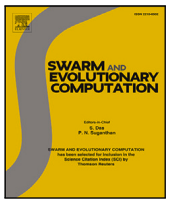
Swarm and Evolutionary Computation 93 (2025) 101849

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



# Enhancing Quality-Diversity algorithm by reinforcement learning for Flexible Job Shop Scheduling with transportation constraints

Haoxiang Qin [a], Yi Xiang [a,*], Fangqing Liu [b], Yuyan Han [c], Yuting Wang [c]

[a] School of Software Engineering, South China University of Technology, Guangzhou 510006, China
[b] School of Management, Guangdong University of Technology, Guangzhou 510520, China
[c] School of Computer Science, Liaocheng University, Liaocheng 252059, China

ARTICLE INFO

*Keywords:*
Flexible job shop scheduling
Transportation constraints
Quality-Diversity
Q-learning
Energy consumption

ABSTRACT

The Flexible Job Shop Scheduling Problem with transportation constraints (FJSP-T) widely exists in manufacturing industry. Despite its prevalence, relatively little research has been done on it. To address the FJSP-T, several meta-heuristic algorithms have been proposed, but they do not take into account the diversity of solution features, potentially leading the algorithm to get trapped in local optima. In addition, transportation constraints directly affect the final objective value, yet there are few efficient strategies specifically dedicated to this aspect of the issue. Moreover, when confronted with the challenge of selecting local search operators, the probabilistic randomization approach introduces considerable uncertainty. This method can engender a substantial volume of invalid searches, consequently degrading the efficiency of the algorithm. To address the above issues, we propose a reinforcement learning enhanced Quality-Diversity (QD) algorithm for FJSP-T. The QD framework, is employed to guarantee the diversity of solution features. This paper asserts that considering job transportation in local search operators can provide advantages in improving problem-solving abilities. Therefore, two local search operators based on job transportation are proposed. To make effective decisions between local search operators, a reinforcement learning method, named Q-learning, is applied. The results on 20 instances demonstrate that the superiority of the proposed algorithm, which has achieved an average reduction of 6% in the optimization objective, outperforming the best-performing algorithm. In summary, while exploiting characteristics of transportation constraints, enhancing QD with reinforcement learning provide a promising avenue to solve FJSP-T.

## 1. Introduction

Flexible Job Shop Scheduling Problem (FJSP) is an NP-hard combinatorial optimization problem, which widely exists in manufacturing industry [1,2]. In FJSP, each job consists of multiple operations that can be flexibly assigned to a set of machines for processing [3]. To efficiently carry out job processing, companies employ various equipments such as cranes, conveyors, and automated guided vehicles for job transportation [4]. It is imperative to acknowledge the profound interdependency between processing and transportation [5]. Processing completion dictates the initiation time for transportation, while transportation profoundly influences the start time of subsequent processing [6]. Hence, considering the characteristic of job transportation in FJSP (FJSP-T) can significantly enhance the practicality and effectiveness of solutions, making it worthy of investigation.

FJSP-T is a more complex problem than FJSP, as it requires addressing the job sequencing, machine allocation, and transportation, simultaneously. To date, numerous methods have been proposed to address the aforementioned problems. Among them, the number of algorithms for FJSP far exceeds those for FJSP-T, which are primarily categorized into mathematical methods [7–14] and meta-heuristic algorithms. Among them, the studies that address the FJSP are [15–25], while those focusing on the FJSP-T are [26–35]. The review work with FJSP can be found at [36–38]. Mathematical/accurate methods can in theory guarantee the quality of solutions, but are difficult to solve large-scale FJSP-T in acceptable time (due to the NP-hardness of FJSP-T) [6], Additionally, they have poor scalability [10]. Evolutionary Algorithms (EAs), as a type of meta-heuristic algorithm, have better scalability compared to mathematical methods [39]. They can obtain comparatively high-performing solutions within a limited time, regardless of whether the scale is small or large [3,4].

However, most EAs used to address the aforementioned problems have three issues: (1) They do not take into account the diversity of

* Corresponding author.
*E-mail addresses:* 987352978@qq.com (H. Qin), xiangyi@scut.edu.cn (Y. Xiang), peerfog@gdut.edu.cn (F. Liu), hanyuyan@lcu-cs.com (Y. Han), wangyuting@lcu-cs.com (Y. Wang).

https://doi.org/10.1016/j.swevo.2025.101849
Received 10 October 2024; Received in revised form 9 December 2024; Accepted 6 January 2025
Available online 23 January 2025
2210-6502/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

solutions in terms of features (the features of the solutions describe behaviors or characteristics when solving the problem) [40,41]; (2) The characteristics of transportation are often overlooked, that is, there are few local searches specifically designed for transportation [6,42]; (3) Their efficiency may be compromised when employing local search operators, and there is significant uncertainty in their approach to randomly selecting or choosing among multiple local search operators based on probability [39,43,44].

For the first issue, as in the literature [18], random solutions are used to maintain the diversity of solutions. In [27], similar colonies are replaced by new generated colonies. In [28], adaptive mutation is used to avoid the loss of population diversity. The above algorithms use different methods to enhance the diversity of solutions in decision space, not the feature space (set of problem features forms the feature space). Studies have demonstrated that maintaining diversity in the feature space of solutions is conducive to prevents solutions from falling into local optima, and greatly enhances the probability of discovering high-performing and diverse solutions [40,41,45]. Therefore, when dealing with the FJSP-T, it is meaningful to incorporate the diversity of solution features. For the second issue, some effective local search operators are proposed, but they are designed to change the job sequence directly rather than from transportation characteristics [28,31,32]. However, strategies designed for transportation characteristics are effective in reducing the objective value [6,42]. Therefore, it is necessary to design local search specifically for transportation. For the third issue, in [21], local and global searches are balanced by selection probability operators. The probability factor is used to ensure that every crossover and variation operator has a chance of being used [17]. Currently, there is evidence that utilizing probabilistic factors of randomness to make strategy decisions can sometimes be inefficient, which can lead to an increase in the number of invalid searches and weaken their overall performance effects [43,44]. In order to improve this situation, Reinforcement Learning (RL) methods have been increasingly integrated with various optimization algorithms for policy-making decisions, as evidenced by [2,6,39,43,44,46–48]. Within the domain of RL for tackling shop scheduling issues, Q-Learning emerges as a particularly effective method. For instance, in [39,43,44], experiments have demonstrated that the incorporation of Q-Learning significantly augments the local search ability of the algorithms. Building on these findings, this paper designs a Q-Learning method to make policy-decisions for better enhance the performance of the algorithm.

To address the above limitations, we propose to use an extended version of Q-Learning enhanced Quality-Diversity (QQD) algorithm [49] for solving the FJSP-T. The QD algorithm is a paradigm within the field of EAs [40]. Unlike traditional EAs, QD focus on exploring diverse and high-performing solutions in a feature space [45]. However, due to the diversity-driven nature of the original QD algorithms, they may exhibit relatively poor ability in local search [41]. To increase the local search ability of QD, we propose two local search operators based on transportation constraints. In addition, when designing specialized local search operators to avoid solutions from falling into local optima, a challenge arises: the algorithm finds it difficult to intelligently select a reasonable search operator in each iteration [46]. Therefore, there are three critical issues that must be addressed when applying the QD algorithm to the FJSP-T: (1) definition of the feature space; (2) enhancement of the local search ability based on transportation constraints; (3) rational utilization of local search operators.

For the first issue, we adopt one of the most renowned QD algorithms, MAP-Elites [41], to discretize the feature space into a grid, where each cell in the grid is populated with a high-performance solution. We utilize the number of job transportation and machine idle times as two features. The reason for choosing these two features is that the above two features have the most direct impact on the energy consumption, which is the objective of this paper. For the second issue, we propose two local search operators designed based on job transportation characteristics, combined with local search operators based on operation sequencing and machine selection, to enhance the local search ability of QD. For the third issue, we adopt the Q-Learning method to intelligently select appropriate local search operators. It can help the QD make better strategy decisions based on historical experience and the current environment, which in turn reduces the ineffective search of the algorithm. Up to now, QD has been successfully applied in various domains, including robotics [50], games [51], and software engineering [52]. Nonetheless, to the best of our knowledge, there is no research investigating the application of QD to address the discrete FJSP-T. This introduces a completely new perspective for solving FJSP-T and represents a highly meaningful research topic.

To validate the performance of the proposed QQD algorithm, we conducted extensive experiments on 20 instances. The results show that QQD reduces energy consumption by 6% on average compared to the best performing algorithm in the state-of-the-art algorithms. Statistical analysis tests and convergence curves show that QQD significantly outperforms the state-of-the-art algorithm in all instances.

In synthesis, the contribution of this paper is as follows:

- This paper provides a new perspective to solve the FJSP-T by combining QD with RL. Unlike existing EAs, QD takes into account the diversity of solutions in terms of feature space, which helps to avoid falling into local optima. For critical feature definitions, we customize a two-dimensional feature space with two dimensions of job transportation and machine idle times. The validity of this feature space definition is demonstrated both from conceptual and experimental aspects. In addition, by monitoring the number of updates in the feature space, we find that the number of updates through the neighborhood solutions is much higher than the number of its own updates. This suggests that finding solutions with more diversity in terms of features as much as possible helps the algorithm to evolve better.

- Two transportation-based search operators are proposed to improve the local search ability of the QD. It is experimentally demonstrated that the proposed two operators contribute the most to the enhancement of the local search ability. Specifically, the improvement in the quality of the overall solution set is accelerated by more efficiently changing the ordering of the operation sequences stored in the feature space. Similarly, for other related FJSP-T, efficient local search operators can be designed based on the transportation constraints.

- The QD framework introduces Q-Learning method, which allows for autonomous learning based on collected feedback and utilizes the knowledge to make automated decisions. Comparing to random selection, the efficiency of using a local search operator can be significantly improved by employing the Q-learning method. Q-Learning intelligently leverages these local search operators through intelligent selection, especially those designed based on transportation constraints. This method not only facilitates the algorithm's evolutionary trajectory towards more promising solutions but also effectively circumvents futile search.

The rest of this paper is organized as follows. Section 2 offers a comprehensive review of the related work on FJSP, QD algorithm and RL methods. Section 3 introduces the problem being addressed. Section 4 presents the proposed QQD. Section 5 consists of experiments and analysis. Finally, we conclude this work and outline potential future work in Section 6.

## 2. Related work

### 2.1. Flexible job shop scheduling

Methods for solving FJSP mainly can be classified into mathematical/accurate methods and meta-heuristic methods. The mathematical

methods mainly include constrained programming and mixed-integer linear programming [9–12]. While these methods can theoretically obtain optimal solutions, their computational time grows exponentially [43]. Meta-heuristic algorithms are more widely applied than mathematical methods, they can more effectively solve large-scale FJSP and related problems such as iterated greedy (IG) with simulated annealing (SA) [3], hybrid Variable Neighborhood Search (VNS) with Genetic Algorithm (GA) [16], and Memetic Algorithm (MA) [23]. For the FJSP, in [16], VNS algorithm based on GA (VNS-GA) was proposed to enhance the search capability with a balance of intensification and diversification. In this paper, the FJSP with the goal of minimizing makespan was considered. In [23], the new memetic algorithms were proposed to minimize the makespan, total workload, and critical workload. Later, MAs were incorporated with RL techniques to solve FJSPs. In [43], a RL-based parameter selection mechanism was proposed to make reasonable decisions. During the search process, Q-Learning was used to select the neighborhood structure, thereby increasing the local search ability of the algorithm. Experiments have shown that the above algorithm outperforms other algorithms that do not utilize the RL techniques. While the above studies addressed FJSP well, none of them considered job transportation in terms of this problem.

