between Q-QD and the compared algorithms are statistically significant, the Wilcoxon signed-rank test was conducted at a 0.05 significance level. The symbol † denotes cases where the performance difference between Q-QD and the baseline algorithms is statistically significant. Experimental outcomes demonstrate that the proposed algorithm achieves the lowest RPI values across all benchmark tests and consistently surpasses the other comparison methods. These findings further confirm the robustness and practical efficiency of the proposed algorithm when applied to real-world FJSP-AGVs.

### 5.12. Discovery and discussion

This section further explores the relationship between Q-Learning action selection and the region where the best solution is eventually obtained. We randomly selected five instances of different scales, i.e., 20J10M, 40J8M, 80J5M, 100J8M, and 120J10M. As shown in Table 6, impr_val denotes the cumulative sum of the reduced makespan values. impr_num denotes the number of times the Q-Learning improves when it selects different regions (actions), and "best solution region" denotes the region where the best solution is located. After analysis, we find that Q-Learning selects the left region (i.e., relatively few AGV transportations) with the highest quality improvement of the solution. Additionally, it is worth noting that the regions where good solutions are found consistently align with areas where Q-Learning has made significant improvements and where the number of AGV transports is low.

The study shows that Q-Learning is effective in identifying regions that contribute significantly to the makespan reduction, which usually coincide with the regions where the best solutions are located. The analysis shows that Q-Learning prefers regions with fewer AGV transports, which contributes to the improvement of solution quality and the makespan reduction. In addition, the best solution tends to appear in the region where Q-learning improvement is significant and the number of AGV transportation is low, confirming that the number of AGV transportation is a key factor affecting makespan. In FJSP-AGV,