### 2.2. Flexible job shop scheduling with transportation constraints

To address the FJSP-T, several meta-heuristic algorithms have been proposed [26–32]. In [31,32], transportation constraints were introduced in FJSP. However, the main focus was on small and medium scale instances rather than the more realistic large scale instances. An IGSA algorithm was developed to minimize the makespan of FJSPs with crane transportation and the energy consumption during transportation [3]. For minimizing the makespan, A GA with tabu search method was presented to solve both assignment of resources and sequencing problems [26]. Due to the NP-hard nature of FJSP-T, an imperialistic competitive algorithm based on SA was suggested to solve this problem [27]. In [34], an efficient MA was presented to solve the distributed FJSP-T with the objectives of minimizing the makespan, maximum workload, and total energy consumption. Subsequently, RL methods have been introduced to collaborate with EAs to solve the FJSP-T. For instance, in [6], a Learning-based Multi-population Evolutionary Optimization algorithm was proposed. To tackle the strongly NP-hard FJSP-T, a Q-Learning-based hyper-heuristic EA was proposed. In this approach, Q-Learning is utilized to assist the algorithm in identifying the most suitable low-level heuristics [39]. The aforementioned algorithms have addressed the FJSP-T. However, most algorithms do not take into account the diversity of solutions in terms of the problem features. Furthermore, many studies overlook the transportation constraints, which can directly affect the calculation of the objective value. When using local search operators, the problem of their selection is predominantly still random. The approaches of random selection based on certain probabilities involves uncertainty. Up to now, there is relatively little research on using RL to assist meta-heuristic algorithms. Other traditional meta-heuristic algorithms may lack efficiency when employing different search strategies or heuristic methods.

### 2.3. Reinforcement learning and quality-diversity algorithms

Recently, RL has achieved many successful use cases in shop scheduling [43,47]. In [43], a RL-based parameter selection mechanism was proposed to make reasonable decisions. By using this mechanism, the diversity of solutions was improved. In [47], a neighborhood structure was introduced to explore the solution space in evolution. During the search process, RL was used to select the neighborhood structure, thereby increasing the local search ability of the algorithm. From the aforementioned studies, it is known that evolutionary algorithms can autonomously achieve a balance between exploration and exploitation by combining RL methods [47]. In fact, many of the current RL methods incorporated into evolutionary algorithms significantly help decision making. The objective is to equip the above algorithms with a wide array of appropriate local search operators during the exploration process, emphasizing enhancement of algorithm exploitation over exploration [44]. Furthermore, while the above algorithms have designed strategies to improve solution diversity, they commonly neglect the solution features. The features of one solution pertain to the behavior exhibited or the specific characteristics of constraints displayed within the problem. If an algorithm can leverage problem properties to formulate a corresponding strategy, such as seeking a multitude of solutions with diverse features, it may more effectively address the problem [40].

In recent years, the combination of RL and evolutionary algorithms has emerged as a promising research theme [6]. Q-Learning, as a model-free RL, has made many contributions to solving the shop scheduling problem [39,43,44]. For instance, a Q-Learning based parameter selection strategy was proposed to improve the diversity of non-dominated solutions [39]. In [43], Q-Learning was used to select suitable heuristics for the proposed algorithm. In addition, Q-Learning was employed to choose different local search strategies for different factories [44]. Empirical evidence highlights the high effectiveness of employing Q-Learning as a decision-making mechanism in algorithms designed to solve shop scheduling problems [47]. However, there has been relatively limited research on the application of Q-learning in FJSP-T.

Currently, there exist applications that integrate learning mechanism with QD. In [53], the policy gradient assisted MAP-Elites, which combined MAP-Elites with the gradient-based operator inspired by deep RL, was demonstrated the benefits to the performance of the algorithm. In [54], by integrating QD with RL, authors provided a valuable and effective approach to advancing insights into deep RL within the QD-RL framework, representing a significant method advancement in the field of QD-RL. However, the above study is limited to solving continuous optimization problems and is not suitable for discrete FJSP-T. Moreover, a QD benchmark suite was provided for deep neural evolution in the field of RL for robot control, and a modified version of the same metric was introduced to quantify the robustness of solutions in terms of environmental randomness [55]. Although these studies have demonstrated the effectiveness of combining machine learning mechanism with QD, they have mostly been applied to continuous optimization problems. So far, there is no application of QD to solve the FJSP-T.

## 3. Problem description

This section describes the FJSP-T and its encoding and decoding schema. Additionally, a brief Gantt chart example is given to further assist in better understanding the problem.

### 3.1. Problem formulation

Considering a collection of $m$ machines $M = \{1, \ldots, k, \ldots, m\}$. There are a set of $n$ jobs $I = \{1, \ldots, i, \ldots, n\}$, where each job $i$ contains a set of $w_i$ operations $J_i = \{1, \ldots, j, \ldots, w_i\}$. Each operation $O_{i,j}$ must be processed on any of the $M_{i,j}$. Different operations in the same job can be handled flexibly, rather than being fixed to a given machine. When a operation $O_{i,j}$ is transferred from machine $k$ to $k'$ for processing, its transportation time is $T_{i,j,k,k'}$. The processing time $T_{i,j,k}$ of $O_{i,j}$ is greater than 0.

It is assumed that all machines and jobs are available. Jobs cannot be interrupted once they start being processed. Each operation can only be processed by one machine and each machine can only process one operation at a time. The transportation time for each operation is determined only by the ending and beginning of the transportation, without taking into account constraints such as conflicts. Loading and unloading time is included in transportation time by default. In addition, once

an operation has begun to be transported or processed, it cannot be interrupted or preempted.

Inspired by [56], the following parameters and variables are used throughout this paper.

$i$: The index of job
$j$: The index of operation
$k$: The index of machine
$n$: 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
$m_{i,j}$: Number of available machines of $O_{i,j}$
$Q_k$: The total number of operations that processed on machine $k$
$I$: The set of jobs, $I = \{1, \dots, i, \dots, n\}$
$J_i$: The operation set of job $i$, $J_i = \{1, \dots, j, \dots, w_i\}$
$M$: The set of machines, $M = \{1, \dots, k, \dots, m\}$
$O_{i,j}$: The operation $j$ of job $i$
$M_{i,j}$: The set of selectable machines of $O_{i,j}$
$T_{i,j,k}$: Processing time of $O_{i,j}$ on machine $k$
$TR_{i,j,k,k'}$: Transportation time of $O_{i,j}$ from machine $k$ to $k'$
$C_{i,j}$: The completion time of $O_{i,j}$
$C_{i^-,j^-}$: Completion time of the preceding operation of $O_{i,j}$ on machine $k$
$C_{i,j,k}$: The completion time of $O_{i,j}$ on machine $k$
$B_{i,j}$: The start time of $O_{i,j}$
$B_{i,j,k}$: The start time of $O_{i,j}$ on machine $k$
$B_{k,q}$: Start processing time of the position $q$ on machine $k$
$X_{i,j,k}$: Binary variable, if operation $O_{i,j}$ is to be processed on machine $k$, it is set to 1; otherwise, it is set to 0
$Y_{i,j,k,q}$: Binary variable, if operation $O_{i,j}$ is to be processed on position $q$ of machine $k$, it is set to 1; otherwise, it is set to 0
$P_k$: The available time set of machine $k$
$U_{i,j}$: The machine that processes $O_{i,j}$
$TP$: Total processing EC
$TI$: Total machine idle EC
$TT$: Total transportation EC
$TP_{i,j,k}$: Processing EC of operation $O_{i,j}$ on machine $k$
$TI_k$: Machine idle EC on machine $k$
$TT_{i,j,k,k'}$: Transportation EC of operation $O_{i,j}$ from machine $k$ to $k'$
$PP_k$: The power of machine $k$ processing $O_{i,j}$ per unit time.
$PI_k$: The power of machine $k$ in idle state per unit time.
$PT_{i,j,k,k'}$: The power of transferring $O_{i,j}$ from machine $k$ to $k'$ per unit time.
$L$: A large number

The objective and constraints of FJSP-T can be formulated as follows:

$$Minimize\ obj = TP + TI + TT \tag{1}$$

$$TP = \sum_{i\in I}\sum_{j\in J_i}\sum_{k\in M_{i,j}} TP_{i,j,k} \tag{2}$$

$$TI = \sum_{k\in M} TI_k \tag{3}$$

$$TT = \sum_{i\in I}\sum_{j\in J_i}\sum_{k,k'\in M_{i,j}} TT_{i,j,k,k'} \tag{4}$$

$$TP_{i,j,k} = T_{i,j,k}\cdot PP_k \quad \forall i\in I,\ j\in J_i,\ k\in M_{i,j} \tag{5}$$

$$TI_k = \left(C_{i,j-1} - T_{i,j,k} + TR_{i,j,k,k'}\right)\cdot PI_k \quad \forall i\in I,\ j\in J_i,\ k,k'\in M_{i,j} \tag{6}$$

$$TT_{i,j,k,k'} = TR_{i,j,k,k'}\cdot PT_{i,j,k,k'} \quad \forall i\in I,\ j\in J_i,\ k,k'\in M_{i,j} \tag{7}$$

$$\sum_{k=1}^{m} B_{i,j,k}\cdot X_{i,j,k} = B_{i,j}, \quad \forall i\in I, j\in J_i \tag{8}$$

$$\sum_{k=1}^{m} T_{i,j,k}\cdot X_{i,j,k} = T_{i,j}, \quad \forall i\in I, j\in J_i \tag{9}$$

$$B_{i,j-1} + \sum_{k=1}^{m}\sum_{k\prime=1}^{m}\left(T_{i,j} + TR_{i,j-1,k,k\prime}\right)\cdot X_{i,j-1,k}\cdot X_{i,j,k} \leqslant B_{i,j} \quad \forall i\in I, j\in J_i \tag{10}$$

$$B_{i,j,k} \geqslant \max\left(C_{i,j-1,k\prime} + TR_{i,j-1,k\prime,k}, C_{i^-,j^-,k}\right) \quad \forall i, i^-\in I, j, j\prime\in J_i \text{ and } j\neq j\prime, k, w\in M \tag{11}$$

$$\sum_{k=1}^{m_{i,j}} X_{i,j,k} = 1, \quad \forall i\in I, j\in J_i \tag{12}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{J_i} Y_{i,j,k,q} = 1, \quad \forall k\in M, q\in Q_k \tag{13}$$

$$\sum_{q=1}^{Q_k} Y_{i,j,k,q} = X_{i,j,k}, \quad \forall i\in I, j\in J_i, k\in M \tag{14}$$

$$B_{k,q} \leqslant T_{i,j,k} + L\cdot\left(1 - Y_{i,j,k,q}\right) \quad \forall i\in I, j\in J_i, k\in M, q\in Q_k \tag{15}$$

$$B_{k,q} \geqslant T_{i,j,k} - L\cdot\left(1 - Y_{i,j,k,q}\right) \quad \forall i\in I, j\in J_i, k\in M, q\in Q_k \tag{16}$$

where Eq. (1) is the optimization objective: energy consumption. Eqs. (2)–(7) indicates the energy consumption of processing, machine idle, and transportation. Constraints (8) and (9) ensure the start time and processing time of operation $O_{i,j}$. Constraint (10) denotes that operation $O_{i,j}$ must be processed after its proceeding operation $O_{i,j-1}$. Constraint (11) represents a relationship between completion time and start time between adjacent operations on the same machine $k$. Constraint (12) ensures that only one machine can be selected for an operation $O_{i,j}$. Constraint (13) guarantees that only one $O_{i,j}$ is assigned to a position $q$ on the machine $k$. Constraint (14) suggests a link between $X_{i,j,k}$ and $Y_{i,j,k,q}$. Constraints (15) and (16) ensure the association between $B_{k,q}$ and $Y_{i,j,k,q}$.

Fig. 1 displays a comparative Gantt chart example when transporting operation $O_{1,2}$ to different machines. In the processing plant, there are a total of 3 machines. The number of operations is 4. Each operation has 3 operations to be processed. Different jobs are represented by different colors, with light blue indicating the transport time when the job is transferred from one machine to another. The length of each rectangle represents the time consumed by the job processing and transportation. We assume that $OS = \{O_{1,1}, O_{3,1}, O_{2,1}, O_{4,1}, O_{3,2}, O_{1,2}, O_{4,2}, O_{3,3}, O_{2,2}, O_{4,3}, O_{2,3}, O_{1,3}\}$. Assuming that the transportation energy consumption per unit time for a process is 1 kW h, the processing energy consumption per unit time is 5 kW h, and the machine idle energy consumption per unit time is 0.5 kW h. According to Fig. 1(a), the transportation energy consumption $TT$ is calculated as: $1 \times 9 = 9$ kW h. The processing energy consumption $TP$ is: $5 \times 33 = 165$ kW h. The idle energy consumption $TI$ is: $0.5 \times 10 = 5$ kW h. Therefore, the total energy consumption $TEC$ is $9 + 165 + 5 = 179$ kW h.

However, the solution can be optimized by transporting the operation to a different processing machine. For example, as shown in Fig. 1(b), for $O_{1,2}$, the total energy consumption can be effectively reduced by transporting it from *M3* to *M2*. Specifically, the energy consumption for transportation is reduced from 9 kW h to 7 kW h, and the idle energy consumption of the machine is reduced from 5 kW h to 3.5 kW h. Thus, the TEC is reduced by 3.5 kW h. If further optimization is performed, a solution with even lower TEC can be found.

### 3.2. Encoding and decoding

The encoding and decoding for FJSP-T are described in this section. As illustrated in Fig. 2, the integer encoding schema is adopted [48]. It contains two vectors: (1) Operation Sequence (OS); (2) Machine Selection (MS). The length of OS and MS equals to the total number

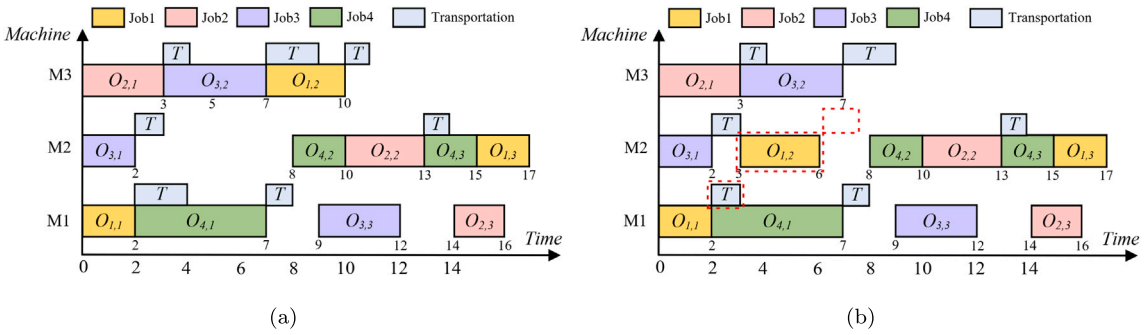



**Fig. 1.** Gantt chart of the FJSP-T. (a) original scheduling solution; (b) the new scheduling solution after transporting operations $O_{1,2}$ from machine *M3* to *M2*.

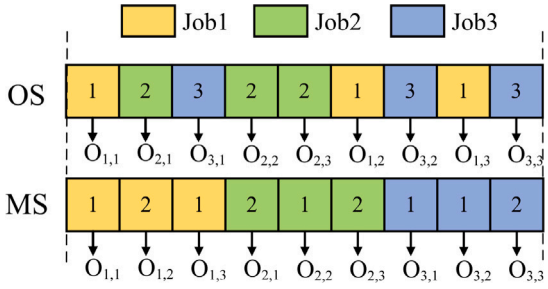



**Fig. 2.** The integer encoding vectors of a solution.

of operations $n_{max}$. The numbers on OS indicate the same job. The numbers on MS represent the selected machine for the operation.

As shown in Algorithm 1, the decoding procedure *Decoding*$(x)$ is designed to evaluate one solution. The input $x$ indicates a solution which includes OS and MS vectors. $obj_x$ and $\boldsymbol{b}_x$ are two outputs, while the former is the objective value of the solution $x$ and the latter is coordinates of $x$ in feature space. The coordinates are determined by two features *num_idle* and *num_trans*, while the former represents the number of machine idle times and the latter represents the number of transportation times of all jobs. Both *num_idle* and *num_trans* form the grid. They are integers, which are ranging from 0 to $n_{max}$.

Before describing *Decoding*$(x)$, there are some important formulas that need to be introduced. It should be noted that the values of $T_{i,j,U_{i,j}}$ and $TR_{i,j,U_{i,j-1},U_{i,j}}$ are known in advance. During initialization, the values of both the $U_{i,j}$ and $C_{i,j}$ sets are 0. $P_{U_{i,j}}$ indicates the available time of machine $U_{i,j}$. If $P_{U_{i,j}} < C_{i,j-1}$, $U_{i,j}$ is in idle state. Eq. (17) is used to calculate $P_{U_{i,j}}$.

$$P_{U_{i,j}} = C_{i,j-1} + TR_{i,j,U_{i,j-1},U_{i,j}} + T_{i,j,U_{i,j}} \tag{17}$$

If $P_{U_{i,j}} \geqslant C_{i,j-1}$, $U_{i,j}$ is not in idle state. When $j = 1$, Eq. (18) is employed to calculate $P_{U_{i,j}}$; Otherwise, Eq. (17) or (18) is adopted to calculate $P_{U_{i,j}}$ (Lines 18–24).

$$P_{U_{i,j}} = P_{U_{i,j}} + T_{i,j,U_{i,j}} \tag{18}$$

As shown in Algorithm 1, Lines 4–5 find the selected machine for operation $O_{i,j}$ from MS vector. Lines 7–13 outline the calculation for $TP_{i,j,U_{i,j}}$, $TI_{U_{i,j}}$, and $TT_{i,j,U_{i,j-1},U_{i,j}}$, and count the number of *num_idle* and *num_trans* when machine $U_{i,j}$ is in an idle state. Lines 15–31 describe the calculation for $TP_{i,j,U_{i,j}}$, $TI_{U_{i,j}}$, and $TT_{i,j,U_{i,j-1},U_{i,j}}$, and count the number of *num_idle* and *num_trans* when $U_{i,j}$ is not in an idle state. In Line 34, the total EC of FJSP-T, i.e., $obj_x$, are calculated. Line 35 obtains the coordinates $\boldsymbol{b}_x$ of two features *num_idle* and *num_trans*.

**Algorithm 1** Decoding $(x)$

**Input:** solution $x$
**Output:** The objective value $obj_x$ of $x$ and its corresponding coordinates in feature space $\boldsymbol{b}_x$

1: $x$ can be represented by a combination of OS *and* MS
2: $num\_idle \leftarrow 0$, $num\_trans \leftarrow 0$
3: **for** $pos \leftarrow 1$ **to** $n_{max}$ **do**
4: $O_{i,j} \leftarrow OS_{pos}$
5: Find $U_{i,j}$ from $MS$ *according to the position of* $O_{i,j}$
6: **if** $P_{U_{i,j}} < C_{i,j-1}$ **then**
// Machine $U_{i,j}$ is idle
7: $num\_idle \leftarrow num\_idle + 1$
8: Calculate the $P_{U_{i,j}}$ using Eq.(17)
9: $C_{i,j} \leftarrow P_{U_{i,j}}$
10: **if** $U_{i,j-1}$ ! $= U_{i,j}$ **then**
11: $num\_trans \leftarrow num\_trans + 1$
12: **end if**
13: Calculate $TP_{i,j,U_{i,j}}$, $TI_{U_{i,j}}$, and $TT_{i,j,U_{i,j-1},U_{i,j}}$ using Eqs.(5-7), respectively.
14: **else**
// Machine $U_{i,j}$ is not idle
15: **if** $j == 1$ **then**
16: Calculate $P_{U_{i,j}}$ using Eq.(18)
17: **else**
18: **if** $P_{U_{i,j}} < C_{i,j-1} + TR_{i,j,U_{i,j-1},U_{i,j}}$ **then**
19: Calculate $TI_{U_{i,j}}$ using Eq.(6)
20: $num\_idle \leftarrow num\_idle + 1$
21: Calculate $P_{U_{i,j}}$ using Eq.(17)
22: **else**
23: Calculate $P_{U_{i,j}}$ using Eq.(18)
24: **end if**
25: **if** $U_{i,j-1}$ ! $= U_{i,j}$ **then**
26: $num\_trans \leftarrow num\_trans + 1$
27: **end if**
28: Calculate $TT_{i,j,U_{i,j-1},U_{i,j}}$ using Eq.(7)
29: **end if**
30: $C_{i,j} \leftarrow P_{U_{i,j}}$
31: Calculate $TP_{i,j,U_{i,j}}$ using Eq.(5)
32: **end if**
33: **end for**
34: Calculate $TP, TI$, and $TT$ using Eqs.(2-4), respectively.
35: $obj_x \leftarrow TP + TI + TT$
36: $\boldsymbol{b}_x \leftarrow (num\_idle, num\_trans)$
**return** $\{obj_x, \boldsymbol{b}_x\}$

**Algorithm 2** Q-Learning enhanced QD Algorithm

**Input:** $N$, $batch$, $Max\_Iter$, $\alpha$, $\gamma$, $\epsilon$
**Output:** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)
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 $(\mathcal{P}, \mathcal{X}, x)$ *// Algorithm 3*
5: **end for**
6: $iter \leftarrow 1$, $s_1 \leftarrow 1$
7: Initialize $Q$ table
8: **while** $iter \leqslant Max\_Iter$ **do**
9: **for** $t \leftarrow 1$ **to** *batch* **do**
10: Randomly select one solution $x$ from $\mathcal{X}$
11: Select $a_t$ using Eq.19
12: $\epsilon \leftarrow \epsilon \cdot 0.999$
13: $x' \leftarrow$ Execute LSS for $x$ according to $a_t$
14: Add_to_grid $(\mathcal{P}, \mathcal{X}, x')$ *// Algorithm 3*
15: **if** $t = batch$ **then**
16: $s_{t+1} \leftarrow 0$
17: **else**
18: $s_{t+1} \leftarrow s_t + 1$
19: **end if**
20: Update $Q(s_t, a_t)$ using Eq.20
21: Update $\alpha$ using Eq.21
22: **end for**
23: **end while**
**return** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)

## 4. Q-Learning enhanced Quality-Diversity algorithm

In this section, we provide a detailed description of the proposed QQD algorithm, which is designed to address the challenges outlined in the previous section regarding the FJSP-T. The algorithm consists of several key components: a novel framework, a Q-learning-based decision-making method, a customized fitness function for evaluating solutions, and an efficient local search strategy to improve solution quality. Each of these elements plays an important role in enhancing the algorithm's performance. The following subsections explain their implementation and integration within the QQD algorithm.