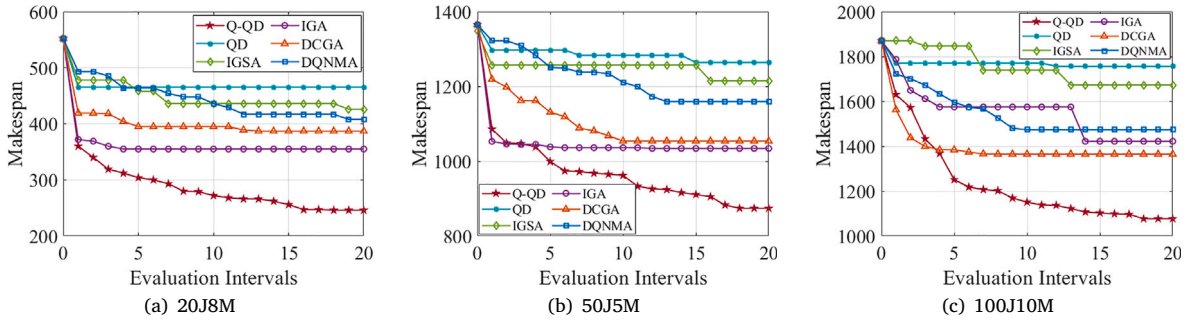



**Fig. 13.** Evolutionary curves of all comparison algorithms.

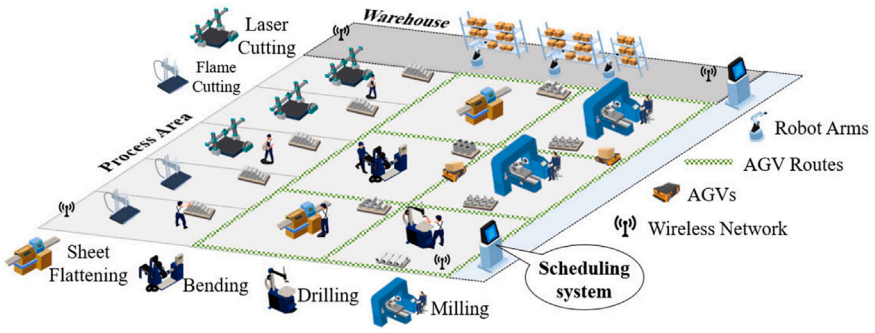



**Fig. 14.** Workshop layout of the Coal machine structural parts production workshop (Yao et al., 2024).

**Table 4**
Comparison results of all algorithms on Chinese coal machine structural parts production workshop.

| 20J11M | QD | IGSA | IGA | DCGA | DQNMA | Q-QD |
|---|---|---|---|---|---|---|
| TEST1 | 0.48† | 0.47† | 0.30† | 0.28† | 0.59† | **0.12** |
| TEST2 | 0.44† | 0.41† | 0.26† | 0.22† | 0.59† | **0.07** |
| TEST3 | 0.55† | 0.58† | 0.29† | 0.27† | 0.55† | **0.09** |
| TEST4 | 0.53† | 0.40† | 0.33† | 0.19† | 0.55† | **0.04** |
| TEST5 | 0.57† | 0.43† | 0.32† | 0.26† | 0.57† | **0.00** |
| TEST6 | 0.57† | 0.46† | 0.26† | 0.30† | 0.58† | **0.01** |
| TEST7 | 0.52† | 0.44† | 0.23† | 0.25† | 0.59† | **0.07** |
| TEST8 | 0.52† | 0.53† | 0.29† | 0.22† | 0.49† | **0.06** |
| TEST9 | 0.49† | 0.50† | 0.35† | 0.27† | 0.57† | **0.03** |
| TEST10 | 0.53† | 0.48† | 0.31† | 0.28† | 0.47† | **0.14** |
| TEST11 | 0.54† | 0.45† | 0.38† | 0.30† | 0.61† | **0.07** |
| TEST12 | 0.56† | 0.44† | 0.28† | 0.26† | 0.52† | **0.10** |
| TEST13 | 0.48† | 0.50† | 0.32† | 0.32† | 0.51† | **0.04** |
| TEST14 | 0.59† | 0.43† | 0.27† | 0.27† | 0.59† | **0.02** |
| TEST15 | 0.54† | 0.46† | 0.28† | 0.28† | 0.60† | **0.00** |
| TEST16 | 0.55† | 0.44† | 0.26† | 0.23† | 0.56† | **0.04** |
| TEST17 | 0.51† | 0.47† | 0.26† | 0.28† | 0.57† | **0.19** |
| TEST18 | 0.58† | 0.47† | 0.21† | 0.32† | 0.59† | **0.10** |
| TEST19 | 0.53† | 0.36† | 0.25† | 0.28† | 0.58† | **0.02** |
| TEST20 | 0.54† | 0.49† | 0.35† | 0.30† | 0.51† | **0.11** |
| ARPI | 0.53 | 0.46 | 0.29 | 0.27 | 0.56 | **0.07** |

the transportation efficiency of AGV directly affects the productivity and job completion time.

Furthermore, the findings of this study offer several managerial insights for manufacturing enterprises seeking to improve production efficiency under transportation constraints. First, the proposed Q-QD framework provides a practical decision-support tool for production managers to allocate limited AGV resources more effectively. By integrating Q-Learning, the method can dynamically identify scheduling behaviors that minimize idle machine time and transportation delays, thereby reducing the overall makespan. Second, the interpretability of the feature-space division allows managers to visualize how transportation frequency and machine utilization interact, enabling more informed operational adjustments. Finally, the adaptive learning ability of QD supports decision-making in real-world FJSP environments, contributing to smarter and more sustainable manufacturing management.

## 6. Conclusions

This paper proposed a Q-QD algorithm to solve the challenges of scheduling AGVs in FJSP. The method leverages Q-Learning for intelligent region selection within the feature space and employs a pairwise selection scheme to determine collaborative solutions. In addition, a heuristic rule-based Local Search integrating AGV transportation