### 4.1. Framework of QQD

Algorithm 2 shows the framework of QQD. The input parameters contains: (1) the dimension of grid $N$; (2) the batch size $batch$; (3) the maximum number of evaluations $Max\_Iter$. (4) learning rate $\alpha$; (5) discount factor $\gamma$; and (6) greedy factor $\epsilon$. The output returns feature-performance grid with $\mathcal{X}$ and $\mathcal{P}$. $\mathcal{X}$ represents the solution set in the grid and $\mathcal{P}$ indicates the corresponding objective values. The grid is composed of two features: $num\_idle$ and $num\_trans$ ($N = 2$). The reason we choose $num\_idle$ and $num\_trans$ is that the above two features will directly affect EC of the process. Whether machines are idle or jobs are transported, these machines will generate excess energy. Hence, striking a equilibrium between the number of machine idle times and the number of job transfers becomes crucial. This is why we opt to incorporate $num\_idle$ and $num\_trans$ as features in the feature space.

The subsequent step involves a gradual explanation of the algorithmic framework. Lines 3–4 list the initialization operations for QQD. $batch$ solutions are generated randomly. Whenever a new solution $x$ is generated, it is transferred to $Add\_to\_Grid\,(\mathcal{P}, \mathcal{X}, x)$ (Algorithm 3) to evaluate. It should be noted that the reward value is not involved in the calculation in initialization stage. In Lines 6–7, $Q$ table is initialized and its values default to 0. The initial value of state $s_1$ is 1, indicating the first solution. The value of state in $Q$ table indicates the number of one solution. In Line 10, a solution is randomly selected from the grid.

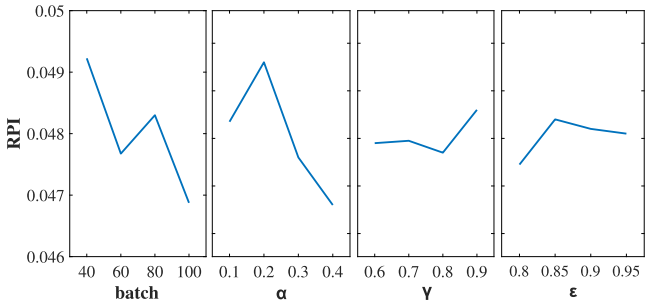



**Fig. 3.** Impacts of all the parameters on RPL.

### 4.2. Q-Learning method

Q-Learning is employed from Line 11 for strategy decision-making. Line 11 depicts the selection of action $a_t$ by using Eq. (19): if a random value $\tau$ is greater than or equal to greedy factor $\epsilon$, action $a_t$ is chosen which has the max $Q$ values. If $\tau$ is less than $\epsilon$, $a_t$ is chosen randomly among all actions. The goal of using this equation is to maintain a balance between exploitation and exploration. In Line 12, $\epsilon$ is gradually decreasing. During the initial iterations, the $Q$ table only acquires limited useful information, which can result in a risk of evolutionary stagnation. Therefore, it is recommended to set a relatively high value for $\epsilon$. By randomly selecting actions for trial and error, the $Q$ table accumulates more valuable information. As the iterations progress, the influence of random selection gradually diminishes, allowing the algorithm to make decisions based on the learned information from the $Q$ table. Lines 13–14 execute a local search operator based on the action $a_t$ and return a new solution $x'$. Subsequently, the new solution is put into the grid for evaluation. In Lines 15–19, the next state $s_t$, i.e., the number of solution is updated. Line 20 updates the value of $Q(s_t, a_t)$ by using Eq. (20). In Eq. (20), the learning rate $\alpha$ is gradually decreased by using Eq. (21) (Line 21). The value of $\alpha$ decides the ability to leverage existing knowledge and learn new knowledge. In the early stages of Q-Learning, the solutions obtained may not have high quality. Setting a higher learning rate is beneficial for expanding the search area and conducting exploration in a wide feature space. As the learning process advances, the solution found will gradually approach the currently determined optimal solution. Using a lower value of $\alpha$ is more conducive to fine-tuning the solution based on the available feature information. In addition, the discount factor $\gamma$ determines the influence of future returns on the current state update. A higher value of $\gamma$ implies that the current state is updated with greater consideration for future rewards, emphasizing the importance of rewards in the long-term decision-making process.

$$a_t = \begin{cases} arg\max\limits_{a \in A} Q\left(s_t, a\right), \ \tau \geqslant \varepsilon \\ a_{random}, \ \tau < \varepsilon \end{cases} \tag{19}$$

where $a_t$ represents one action, it determines which strategy is adopted. $Q\left(s_t, a\right)$ indicates a certain $Q$ value in $Q$ table at state $s_t$. $s_t$ is the current state, in this paper, a state is defined as one solution, $t$ is the specific number of the solution. By treating the solution as a state, the algorithm is able to explore different regions of the feature space, and the local search operator can target these regions in more detail, thus improving the algorithm's global search and local exploitation abilities [57]. $A$ represents the set of actions. We denote different actions as various local search operators. $arg\max_{a \in A} Q\left(s_t, a\right)$ means to select the action with the largest value among $Q$ values of all actions at state $s_t$. $a_{random}$ means to randomly select an action. $\epsilon$ is a greedy factor and $\tau$ is a random value ranging from [0,1].

$$Q\left(s_t, a_t\right) = Q\left(s_t, a_t\right) + \alpha \cdot \left[r_t + \gamma \cdot arg\max_{a \in A} Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right] \tag{20}$$

where $Q(s_t, a_t)$ represents the $Q$ value that execute action $a_t$ at state $s_t$. $\alpha$ is the learning rate. $\gamma$ is the discount factor. $r_t$ is the reward after executing action $a_t$. $arg\max_{a \in A} Q(s_{t+1}, a)$ represents the max $Q$ value of the next state $s_{t+1}$.

$$\alpha = \alpha - (\alpha - 0.01) \cdot iter / Max\_Iter \tag{21}$$

where $\alpha$ is the learning rate. $iter$ is the current number of iteration. $Max\_Iter$ is the total number of iterations.

### 4.3. Fitness function

The goal of Algorithm 3 is to compute the value of the objective function for the new solution and update the grid of the feature space. In Algorithm 3, $\mathcal{X}(\boldsymbol{b}_x)$ and $\mathcal{P}(\boldsymbol{b}_x)$ represent the solution in cell $\boldsymbol{b}_x$ and the corresponding objective value. The feedback signal $r_t$ represents the reward, which is used to evaluate the effectiveness of taking actions in each state. Line 1 depicts the calculation of $\boldsymbol{b_x}$ and its $obj_x$. In Lines 3–5, if $\boldsymbol{b_x}$ is empty, the new solution $x$ is directly put into the cell and reward $r_t$ is 1; If $\boldsymbol{b_x}$ is not empty, $x$ is greater than the old solution, it replaces the old one and $r_t$ is computed based on line 9; Otherwise, $x$ is discarded and the $r_t$ is 0.

**Algorithm 3** Add_to_Grid $(\mathcal{P}, \mathcal{X}, x)$

**Input:** $\mathcal{P}, \mathcal{X}, x$
**Output:** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)
1: $\{obj_x, \boldsymbol{b}_x\} \leftarrow Decoding(x)$ // *Algorithm* 1
2: **if** $\mathcal{P}(\boldsymbol{b}_x) = \emptyset$ **then**
3: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow obj_x$
4: $\mathcal{X}(\boldsymbol{b}_x) \leftarrow x$
5: $r_t \leftarrow 1$
6: **else if** $\mathcal{P}(\boldsymbol{b}_x) > obj_x$ **then**
7: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow obj_x$
8: $\mathcal{X}(\boldsymbol{b}_x) \leftarrow x$
9: $r_t \leftarrow (\mathcal{P}(\boldsymbol{b}_x) - obj_x)/\mathcal{P}(\boldsymbol{b}_x)$
10: **else**
11: $r_t \leftarrow 0$
12: **end if**
**return** feature-performance grid ($\mathcal{P}$ *and* $\mathcal{X}$)

### 4.4. Local search strategy

In this subsection, we describe six local search operators in LSS (Line 13, Algorithm 2). The input action $a_t$ is utilized to select the corresponding local search operator for the solution $x$, which in turn yields $x'$. Among them, LS1 is formulated in accordance with the machine utilization rate, while LS2-3 take into account the transportation constraints. Furthermore, LS4-6 are utilized to change the positions of critical operations in OS and MS.

(1) *LS1:* See neighborhood structure ***Re*** in [16], to increase the utilization of machines, *LS1* is used to move operations processed on machines with larger loads to machines with the smallest loads. At the beginning of the program, find the machine with the smallest load *Minload*, and traverse all the operations on other machines, and if they can be processed on *Minload*, move them to this machine.

(2) *LS2:* In order to reduce EC during transportation, *LS2* is employed to change the machine selection for the operations in a given job that have the longest transportation time. Randomly select a job in the scheduling sequence and find the operation with the longest transportation time. Subsequently, change the selected machine (different from the original machine) for this operation.

(3) *LS3:* To further reduce the impact of transportation time on the objective value, *LS3* is used to change the machine for the operation with the longest transportation time on the critical path. Before giving this operator, we first introduce the concept of a critical path and critical operations. The critical path indicates the path with the longest duration, while the operations on this path are known as critical operations [48,58]. By reducing the duration of critical path, it is possible to increase the utilization of machines (reduce EC of idle machines and transportation) or reduce the processing EC of operations. First, LS3 finds all critical operations. Subsequently, the operation with the longest transportation time is found. Finally, the selected machine for this operation is changed.

(4) *LS4:* Randomly change the sequence of two critical operations to reduce EC. Find all the key operations at first. Then randomly select two critical operations (operations are not in the same job). Swap the positions of these two operations.

(5) *LS5:* It is possible to increase machine utilization by changing the allocation of machines for critical operations. Find all the critical operations and later change the machine for a certain critical operation.

(6) *LS6:* Randomly select a critical operation and an arbitrary operation. Afterwards, find their position in the OS. Immediately thereafter, insert the later operation in front of the earlier operation.

## 5. Experiments

In this section, to test the performance of QQD, we designed parameter calibration, ablation and comparison experiments.

### 5.1. Experimental settings

Following [43], the number of jobs is recognized as $i \in \{20, 30, 40, 50, 100\}$. The number of machines is regarded as $m \in \{5, 6, 7, 8\}$. There are 5*4 = 20 different combinations in total. Each instance is repeated for 20 independent tests. Thus, a total of 400 tests are performed for each algorithm. For each job, the range of processing time $T_{i,j,k}$ for its operations is $[5, 20]$. We refer to the test set from [59] to set the transportation time for all jobs. In addition, the maximum number of evaluations for all tests in this section is set to $Max\_Iter = 20 \cdot n_{max} \cdot m$. The experiments were performed on Windows 11 operating system with an Intel Core i7 @ 2.10 GHz CPU and 16.0 GB RAM. See [44,60], we use Relative Percentage Increment (RPI) to evaluate the quality of an algorithm. This indicator indicates the difference between the objective value obtained by the current algorithm and the best objective value obtained by all algorithms. It is defined as follows:

$$RPI_a = (c_a - c_{best}) / c_{best} \times 100\% \tag{22}$$

where $c_a$ represents EC obtained from algorithm $a$, and $c_{best}$ is the minimum EC of all algorithms. A lower $RPI_a$ indicates better algorithm performance.

### 5.2. Parameters calibration

We use Taguchi approach [48,61] to test the following parameters: batch size $batch$, learning rate $\alpha$, discount factor $\gamma$, and greedy factor $\epsilon$. The range of values for parameters is shown as follows: $batch \in \{40, 60, 80, 100\}$, $\alpha \in \{0.1, 0.2, 0.3, 0.4\}$, $\gamma \in \{0.6, 0.7, 0.8, 0.9\}$, and $\epsilon \in \{0.8, 0.85, 0.9, 0.95\}$. An orthogonal array that contains 16 $(batch, \alpha, \gamma, \epsilon)$ combinations of parameters were tested. Fig. 3 presents the main effect plots for four parameters of RPI.

When the $batch$ size is equal to 100, the performance is best. This may suggest that with a larger number of solutions in the feature space at initialization, the diversity of solutions is enhanced, and consequently, the speed at which the algorithm identifies high-quality solutions during iteration is also increased. A higher learning rate may result in better performance because it allows the algorithm to learn new information more quickly, explore and exploit new action strategies more effectively, thereby achieving higher rewards. A discount factor of 0.8 likely offers a balanced approach, weighing immediate rewards while also appropriately valuing future ones. At 0.7, the future rewards might be underweighted, causing the algorithm to focus excessively on short-term gains. Conversely, at 0.9, the excessive weight on

**Table 1**
The results of all variants. Bold font represents the best value.

| Instance | RPI | | | | MEAN | | | | BEST | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QQD | V-QD | V-Q-learning | V-LS | QQD | V-QD | V-Q-learning | V-LS | QQD | V-QD | V-Q-learning | V-LS |
| 20J5M | **0.04** | 0.20[a] | 0.21[a] | 0.18[a] | **4720.65** | 5462.8 | 5507.15 | 5330.25 | **4535** | 5325 | 5409 | 5172 |
| 20J6M | **0.04** | 0.21[a] | 0.24[a] | 0.19[a] | **4639.1** | 5383 | 5523.85 | 5299.65 | **4445** | 5185 | 5401 | 5159 |
| 20J7M | **0.04** | 0.25[a] | 0.27[a] | 0.22[a] | **4416.85** | 5302.15 | 5366 | 5148.6 | **4230** | 5207 | 5171 | 4857 |
| 20J8M | **0.03** | 0.24[a] | 0.27[a] | 0.22[a] | **4367.35** | 5259.7 | 5409 | 5188.55 | **4257** | 5069 | 5228 | 5008 |
| 30J5M | **0.04** | 0.18[a] | 0.20[a] | 0.16[a] | **7309.95** | 8351.35 | 8448 | 8204.1 | **7060** | 8203 | 8273 | 8012 |
| 30J6M | **0.03** | 0.20[a] | 0.22[a] | 0.18[a] | **7113.3** | 8306.95 | 8406.65 | 8132.55 | **6896** | 8113 | 8223 | 7869 |
| 30J7M | **0.03** | 0.22[a] | 0.23[a] | 0.19[a] | **6910.5** | 8119.25 | 8223.25 | 7957.4 | **6679** | 7959 | 7913 | 7786 |
| 30J8M | **0.03** | 0.24[a] | 0.26[a] | 0.21[a] | **6649.1** | 8004.75 | 8127.85 | 7828.8 | **6455** | 7857 | 7787 | 7531 |
| 40J5M | **0.04** | 0.16[a] | 0.17[a] | 0.16[a] | **9720.3** | 10 844 | 10 993.7 | 10 818.9 | **9385** | 10 570 | 10 741 | 10 620 |
| 40J6M | **0.02** | 0.17[a] | 0.19[a] | 0.15[a] | **9428.5** | 10 776.85 | 10 933.05 | 10 585.65 | **9199** | 10 413 | 10 676 | 10 431 |
| 40J7M | **0.05** | 0.20[a] | 0.23[a] | 0.18[a] | **9154.3** | 10 532.35 | 10 724.45 | 10 312.45 | **8751** | 10 093 | 10 406 | 9971 |
| 40J8M | **0.04** | 0.23[a] | 0.26[a] | 0.22[a] | **8893.05** | 10 452.5 | 10 707.95 | 10 372.1 | **8520** | 10 207 | 10 439 | 10 192 |
| 50J5M | **0.04** | 0.17[a] | 0.18[a] | 0.15[a] | **12 226.2** | 13 650.9 | 13 772.95 | 13 514.95 | **11 705** | 13 371 | 13 475 | 13 216 |
| 50J6M | **0.03** | 0.16[a] | 0.18[a] | 0.15[a] | **11 953.1** | 13 499.15 | 13 708.8 | 13 357.1 | **11 635** | 13 152 | 13 458 | 12 993 |
| 50J7M | **0.04** | 0.19[a] | 0.21[a] | 0.18[a] | **11 558.4** | 13 244.25 | 13 419.2 | 13 071.85 | **11 120** | 13 024 | 13 069 | 12 904 |
| 50J8M | **0.03** | 0.20[a] | 0.22[a] | 0.18[a] | **11 172.45** | 13 035.8 | 13 229.65 | 12 803.25 | **10 863** | 12 781 | 12 845 | 12 523 |
| 100J5M | **0.02** | 0.11[a] | 0.11[a] | 0.09[a] | **25 519.75** | 27 578.55 | 27 711.4 | 27 298.25 | **24 937** | 27 179 | 27 384 | 26 923 |
| 100J6M | **0.04** | 0.13[a] | 0.14[a] | 0.12[a] | **24 784.7** | 27 057.55 | 27 199.35 | 26 679.6 | **23 914** | 26 646 | 26 697 | 26 243 |
| 100J7M | **0.03** | 0.16[a] | 0.17[a] | 0.14[a] | **23 399.1** | 26 191.2 | 26 543.65 | 25 895.55 | **22 653** | 25 708 | 25 851 | 25 366 |
| 100J8M | **0.03** | 0.19[a] | 0.21[a] | 0.18[a] | **22 452.65** | 25 871.3 | 26 297.15 | 25 516.9 | **21 699** | 24 939 | 25 768 | 25 045 |
| AVG | **0.04** | 0.19 | 0.21 | 0.17 | **11 319.47** | 12 846.22 | 13 012.65 | 12 665.82 | **10 946.90** | 12 550.05 | 12 710.70 | 12 391.05 |

[a] Indicates that the QQD is significantly different from other variants.

**Table 2**
The Friedman's test for significant differences among variants (confidence level $\alpha = 0.05$).

| Variants | RPI | |
|---|---|---|
| | Rank | *p*-value |
| QQD | 1.0 | 7.86E−13 |
| V-QD | 3.0 | |
| V-Q-learning | 3.975 | |
| V-LS | 2.025 | |

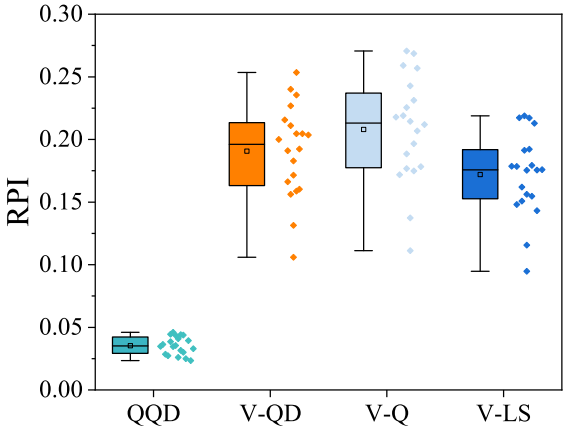



**Fig. 4.** RPI values of QQD and its variants, shown in the form of box plot.

future rewards could make the algorithm overly conservative, neglecting potential immediate benefits. A lower $\epsilon$ value may imply that the algorithm is more inclined to select the currently optimal action, while a higher $\epsilon$ value may lead the algorithm to engage in excessive random exploration, potentially introducing unnecessary action selections and thereby reducing learning efficiency and performance. Based on the plots of all parameters in Fig. 3, we finally set: $batch = 100$, $\alpha = 0.4$, $\gamma = 0.8$, and $\epsilon = 0.8$.

### 5.3. Ablation experiment

To evaluate the effectiveness of strategies, we compared QQD with three variants that are: (1) V-QD: variant without QD framework. (2) V-Q: variant without Q-Learning method. (3) V-LS: variant without local search strategy. All variants are tested on all instances. Table 1 presents the performance of all variants under the metrics of RPI, Mean and Best Objective Values. Additionally, we conducted a Wilcoxon test on the values obtained for the RPI metric, with a 0.05 confidence interval, to evaluate whether there is a statistically significant difference between all variants and the proposed QQD algorithm. † indicates that the variant is significantly different from QQD. The results in Table 1 show that the proposed algorithm is better than the other variants in all the instances. Moreover, the results of Wilcoxon test show that QQD is superior to other variants in terms of significance. To further evaluate the overall performance of all algorithms, we employed the Friedman test. As depicted in Table 2, the proposed QQD algorithm is ranked first, indicating superior performance. The *p*-value is significantly lower than the predetermined significance level of 0.05, leading to the rejection of the null hypothesis and indicating that there are statistically significant differences among the algorithms.

To further validate the differences between the proposed algorithm and its variants, we conducted a statistical experiment. Specifically, we evaluated the proportion of solutions stored by QQD in the feature space that outperformed the best solutions of its variants (one by one comparison). This experiment not only validates that QQD can obtain high-performance solutions, but also verifies its property of obtaining a diverse set of solutions in the feature space. The experiments were categorized into small, medium, and large-scale instances. Small-scale instances include 20 and 30 jobs, medium-scale instances include 40 and 50 jobs, and large-scale instances include 100 jobs. As shown in Fig. 5, $'l'$ represents large-scale, $'m'$ represents medium-scale, and $'s'$ represents small-scale instances. $'\uparrow'$ indicates the proportion of solutions obtained by QQD in the feature space that outperform the best solutions of the variants in small, medium, and large-scale instances. $'\uparrow'$ indicates the proportion of solutions that are worse than the best solutions of the variants, and $'\sim'$ represents the proportion of solutions approximately equal to the best solutions of the variants. Compared to v-QD, approximately 70% of the solutions stored by QQD in the feature space are superior, while 30% are inferior. For v-Q, 72.9% of the solutions stored by QQD outperform its variant, with 27.1% being inferior. Similarly, against v-LS, 67% of the solutions stored by QQD are superior, and 33% are inferior. The results highlight the