**Table 5**
Comparison results of all algorithms on Chinese machining workshop, Nanjing.

| 10J10M | QD | IGSA | IGA | DCGA | DQNMA | Q-QD |
|---|---|---|---|---|---|---|
| TEST1 | 1.02† | 0.70† | 1.23† | 0.73† | 0.92† | **0.26** |
| TEST2 | 0.99† | 0.86† | 0.84† | 0.80† | 0.99† | **0.20** |
| TEST3 | 1.00† | 0.60† | 0.90† | 0.80† | 0.99† | **0.28** |
| TEST4 | 0.82† | 0.73† | 1.11† | 0.54† | 0.76† | **0.35** |
| TEST5 | 1.06† | 0.84† | 0.80† | 0.42† | 0.92† | **0.17** |
| TEST6 | 1.04† | 0.89† | 0.69† | 0.79† | 0.83† | **0.26** |
| TEST7 | 1.03† | 0.77† | 0.60† | 0.65† | 0.99† | **0.25** |
| TEST8 | 1.10† | 0.68† | 0.76† | 0.62† | 0.82† | **0.12** |
| TEST9 | 1.02† | 0.73† | 1.19† | 0.57† | 1.05† | **0.00** |
| TEST10 | 1.01† | 0.64† | 0.93† | 0.82† | 1.00† | **0.07** |
| TEST11 | 1.07† | 0.70† | 0.71† | 0.86† | 0.91† | **0.08** |
| TEST12 | 1.05† | 0.75† | 1.09† | 0.73† | 1.01† | **0.14** |
| TEST13 | 0.99† | 0.91† | 0.97† | 0.71† | 0.88† | **0.23** |
| TEST14 | 0.98† | 0.85† | 0.71† | 0.72† | 0.88† | **0.11** |
| TEST15 | 1.09† | 0.74† | 0.77† | 0.74† | 0.84† | **0.31** |
| TEST16 | 0.98† | 0.73† | 0.78† | 0.86† | 0.89† | **0.25** |
| TEST17 | 1.07† | 0.91† | 0.86† | 0.89† | 0.90† | **0.08** |
| TEST18 | 1.11† | 0.85† | 0.52† | 0.64† | 0.76† | **0.37** |
| TEST19 | 0.97† | 0.72† | 0.84† | 0.93† | 0.91† | **0.12** |
| TEST20 | 1.02† | 0.78† | 0.88† | 0.65† | 0.97† | **0.23** |
| ARPI | 1.02 | 0.77 | 0.86 | 0.72 | 0.91 | **0.19** |

**Table 6**
The relationship between the best solution region and Q-learning action selection.

| | | | | | | |
|---|---|---|---|---|---|---|
| **20J10M** | impr_val | impr_num | best solution region: | 1 | | |
| action:1 | **87775** | **965** | makespan | 247 | | |
| action:2 | 74380 | 907 | NT: | 156 | NI: | 48 |
| **40J8M** | impr_val | impr_num | best solution region: | 1 | | |
| action:1 | **318236** | **2193** | makespan | 517 | | |
| action:2 | 252990 | 2042 | NT: | 297 | NI: | 59 |
| **80J5M** | impr_val | impr_num | best solution region: | 1 | | |
| action:1 | **1434240** | **3232** | makespan | 1471 | | |
| action:2 | 1336849 | 2921 | NT: | 520 | NI: | 56 |
| **100J8M** | impr_val | impr_num | best solution region: | 1 | | |
| action:1 | **1837804** | **5453** | makespan | 1179 | | |
| action:2 | 1685459 | 4782 | NT: | 678 | NI: | 110 |
| **120J10M** | impr_val | impr_num | best solution region: | 1 | | |
| action:1 | **2504412** | **9143** | makespan | 1383 | | |
| action:2 | 1760805 | 6126 | NT: | 881 | NI: | 211 |

knowledge was designed to enhance the search process. The proposed approach effectively handles the complex issues of FJSP-AGVs, offering a significant improvement over previous optimization methods.

### 6.1. Findings

The results verify that the proposed Q-QD substantially outperforms existing state-of-the-art algorithms. The integration of Q-Learning for region selection proved to be a key enhancement, as it adaptively focuses the search on regions with fewer AGV transfers, thereby reducing the makespan. The results also show that the best-performing solutions consistently emerge from regions where Q-Learning has guided the exploration process, underscoring the critical influence of AGV transportation times on scheduling quality. Moreover, the Local Search based on AGV transportation heuristics effectively identifies high-quality and diverse solutions, further strengthening the robustness and efficiency of Q-QD.

### 6.2. Research limitations

Despite these promising results, certain limitations remain. In the design of the current grid, the distribution of solutions is concentrated within specific regions, resulting in partial coverage of the grid. Due to the discrete nature of the FJSP-AGV problem, the feature space cannot be completely filled as in continuous optimization problems. This leads to inefficient memory utilization and a moderate reduction in search efficiency. Furthermore, the present study focuses primarily on single-factory scheduling scenarios. More complex conditions, such as multi-factory coordination, job assembly constraints, machine breakdowns, and emergency job arrivals, have not yet been incorporated into the problem.

### 6.3. Recommendations for future research

Future work will aim to overcome these limitations. First, refining the Q-Learning mechanism to dynamically adapt to real-time changes in manufacturing environments could improve long-term optimization stability. Second, changing the feature-space structure to enhance solution diversity and grid coverage would increase search efficiency. Additionally, extending the Q-QD framework to multi-factory systems and incorporating more realistic constraints — such as job assembly, machine breakdowns, and dynamic job arrivals — would enhance its applicability to complex industrial settings. Finally, integrating the proposed approach with advanced optimization and learning techniques, including evolutionary strategies (Huang et al., 2018; Yang et al., 2023; Kang et al., 2024), critical-path-based accelerated evaluation (He et al., 2023), and graph neural networks (Huang et al., 2023), may further boost scheduling efficiency and adaptability.

## CRediT authorship contribution statement

**Haoxiang Qin:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yi Xiang:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Yuyan Han:** Writing – review & editing, Supervision, Formal analysis. **Yuting Wang:** Writing – review & editing, Supervision, Formal analysis. **Junqing Li:** Writing – review & editing, Supervision. **Quanke Pan:** Writing – review & editing, Supervision.

## Declaration of competing interest

Authors declare that they have no personal relationships with other people or organizations that can inappropriately influence our work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

## Acknowledgments

This work is supported by Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515030022); Natural Science Foundation of Guangdong Province, China (No. 2022A1515110058); the Fundamental Research Funds for the Central Universities, China (No. 2024ZYGXZR097, 93K172024K03); National Natural Science Foundation of China (No. 61906069, 62566012); the Natural Science Research Project of Education Department of Guizhou Province (No. QJJ2023061). We also acknowledge the use of ChatGPT-5 (OpenAI) to assist with English grammar correction and language refinement during manuscript preparation. The tool was used for linguistic editing, and it did not contribute to data collection or other generation of scientific content.

## Data availability

No data was used for the research described in the article.

## References

Abbasi, S., Soltanifar, E., Tahmasebi Aghbelaghi, D., Ghasemi, P., 2025. Designing a sustainable and smart supply chain considering a green computing approach in the post-COVID period. Sustain. Comput.: Informatics Syst. 46, 101117. http://dx.doi.org/10.1016/j.suscom.2025.101117.

Alazaidah, R., Al-Shaikh, A., Al-Mousa, M.R., Khafajah, H., Samara, G., Alzyoud, M., Al-Shanableh, N., Almatarneh, S., 2024. Website phishing detection using machine learning techniques. J. Stat. Appl. Probab. 13 (1), 119–129. http://dx.doi.org/10.18576/jsap/130108.

Alvarez, A., Dahlskog, S., Font, J., Togelius, J., 2022. Interactive constrained MAP-elites: Analysis and evaluation of the expressiveness of the feature dimensions. IEEE Trans. Games 14 (2), 202–211. http://dx.doi.org/10.1109/TG.2020.3046133.

An, Y., Chen, X., Gao, K., Li, Y., Zhang, L., 2023. Multiobjective flexible job-shop rescheduling with new job insertion and machine preventive maintenance. IEEE Trans. Cybern. 53 (5), 3101–3113. http://dx.doi.org/10.1109/TCYB.2022.3151855.

Aripov, M., Alhag, A.A., Shaddad, S.A., Ali, N.B.M., Hussin, H.A.A.A., 2025. Analysis of the stock market using the integration of statistical and machine learning models. J. Stat. Appl. Probab. 14 (1), 115–122. http://dx.doi.org/10.18576/jsap/140109.

Bossens, D.M., Tarapore, D., 2022. Quality-diversity meta-evolution: Customizing behavior spaces to a meta-objective. IEEE Trans. Evol. Comput. 26 (5), 1171–1181.

Cao, Z., Lin, C., Zhou, M., 2021. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. IEEE Trans. Autom. Sci. Eng. 18 (1), 56–69. http://dx.doi.org/10.1109/TASE.2019.2945717.

Cazenille, L., 2018. QDpy: A python framework for quality-diversity. GitLab repository. https://gitlab.com/leo.cazenille/qdpy.

Chatzilygeroudis, K., Cully, A., Vassiliades, V., Mouret, J.-B., 2020. Quality-diversity optimization: A novel branch of stochastic optimization. arXiv preprint arXiv:2012.04322.

Chen, X., Li, J., Wang, Z., Chen, Q., Gao, K., Pan, Q., 2025. Optimizing dynamic flexible job shop scheduling using an evolutionary multi-task optimization framework and genetic programming. IEEE Trans. Evol. Comput. http://dx.doi.org/10.1109/TEVC.2025.3543770, Swarm and Evolutionary Computation, 90 101658. DOI: 10.1016/j.swevo.2024.101658.