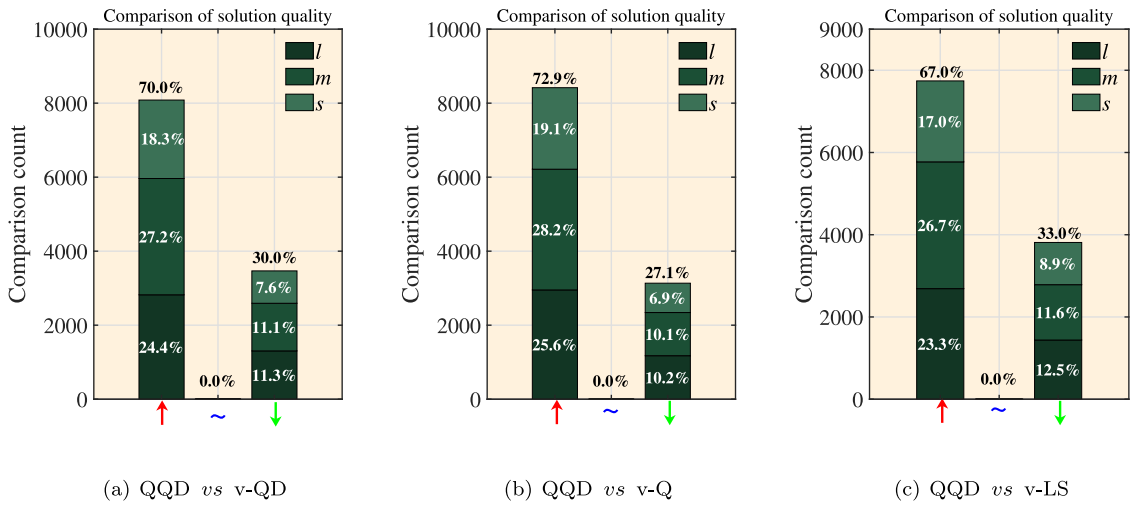



**Fig. 5.** Comparison of solution quality between QQD and other algorithms on small, medium, and large instances.

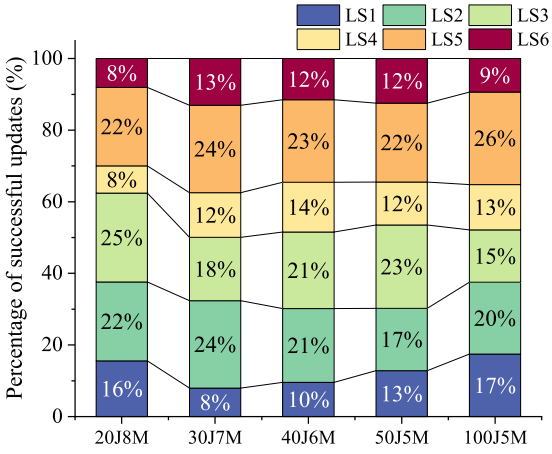



**Fig. 6.** Percentage of successful updates for all local search operators in five instances.

effectiveness of the QD optimization framework in generating a diverse set of high-quality solutions across all problem scales. By leveraging the QD framework, QQD efficiently explores the feature space, capturing varied features to enhance both solution quality and diversity. Additionally, the integration of Q-learning-based selection mechanism and transportation-based local search strategies significantly improves performance. Q-learning dynamically selects optimal search operators, aiding in escaping local optima, while the AGV-based local search reduces transportation and idle energy consumption, further enhancing solution efficiency.

Additionally, to provide a more intuitive observation of the differences between the various variants, such as the distribution of solutions, we plotted the box plot of RPI values. As shown in Fig. 4, the horizontal line in each box indicates the median value and the square blocks indicate the mean value. The RPI values obtained by QQD in 20 instances are significantly better than the other variants. In addition, to verify the utilization and usefulness of all local search operators under the Q-Learning option, especially the proposed operators (LS2 and LS3), we randomly selected five test cases of different sizes to test the percentage of successfully updated solutions using a specific search operator against all successfully updated solutions. As can be seen from Fig. 6, all local search operators contribute to the solution improvement. Of them, the top three contributors are LS2, LS3 and LS5, respectively. This demonstrates that by adjusting the job transportation and improving machine utilization, EC of FJSP-T can be reduced effectively.

To analyze the selection of local search operators by the Q-learning method, we plotted the proportions of different operators chosen during the early, middle, and late stages of evolution. Two random instances (30J7M and 50J5M) were carefully chosen for this analysis. As shown in Fig. 7, the selection proportion for LS2-3 steadily increased over time and consistently surpassed the selection rates of other operators. This trend confirms the effectiveness of the proposed LS strategies that incorporate transportation constraints, aligning closely with the conclusions drawn from Fig. 6. The consistent preference for LS2-3 demonstrates the Q-learning method's ability to adaptively identify and prioritize effective operators throughout the optimization process. This adaptability not only improves solution refinement but also highlights the value of integrating domain-specific strategies, such as transportation constraints, into local search mechanisms to enhance both convergence efficiency and solution quality.

Based on the aforementioned results, we conclude that the reasons for the superior performance of the proposed QQD algorithm over the state-of-the-art algorithms are:

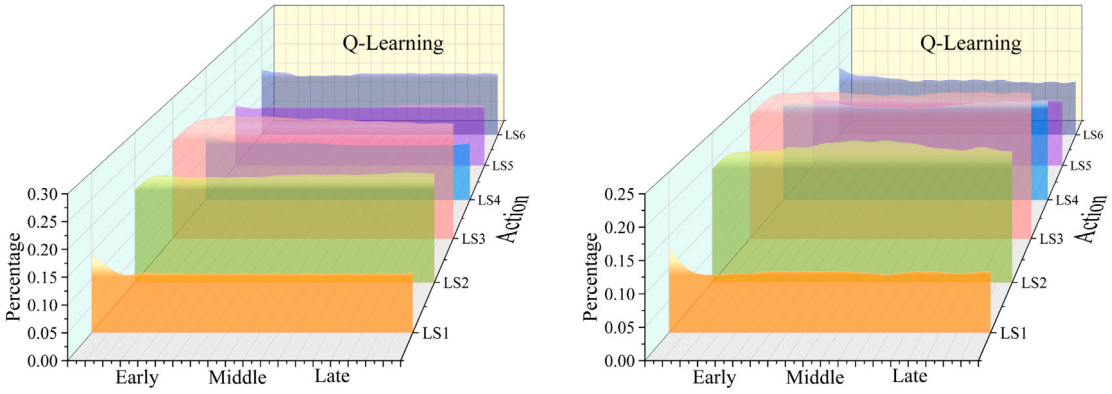
- The comparison between QD and V-QD demonstrates that QD algorithms are conducive to finding diverse solutions in the feature space, preventing the algorithms from getting trapped in local optima. QD algorithms can explore different regions of the solution space and have a higher probability of discovering effective solutions.

- The comparison between QD and V-Q has validated that after selecting local search operators for the QD algorithm, Q-Learning can guide the choice of better local search operators based on past experiences of search. After employing local search operators, Q-Learning is capable of balancing long-term rewards with the effects of local search during the search process, thereby better guiding the decision-making of the search algorithm and avoiding the selection of ineffective operators.

- The comparison of QD and V-LS validates the adopted local search operators, especially the proposed job transportation-based search operator, which greatly improves the local search ability of the proposed QQD algorithm. The strategy further updates the optimal scheme stored in the feature space, which in turn addresses the lack of local search ability of the QD algorithm.

### 5.4. Comparison experiment and analysis

In this section, we compare QQD with five state-of-the-art algorithms. We selected the classical Quality-Diversity (QD) algorithm




(a) 30J7M (b) 50J5M

**Fig. 7.** The proportion of local search operators selected by the Q-learning method at different stages of evolution.

**Table 3**
RPI, mean and best objective values of all comparison algorithms. Bold font represents the best value.

| Instance | RPI | | | | | | MEAN | | | | | | BEST | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IGSA | EDA-VNS | LRVMA | DQCE | QD | QQD | IGSA | EDA-VNS | LRVMA | DQCE | QD | QQD | IGSA | EDA-VNS | LRVMA | DQCE | QD | QQD |
| 20J5M | 0.16† | 0.13† | 0.18† | 0.18† | 0.21† | **0.04** | 5273.65 | 5188.8 | 5330.25 | 5340.05 | 5494.15 | **4720.65** | 4854 | 4746 | 5172 | 5152 | 5412 | **4535** |
| 20J6M | 0.18† | 0.09† | 0.19† | 0.20† | 0.23† | **0.04** | 5225.45 | 5141.45 | 5269.25 | 5345.85 | 5465.55 | **4639.1** | 4943 | 4820 | 5007 | 4963 | 5373 | **4445** |
| 20J7M | 0.17† | 0.15† | 0.22† | 0.24† | 0.27† | **0.04** | 4970 | 4971.1 | 5159.5 | 5247.45 | 5370.3 | **4416.85** | 4619 | 4555 | 5002 | 5103 | 5278 | **4230** |
| 20J8M | 0.19† | 0.16† | 0.21† | 0.21† | 0.27† | **0.03** | 5074.1 | 5064.6 | 5150.9 | 5162.2 | 5391.95 | **4367.35** | 4863 | 4763 | 5036 | 4980 | 5307 | **4257** |
| 30J5M | 0.13† | 0.07† | 0.15† | 0.16† | 0.19† | **0.04** | 7972.6 | 7758.05 | 8117.8 | 8178.25 | 8389.15 | **7309.95** | 7589 | 7427 | 7914 | 7821 | 8255 | **7060** |
| 30J6M | 0.13† | 0.14† | 0.17† | 0.18† | 0.21† | **0.03** | 7780.35 | 7857.6 | 8092.8 | 8169.15 | 8365.25 | **7113.3** | 7344 | 7512 | 7874 | 7935 | 8227 | **6896** |
| 30J7M | 0.14† | 0.12† | 0.18† | 0.19† | 0.22† | **0.03** | 7644.3 | 7584.45 | 7848.55 | 7955.55 | 8156.45 | **6910.5** | 6893 | 7132 | 7709 | 7685 | 7985 | **6679** |
| 30J8M | 0.17† | 0.10† | 0.19† | 0.19† | 0.25† | **0.03** | 7564.95 | 7413.5 | 7713.2 | 7709.8 | 8096.2 | **6649.1** | 7103 | 7051 | 7509 | 7454 | 7960 | **6455** |
| 40J5M | 0.13† | 0.09† | 0.12† | 0.14† | 0.17† | **0.04** | 10 608.15 | 10 189.65 | 10 556.85 | 10 687.55 | 10 946.4 | **9720.3** | 10 062 | 9631 | 10 352 | 10 480 | 10 842 | **9385** |
| 40J6M | 0.12† | 0.09† | 0.13† | 0.14† | 0.18† | **0.02** | 10 309.8 | 10 111 | 10 436.9 | 10 510.3 | 10 864.9 | **9428.5** | 10 004 | 9591 | 10 242 | 9988 | 10 681 | **9199** |
| 40J7M | 0.17† | 0.09† | 0.15† | 0.17† | 0.21† | **0.05** | 10 203.4 | 9871.55 | 10 088.25 | 10 233.8 | 10 624.85 | **9154.3** | 9535 | 9252 | 9855 | 9881 | 10 314 | **8751** |
| 40J8M | 0.19† | 0.14† | 0.18† | 0.18† | 0.25† | **0.04** | 10 151.05 | 9716.5 | 10 056.25 | 10 035.2 | 10 664.4 | **8893.05** | 9639 | 9340 | 9937 | 9673 | 10 477 | **8520** |
| 50J5M | 0.14† | 0.09† | 0.14† | 0.14† | 0.16† | **0.04** | 13 352.25 | 12 772.15 | 13 315.25 | 13 371.85 | 13 623.95 | **12 226.2** | 12 966 | 12 250 | 13 104 | 12 941 | 13 421 | **11 705** |
| 50J6M | 0.13† | 0.09† | 0.13† | 0.13† | 0.17† | **0.03** | 13 110.6 | 12 656.45 | 13 117.5 | 13 193.65 | 13 584.95 | **11 953.1** | 12 465 | 12 013 | 12 922 | 12 850 | 13 464 | **11 635** |
| 50J7M | 0.14† | 0.10† | 0.15† | 0.20† | **0.04** | | 12 728.95 | 12 270.95 | 12 761.35 | 12 799.75 | 13 309.35 | **11 558.4** | 12 047 | 11 725 | 12 445 | 12 481 | 13 013 | **11 120** |
| 50J8M | 0.16† | 0.12† | 0.14† | 0.14† | 0.21† | **0.03** | 12 598.7 | 12 151.5 | 12 417.9 | 12 395.35 | 13 189.35 | **11 172.45** | 12 145 | 11 511 | 12 288 | 11 939 | 12 888 | **10 863** |
| 100J5M | 0.07† | 0.05† | 0.08† | 0.09† | 0.10† | **0.02** | 26 794.95 | 26 084.05 | 26 817.95 | 27 069 | 27 528.1 | **25 519.75** | 26 042 | 25 270 | 26 483 | 26 467 | 27 199 | **24 937** |
| 100J6M | 0.11† | 0.06† | 0.08† | 0.10† | 0.13† | **0.04** | 26 456.5 | 25 440.15 | 25 932.55 | 26 384.85 | 27 006.15 | **24 784.7** | 25 484 | 24 011 | 25 576 | 25 807 | 26 762 | **23 914** |
| 100J7M | 0.12† | 0.07† | 0.08† | 0.11† | 0.15† | **0.03** | 25 464.25 | 24 418.5 | 24 556.3 | 25 154.75 | 26 135.45 | **23 399.1** | 24 582 | 22 879 | 24 224 | 24 716 | 25 528 | **22 653** |
| 100J8M | 0.15† | 0.07† | 0.10† | 0.13† | 0.20† | **0.03** | 24 870.45 | 23 766.7 | 23 845.85 | 24 532.35 | 25 994.7 | **22 452.65** | 23 325 | 22 492 | 23 605 | 23 979 | 25 745 | **21 699** |
| AVG | 0.15 | 0.10 | 0.15 | 0.16 | 0.20 | **0.04** | 12 407.72 | 12 021.44 | 12 329.26 | 12 473.84 | 12 910.08 | **11 319.47** | 11 825.20 | 11 398.55 | 12 112.80 | 12 114.75 | 12 706.55 | **10 946.90** |

**Table 4**
The Friedman non-parametric repeated measures ANOVA test for significant differences among algorithms.

| Variants | RPI | |
|---|---|---|
| | Rank | *p*-value |
| IGSA | 3.575 | 9.28E−18 |
| EDA-VNS | 2.2 | |
| LRVMA | 3.75 | |
| DQCE | 4.475 | |
| QD | 6 | |
| QQD | 1 | |

based on MAP-Elites [41] for our study. In addition, we included IGSA, an algorithm developed specifically to address the transportation aspect of the FJSP [3], and EDA-VNS, a hybrid algorithm combining estimation of distribution algorithm with variable neighborhood search, which has been applied to solve the FJSP with transportation constraints [33]. Furthermore, we incorporated machine learning-based algorithms such as LRVMA, which utilizes Q-Learning [43], and DQCE, employing Deep Reinforcement Learning [48], into our comparative analysis. To guarantee fairness, we implemented all comparison algorithms according to their original works (including parameter settings). We followed the encoding and decoding schema in QQD. Table 3 presents the results of all the compared algorithms. The evaluation indicators we employ are RPI, MEAN, and BEST. AVG represents the average value of each indicator.

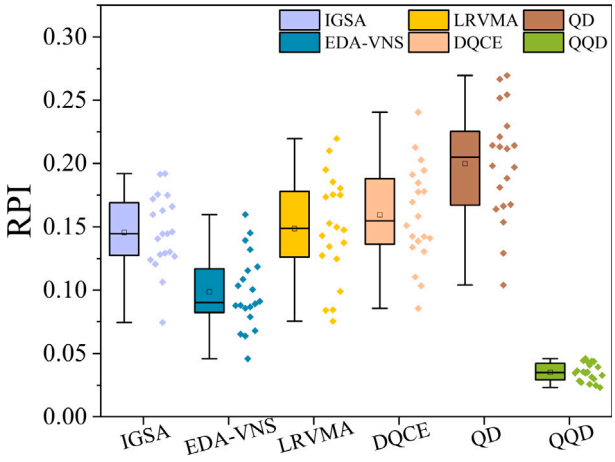



**Fig. 8.** RPI values of QQD and comparison algorithms, shown in the form of box plot.

As can be seen from the Table 3, QQD outperforms other algorithms on all instances. Additionally, to verify the significant difference between the comparison algorithms and QQD, a Wilcoxon test with a confidence interval of 0.05 was performed. † indicates that the current algorithm is significantly different from QQD. The results demonstrate that the QQD algorithm outperforms the state of the art algorithms in all instances. The Wilcoxon test shows that the QQD algorithm is significantly superior to the other compared algorithms. Additionally,

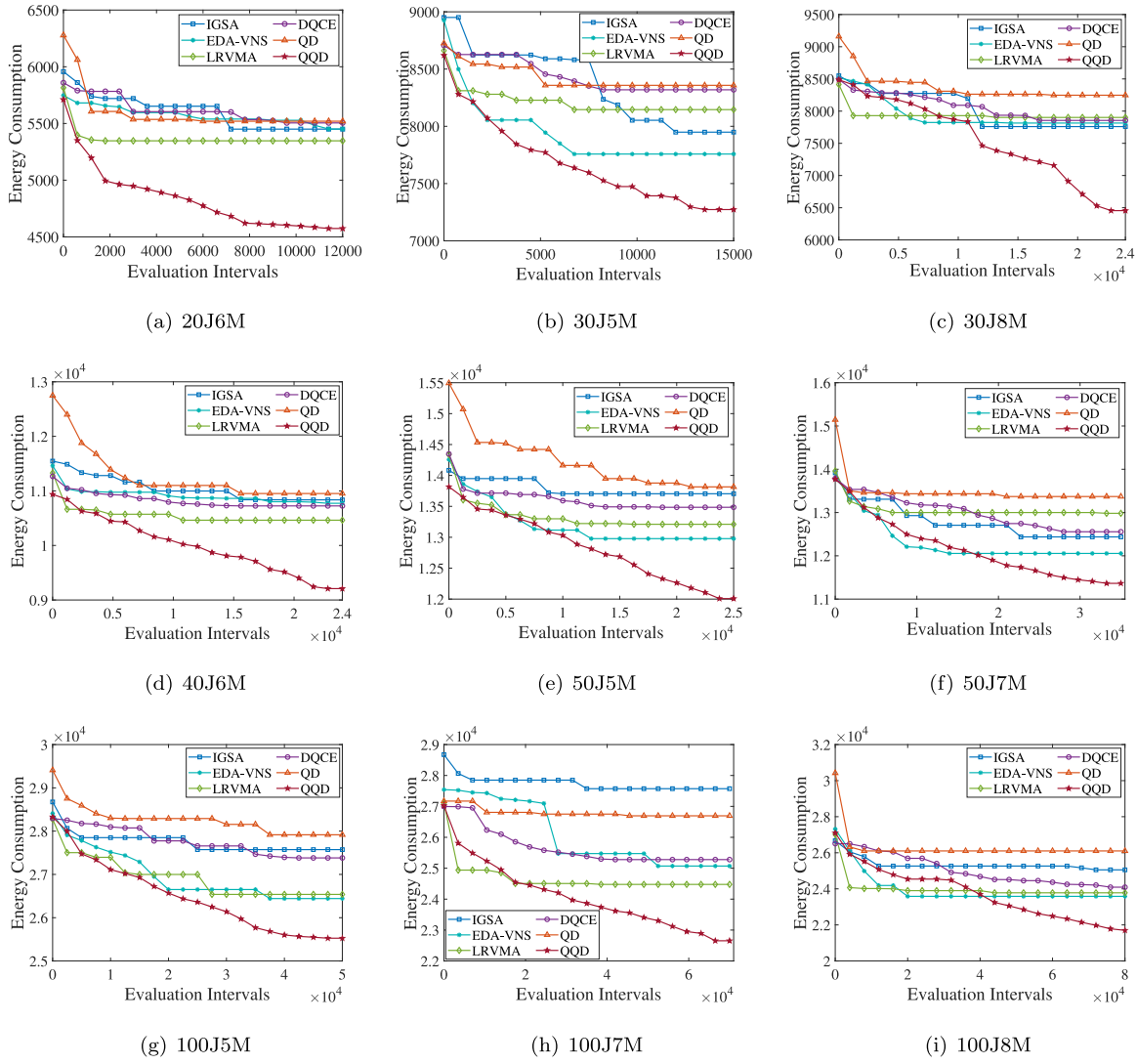



**Fig. 9.** Convergence curves of all algorithms.

to compare the performance of all algorithms, we also conducted a Friedman test. As shown in Table 4, the proposed QQD algorithm is ranked 1st. The *p*-values obtained is less than 0.05 and hence the hypothesis is rejected. This shows a significant difference among the algorithms. QQD shows significant improvement, which can be attributed to (1) the QD framework guarantees diversity of solution features, which enables QQD to avoid local optima and find high-performance solutions in a limited number of evaluations; (2) local search operators, especially based on transportation constraints, can reduce the EC more efficiently; and (3) integrating with the QD framework, Q-Learning can facilitate improved decision-making by utilizing historical experiences of local search operators and current environment, resulting in superior performance compared to random selection.

The above results show that QQD is significantly better than other comparison algorithms. Furthermore, box plots of all compared algorithms are exhibited, the details can be found in Fig. 8. As shown in Fig. 8, QQD yields the best mean and median RPI values and the solutions have outstanding distributivity and quality. The distribution of solutions is stable and dense, with no high outliers. By comparing with the other five state-of-the-art algorithms, it is learned that the proposed QQD algorithm is to be the most efficient in solving FJSP-T. In order to verify the convergence of all the algorithms, we randomly selected 9 different scale instances. As shown in Fig. 9, the *x*-axis represents the number of evaluation intervals and the *y*-axis indicates the objective value: energy consumption. From Fig. 9, it can be seen that the proposed QQD algorithm has the best convergence, followed by EDA-VNS, LRVMA, DQCE, IGSA, and QD. Traditional QD without employing Q-Learning, *LS2*, and *LS3* are inferior to QQD. This further illustrates that the local search ability of the algorithm can be enhanced by combining QD with the Q-Learning method and local search operators based on problem characteristics.

### 5.5. Discovery and discussion

In this section, we discuss some of our findings on solution updating when using the QD framework. We randomly select three different instances (30J5M, 50J7M and 100J8M) and count the number of successful updates of solutions in all cells in the feature space. Successfully updated solutions are attributed to two main aspects, i.e., self and neighboring solutions. To assess the contribution of itself and neighboring solutions to the overall evolution, the percentage of successfully updated solutions in the feature space is recorded. We find that the percentage of solutions obtained from neighborhood solution updates accounts for 86.7% (3923/4535), 85%(8616/10135), 82%(19611/23818) of all successful updates in 30J5M, 50J7M and 100J8M, respectively. We obtain that the updates primarily originate from neighboring solutions rather than from the solution itself. These findings emphasize the importance of diversity of understanding and