Chen, R., Wu, B., Wang, H., Tong, H., Yan, F., 2024. A Q-learning based NSGA-II for dynamic flexible job shop scheduling with limited transportation resources.

Colas, C., Madhavan, V., Huizinga, J., Clune, J., 2020. Scaling MAP-elites to deep neuroevolution. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, New York, NY, USA, pp. 67–75. http://dx.doi.org/10.1145/3377930.3390217.

Cully, A., Clune, J., Tarapore, D., Mouret, J.-B., 2015. Robots that can adapt like animals. Nature 521 (7553), 503–507. http://dx.doi.org/10.1038/nature14422.

Cully, A., Demiris, Y., 2018. Quality and diversity optimization: A unifying modular framework. IEEE Trans. Evol. Comput. 22 (2), 245–259. http://dx.doi.org/10.1109/TEVC.2017.2704781.

Dai, M., Tang, D., Giret, A., Salido, M.A., 2019. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robot. Comput.-Integr. Manuf. 59, 143–157. http://dx.doi.org/10.1016/j.rcim.2019.03.011.

Dauzère-Pérès, S., Ding, J., Shen, L., Tamssaouet, K., 2023. The flexible job shop scheduling problem: A review. European J. Oper. Res. http://dx.doi.org/10.1016/j.ejor.2023.05.017.

Ding, J., Lü, Z., Li, C.-M., Shen, L., Xu, L., Glover, F., 2019. A two-individual based evolutionary algorithm for the flexible job shop scheduling problem. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI Press, p. 280. http://dx.doi.org/10.1609/aaai.v33i01.33012262.

Du, Y., Li, J., Chen, X., Duan, P., Pan, Q., 2023. Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. IEEE Trans. Emerg. Top. Comput. Intell. 7 (4), 1036–1050. http://dx.doi.org/10.1109/TETCI.2022.3145706.

Du, Y., Li, J., Li, C., Duan, P., 2024. A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. IEEE Trans. Neural Networks Learn. Syst. 35 (4), 5695–5709. http://dx.doi.org/10.1109/TNNLS.2022.3208942.

Du, Y., Li, J., Luo, C., Meng, L., 2021. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. Swarm Evol. Comput. 62, 100861. http://dx.doi.org/10.1016/j.swevo.2021.100861.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J., 2021. First return, then explore. Nature 590 (7847), 580–586. http://dx.doi.org/10.1038/s41586-020-03157-9.

Elhag, A.A., 2024. Prediction and classification of tuberculosis using machine learning. J. Stat. Appl. Probab. 13 (3), 939–946. http://dx.doi.org/10.18576/jsap/130308.

Flageat, M., Chalumeau, F., Cully, A., 2023. Empirical analysis of PGA-MAP-elites for neuroevolution in uncertain domains. ACM Trans. Evol. Learn. Optim. 3 (1), 1. http://dx.doi.org/10.1145/3577203.

Gao, K., Yang, F., Zhou, M., Pan, Q., Suganthan, P.N., 2019. Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm. IEEE Trans. Cybern. 49 (5), 1944–1955. http://dx.doi.org/10.1109/TCYB.2018.2817240.

Girgis, M.R., Zaki, A.M., Elgeldawi, E., Abdallah, M.M., Ahmed, A.A., 2025. MACT: A novel framework for automated mobile application testing using machine learning. Appl. Math. Inf. Sci. 19 (5), 1079–1092. http://dx.doi.org/10.18576/amis/190509.

Grillotti, L., Cully, A., 2022. Unsupervised behavior discovery with quality-diversity optimization. IEEE Trans. Evol. Comput. 26 (6), 1539–1552. http://dx.doi.org/10.1109/TEVC.2022.3159855.

Hamid, I.Y., Yousif, S.M.A., Aljaloud, S., Abaker, A.O.I., Elemam, H.Z.S., Al-ruwaitee, K.A., Anja, M.I.A., 2025. Machine learning techniques in a hybrid forecasting model for oil prices combining ARIMA. Appl. Math. Inf. Sci. 19 (6), 1241–1252. http://dx.doi.org/10.18576/amis/190601.

Han, X., Cheng, W., Meng, L., Zhang, B., Gao, K., Zhang, C., Duan, P., 2024. A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV. Swarm Evol. Comput. 86, 101538. http://dx.doi.org/10.1016/j.swevo.2024.101538.

He, X., Pan, Q.-K., Gao, L., Neufeld, J.S., Gupta, J.N.D., 2024. Historical information based iterated greedy algorithm for distributed flowshop group scheduling problem with sequence-dependent setup times. Omega 123, 102997. http://dx.doi.org/10.1016/j.omega.2024.102997.

He, X., Pan, Q.-K., Gao, L., Wang, L., Suganthan, P.N., 2023. A greedy cooperative co-evolutionary algorithm with problem-specific knowledge for multiobjective flowshop group scheduling problems. IEEE Trans. Evol. Comput. 27 (3), 430–444.

Hu, H., Jia, X., He, Q., Fu, S., Liu, K., 2020. Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. Comput. Ind. Eng. 149, 106749. http://dx.doi.org/10.1016/j.cie.2020.106749.

Huang, J.-P., Gao, L., Li, X.-Y., 2024. A hierarchical multi-action deep reinforcement learning method for dynamic distributed job-shop scheduling problem with job arrivals. IEEE Trans. Autom. Sci. Eng. 1–13. http://dx.doi.org/10.1109/TASE.2024.3380644.

Huang, J.-P., Gao, L., Li, X.-Y., Zhang, C.-J., 2023. A novel priority dispatch rule generation method based on graph neural network and reinforcement learning for distributed job-shop scheduling. J. Manuf. Syst. 69, 119–134. http://dx.doi.org/10.1016/j.jmsy.2023.06.007.

Huang, H., Liu, F., Yang, Z., Hao, Z., 2018. Automated test case generation based on differential evolution with relationship matrix for iFogSim toolkit. IEEE Trans. Ind. Informatics 14 (11), 5005–5016.

Kang, N., Miao, Z., Pan, Q.-K., Li, W., Tasgetiren, M.F., 2024. Multi-objective teaching-learning-based optimizer for a multi-weeding robot task assignment problem. Tsinghua Sci. Technol. 29 (5), 1249–1265.

Lei, D., Li, M., Wang, L., 2019. A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. IEEE Trans. Cybern. 49 (3), 1097–1109. http://dx.doi.org/10.1109/TCYB.2018.2796119.

Li, J., Du, Y., Gao, K., Duan, P., Gong, D., Pan, Q., Suganthan, P.N., 2022a. A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. IEEE Trans. Autom. Sci. Eng. 19 (3), 2153–2170.

Li, X., Gao, L., 2016. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int. J. Prod. Econ. 174, 93–110.

Li, R., Gong, W., Wang, L., Lu, C., Dong, C., 2023a. Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling. IEEE Trans. Syst. Man, Cybern.: Syst. 1–11. http://dx.doi.org/10.1109/TSMC.2023.3305541.

Li, R., Gong, W., Wang, L., Lu, C., Pan, Z., Zhuang, X., 2023b. Double DQN-based coevolution for green distributed heterogeneous hybrid flowshop scheduling with multiple priorities of jobs. IEEE Trans. Autom. Sci. Eng. 1–13. http://dx.doi.org/10.1109/TASE.2023.3327792.

Li, Y., Gu, W., Yuan, M., Tang, Y., 2022b. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. Robot. Comput.-Integr. Manuf. 74, 102283. http://dx.doi.org/10.1016/j.rcim.2022.102283.

Li, M., Lei, D., 2021. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. Eng. Appl. Artif. Intell. 103, 104307.

Li, J., Li, J., Gao, K., Duan, P., 2024a. A hybrid graph-based imitation learning method for a realistic distributed hybrid flow shop with family setup time. IEEE Trans. Syst. Man, Cybern.: Syst. 54 (12), 7291–7304. http://dx.doi.org/10.1109/TSMC.2024.3449413.

Li, W., Li, H., Wang, Y., Han, Y., 2024b. Optimizing flexible job shop scheduling with automated guided vehicles using a multi-strategy-driven genetic algorithm. Egypt. Informatics J. 25, 100437.

Li, J., Liu, Z., Li, C., Zheng, Z., 2021. Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem. IEEE Trans. Fuzzy Syst. 29 (11), 3234–3248. http://dx.doi.org/10.1109/TFUZZ.2020.3016225.

Lin, C., Cao, Z., Zhou, M., 2022. Learning-based grey wolf optimizer for stochastic flexible job shop scheduling. IEEE Trans. Autom. Sci. Eng. 19 (4), 3659–3671. http://dx.doi.org/10.1109/TASE.2021.3129439.

Lu, C., Gao, L., Yi, J., Li, X., 2021. Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China. IEEE Trans. Ind. Informatics 17 (10), 6687–6696. http://dx.doi.org/10.1109/TII.2020.3043734.

Luo, Q., Deng, Q., Gong, G., Zhang, L., Han, W., Li, K., 2020. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers. Expert Syst. Appl. 160, 113721. http://dx.doi.org/10.1016/j.eswa.2020.113721.

Marzouki, B., Driss, O.B., Ghedira, K., 2022. Improved chemical reaction optimization for distributed flexible job shop problem with transportation times. IFAC-PapersOnLine 55 (10), 1249–1254. http://dx.doi.org/10.1016/j.ifacol.2022.09.561.

Meng, L., Cheng, W., Zhang, B., Zou, W., Fang, W., Duan, P., 2023. An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem. Sensors 23 (8).

Meng, L., Zhang, C., Ren, Y., Zhang, B., Lv, C., 2020. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Comput. Ind. Eng. 142, 106347.

Mouret, J.-B., Clune, J., 2015. Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909.

Padre, N., Seixas, M.B., dos Santos, P.V., Bressan, G.M., Lima, H.O.S., Scoczynski, M., 2025. Transcriptomic pattern analysis in breast cancer patients: A machine learning approach. Appl. Math. Inf. Sci. 19 (5), 1183–1192. http://dx.doi.org/10.18576/amis/190517.

Pan, Z., Lei, D., Wang, L., 2022a. A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling. IEEE Trans. Syst. Man, Cybern.: Syst. 52 (8), 5295–5307. http://dx.doi.org/10.1109/TSMC.2021.3120702.

Pan, Z., Wang, L., Wang, J., Yu, Y., Li, R., 2024. Distributed energy-efficient flexible manufacturing with assembly and transportation: A knowledge-based bi-hierarchical optimization approach. IEEE Trans. Autom. Sci. Eng. 1–17.

Pan, Z., Wang, L., Zheng, J., Chen, J.-F., Wang, X., 2022b. A learning-based multi-population evolutionary optimization for flexible job shop scheduling problem with finite transportation resources. IEEE Trans. Evol. Comput. http://dx.doi.org/10.1109/TEVC.2022.3219238, 1-1.

Pugh, J.K., Soros, L.B., Stanley, K.O., 2016. Quality diversity: A new frontier for evolutionary computation. Front. Robot. AI 3.

Qian, C., Xue, K., Wang, R.-J., 2024. Quality-diversity algorithms can provably be helpful for optimization. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. IJCAI '24, Jeju, Korea, pp. 1–9. http://dx.doi.org/10.24963/ijcai.2024/773, 773.

Qin, H., Han, Y., Chen, Q., Wang, L., Wang, Y., Li, J., Liu, Y., 2023. Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints. IEEE Trans. Emerg. Top. Comput. Intell. 7 (5), 1442–1457. http://dx.doi.org/10.1109/TETCI.2023.3271331.

Qin, H.-X., Han, Y.-Y., Zhang, B., Meng, L.-L., Liu, Y.-P., Pan, Q.-K., Gong, D.-W., 2022. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. Swarm Evol. Comput. 69, 100992. http://dx.doi.org/10.1016/j.swevo.2021.100992.

Qin, H., Xiang, Y., Han, Y., Yan, X., 2024. Optimizing energy-efficient flexible job shop scheduling with transportation constraints: A Q-learning enhanced quality-diversity algorithm. In: Proceedings of the 2024 6th International Conference on Data-Driven Optimization of Complex Systems. DOCS, pp. 373–378. http://dx.doi.org/10.1109/DOCS63458.2024.10704469.

Qin, H., Xiang, Y., Liu, F., Han, Y., Wang, Y., 2025. Enhancing quality-diversity algorithm by reinforcement learning for flexible job shop scheduling with transportation constraints. Swarm Evol. Comput. 93, 101849. http://dx.doi.org/10.1016/j.swevo.2025.101849.

Sfikas, K., Liapis, A., Yannakakis, G.N., 2021. Monte Carlo elites: Quality-diversity selection as a multi-armed bandit problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO'21, Association for Computing Machinery, New York, NY, USA, pp. 180–188. http://dx.doi.org/10.1145/3449639.3459321.

Su, J., Huang, H., Li, G., Li, X., Hao, Z., 2023. Self-organizing neural scheduler for the flexible job shop problem with periodic maintenance and mandatory outsourcing constraints. IEEE Trans. Cybern. 53 (9), 5533–5544.

Tao, X.-R., Pan, Q.-K., Gao, L., 2024. An iterated greedy algorithm with reinforcement learning for distributed hybrid FlowShop problems with job merging. IEEE Trans. Evol. Comput. http://dx.doi.org/10.1109/TEVC.2024.3443874, 1-1.

Tjanaka, B., Fontaine, M.C., Lee, D.H., Kalkar, A., Nikolaidis, S., 2023. Training diverse high-dimensional controllers by scaling covariance matrix adaptation MAP-annealing. IEEE Robot. Autom. Lett. 8 (10), 6771–6778.

Urquhart, N., Hart, E., 2018. Optimisation and illumination of a real-world workforce scheduling and routing application (WSRP) via map-elites. In: Auger, A., Fonseca, C., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (Eds.), In: Parallel Problem Solving from Nature – PPSN XV, vol. 11101, Springer, Cham.

Urquhart, N., Hart, E., Hutcheson, W., 2020. Using MAP-elites to support policy making around workforce scheduling and routing. At - Autom. 68, 110–117, Retrieved from https://api.semanticscholar.org/CorpusID:210921499.

Urquhart, N., Höhl, S., Hart, E., 2019. An illumination algorithm approach to solving the micro-depot routing problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. Association for Computing Machinery, New York, NY, USA, pp. 1347–1355. http://dx.doi.org/10.1145/3321707.3321767.

Wang, G.-G., Gao, D., Pedrycz, W., 2022. Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. IEEE Trans. Ind. Informatics 18 (12), 8519–8528. http://dx.doi.org/10.1109/TII.2022.3165636.

Xiang, Y., Huang, H., Li, S., Li, M., Luo, C., Yang, X., 2023. Automated test suite generation for software product lines based on quality-diversity optimization. ACM Trans. Softw. Eng. Methodol. 33 (2), http://dx.doi.org/10.1145/1011453628158.

Xiang, Y., Huang, H., Li, M., Li, S., Yang, X., 2022a. Looking for novelty in search-based software product line testing. IEEE Trans. Softw. Eng. 48 (7), 2317–2338. http://dx.doi.org/10.1109/TSE.2021.3057853.

Xiang, Y., Huang, H., Zhou, Y., Li, S., Luo, C., Lin, Q., Li, M., Yang, X., 2022b. Search-based diverse sampling from real-world software product lines. In: Proceedings of the 2022 IEEE/ACM 44th International Conference on Software Engineering. ICSE, pp. 1945–1957.

Xu, G., Bao, Q., Zhang, H., 2023. Multi-objective green scheduling of integrated flexible job shop and automated guided vehicles. Eng. Appl. Artif. Intell. 126, 106864.

Yang, S., Huang, H., Luo, F., Xu, Y., Hao, Z., 2023. Local-diversity evaluation assignment strategy for decomposition-based multiobjective evolutionary algorithm. IEEE Trans. Syst. Man, Cybern.: Syst. 53 (3), 1697–1709.

Yao, Y., Liu, Q., Fu, L., Li, X., Yu, Y., Gao, L., Zhou, W., 2024. A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles. IEEE Trans. Autom. Sci. Eng. 1–14.

Yuan, Y., Xu, H., 2015. Multiobjective flexible job shop scheduling using memetic algorithms. IEEE Trans. Autom. Sci. Eng. 12 (1), 336–353.

Yuan, M., Zheng, L., Huang, H., Zhou, K., Pei, F., Gu, W., 2023. Research on flexible job shop scheduling problem with AGV using double DQN. J. Intell. Manuf. http://dx.doi.org/10.1007/s10845-023-02089-4.

Zhang, F., Li, R., Gong, W., 2024. Deep reinforcement learning-based memetic algorithm for energy-aware flexible job shop scheduling with multi-AGV. Comput. Ind. Eng. 189, 109917. http://dx.doi.org/10.1016/j.cie.2024.109917.

Zhang, S., Wang, S., 2018. Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. IEEE Trans. Eng. Manage. 65 (3), 487–504. http://dx.doi.org/10.1109/TEM.2017.2785774.

Zhang, Z., Wu, F., Qian, B., Hu, R., Wang, L., Jin, H., 2023. A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. Expert Syst. Appl. 234, 121050. http://dx.doi.org/10.1016/j.eswa.2023.121050.