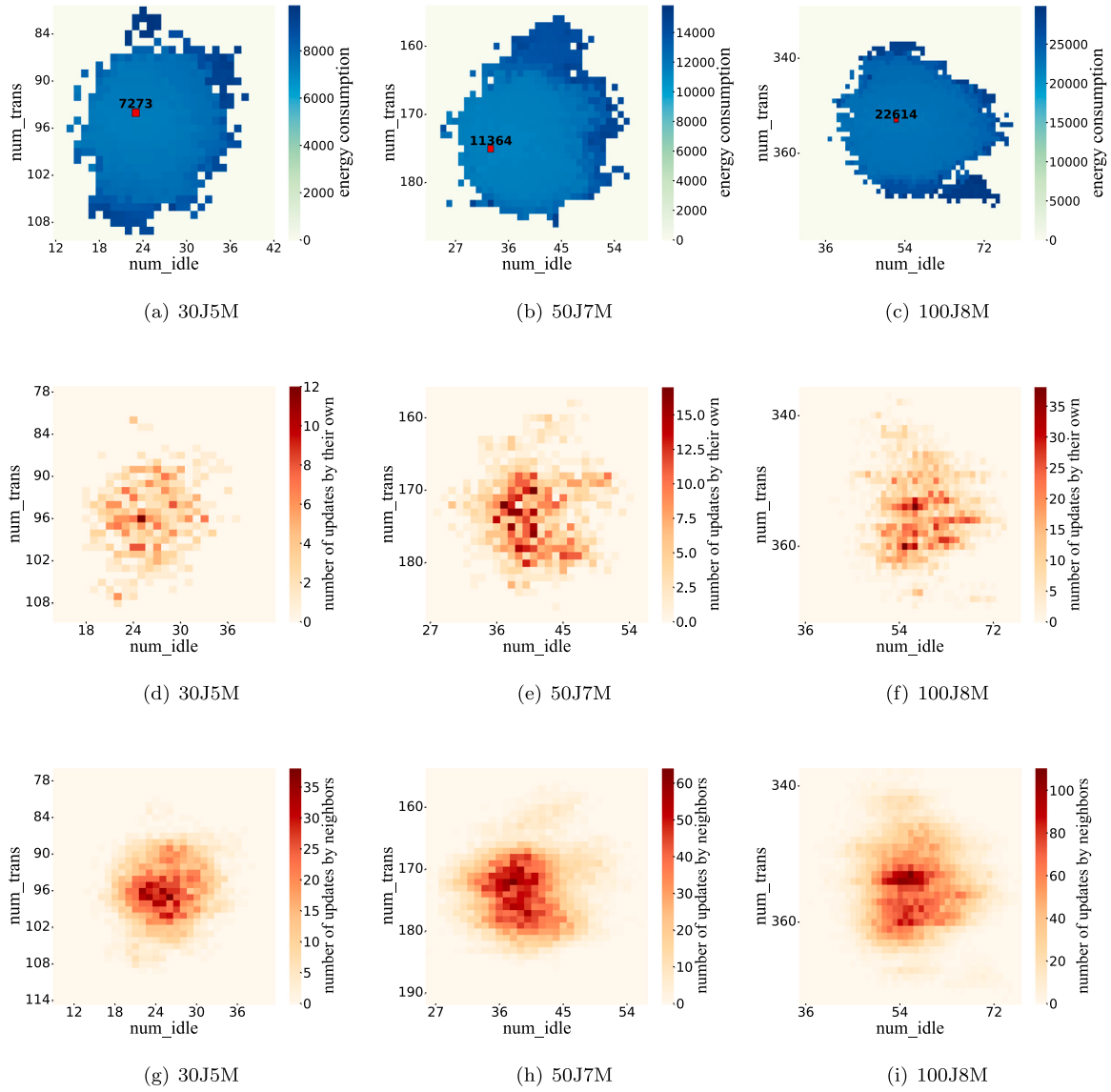



**Fig. 10.** Successfully updated solutions in feature space returned by QQD in three instances (a–c). (d–f) indicate successful updates by their own, (g–i) represent successful updates by their neighboors.

co-evolution. Intelligent decision making through Q-Learning can be further improved by absorbing the knowledge of neighboring solutions using local search operators and thus exploring solutions with different features. This demonstrates the effectiveness of combining the QD framework with Q-Learning.

We randomly select three different instances, i.e., 30J5M, 50J7M, and 100J8M, where J denotes the job and M denotes the machine. Fig. 10 returns all the solutions that are successfully updated in the feature space. the $x$-axis and $y$-axis represent two features: the number of machine idle times (num_idle) and the number of job transportations (num_trans), respectively. As shown in (a–c), in these three instances, QQD obtains 4535, 10135, and 23818 successfully updated solutions in the feature space, respectively. And the best objective values obtained are 7273, 11364, and 22614, respectively.

In addition, to see the number of successful updates by their own and neighbors and the percentages of all successfully updated solutions. Here we draw the solutions that successfully updated in the feature space, as demonstrated in (d–i), Fig. 10, collectively form all the successful updated solutions depicted in (a–c), Fig. 10. Statistics show that the number of solutions successfully updated by themselves in these three instances are 602, 1519, and 4207, which account for 13.3%, 15%, and 18% of the total number of successful updates, respectively. The number of solutions successfully updated by their neighbors are 3923, 8616, and 19611, which account for 86.7%, 85%, and 82% of all successful updates, respectively.

From the depth of the rectangle colors in these figures and statistic results, we know that the number of successful updates by neighbors is much more than the number of successful updates by themselves. It also provides further evidence that the evolution of the solution is accomplished by the solution itself in cooperation with the neighboring solutions.

## 6. Conclusions

In this paper, we propose a QQD algorithm to solve the FJSP-T with the aim of minimizing the energy consumption. QQD employs the Quality-Diversity algorithm in tandem with Q-Learning to cultivate a set of high-performing and diverse solutions. This is achieved by leveraging local search operators that are tailored to the intrinsic features of the problem, thereby enhancing the local search ability while simultaneously preserving solution diversity in the feature space. Extensive experimental comparisons with five state-of-the-art algorithms substantiate the efficacy of the QQD algorithm.

This study demonstrates the effectiveness of the QD framework combined with Q-learning and problem-specific local search strategies in addressing FJSP-T. This study identifies the number of machine idle and the number of job transportation times as effective features in reducing EC. By utilizing the QD framework, the diversity of solutions is maintained, which helps to prevent the algorithm from falling into local optima. The local search operator designed based on transportation constraints can further enhance the local search ability of the QD framework by combining it with the operation sorting, machine ordering search strategies. The utilization of Q-learning plays a vital role in intelligently determining the appropriate strategy. Finally, the cooperative evolution of solutions further supports the effectiveness of the QD framework.

Future research directions may include optimizing the structure of QD, in addition to exploring the integration of QD with contemporary techniques such as evolutionary strategy [62–64], micro-search [65], and deep RL [48].

## CRediT authorship contribution statement

**Haoxiang Qin:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yi Xiang:** Supervision, Formal analysis. **Fangqing Liu:** Supervision. **Yuyan Han:** Supervision. **Yuting Wang:** Supervision.

## Declaration of competing interest

We declare that we 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, China (No. 2024A1515030022); the Fundamental Research Funds for the Central Universities, China (No. 2024ZYGXZR097, 93K172024K03); National Natural Science Foundation of China (No. 61906069); the Natural Science Research Project of Education Department of Guizhou Province, China (No. QJJ2023061); Natural Science Foundation of Guangdong Province, China (No. 2022A1515110058).

## Data availability

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

## References

[1] W. Song, X. Chen, Q. Li, Z. Cao, Flexible job-shop scheduling via graph neural network and deep reinforcement learning, IEEE Trans. Ind. Inform. 19 (2) (2023) 1600–1610, http://dx.doi.org/10.1109/TII.2022.3189725.

[2] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, L. Qian, Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning, IEEE Trans. Ind. Inform. 20 (1) (2024) 1007–1018, http://dx.doi.org/10.1109/TII.2023.3272661.

[3] J.-Q. Li, Y. Du, K.-Z. Gao, P.-Y. Duan, D.-W. Gong, Q.-K. Pan, P.N. Suganthan, A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem, IEEE Trans. Autom. Sci. Eng. 19 (3) (2022) 2153–2170, http://dx.doi.org/10.1109/TASE.2021.3062979.

[4] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, V. Mahmoodian, An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs, Comput. Ind. Eng. (ISSN: 0360-8352) 86 (2015) 2–13.

[5] W. Ren, Y. Yan, Y. Hu, Y. Guan, Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints, Int. J. Prod. Res. 60 (2022).

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

[7] P. Fattahi, M.S. Mehrabad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, J. Intell. Manuf. 18 (3) (2007) 331–342.

[8] Y. Demir, S. Kürşat İşleyen, Evaluation of mathematical models for flexible job-shop scheduling problems, Appl. Math. Model. (ISSN: 0307-904X) 37 (3) (2013) 977–988, http://dx.doi.org/10.1016/j.apm.2012.03.020.

[9] D. Müller, M.G. Müller, D. Kress, E. Pesch, An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning, European J. Oper. Res. 302 (2022).

[10] MILP models for energy-aware flexible job shop scheduling problem, J. Clean. Prod. (ISSN: 0959-6526) 210 (2019) 710–723, http://dx.doi.org/10.1016/j.jclepro.2018.11.021.

[11] Y. Yao, Q. Liu, L. Fu, X. Li, Y. Yu, L. Gao, W. Zhou, A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles, IEEE Trans. Autom. Sci. Eng. (2024) 1–14, http://dx.doi.org/10.1109/TASE.2024.3356255.

[12] S. Karimi, Z. Ardalan, B. Naderi, M. Mohammadi, Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm, Appl. Math. Model. (ISSN: 0307-904X) 41 (2017) 667–682.

[13] S.M. Homayouni, D.B.M.M. Fontes, Production and transport scheduling in flexible job shop manufacturing systems, J. Glob. Optim. (ISSN: 0925-5001) 79 (2) (2021) 463–502.

[14] L. Meng, C. Zhang, Y. Ren, B. Zhang, C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, Comput. Ind. Eng. (ISSN: 0360-8352) 142 (2020) 106347.

[15] N. Zribi, I. Kacem, A.E. Kamel, P. Borne, Assignment and scheduling in flexible job-shops by hierarchical optimization, IEEE Trans. Syst. Man Cybern. 37 (4) (2007) 652–661, http://dx.doi.org/10.1109/TSMCC.2007.897494.

[16] G. Zhang, L. Zhang, X. Song, Y. Wang, C. Zhou, A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem, Cluster Comput. (2018) 1–12, URL https://api.semanticscholar.org/CorpusID:4548038.

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

[18] J. Ding, S. Dauzère-Pérès, L. Shen, Z. Lü, A novel evolutionary algorithm for energy-efficient scheduling in flexible job shops, IEEE Trans. Evol. Comput. 27 (5) (2023) 1470–1484, http://dx.doi.org/10.1109/TEVC.2022.3222791.

[19] T. Jamrus, C.-F. Chien, M. Gen, K. Sethanan, Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing, IEEE Trans. Semicond. Manuf. 31 (2018) 32–41.

[20] M. Jensen, Generating robust and flexible job shop schedules using genetic algorithms, IEEE Trans. Evol. Comput. 7 (3) (2003) 275–288, http://dx.doi.org/10.1109/TEVC.2003.810067.

[21] L. Sun, L. Lin, M. Gen, H. Li, A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling, IEEE Trans. Fuzzy Syst. 27 (5) (2019) 1008–1022, http://dx.doi.org/10.1109/TFUZZ.2019.2895562.

[22] M. Xu, Y. Mei, F. Zhang, M. Zhang, Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling, IEEE Trans. Evol. Comput. (2023) 1, http://dx.doi.org/10.1109/TEVC.2023.3244607.

[23] Y. Yuan, H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms, IEEE Trans. Autom. Sci. Eng. 12 (1) (2015) 336–353, http://dx.doi.org/10.1109/TASE.2013.2274517.

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

[25] I. Kacem, S. Hammadi, P. Borne, Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, IEEE Trans. Syst. Man Cybern. 32 (1) (2002) 1–13, http://dx.doi.org/10.1109/TSMCC.2002.1009117.

[26] Q. Zhang, H. Manier, M.-A. Manier, A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times, Comput. Oper. Res. (ISSN: 0305-0548) 39 (7) (2012) 1713–1723.

[27] S. Karimi, Z. Ardalan, B. Naderi, M. Mohammadi, Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm, Appl. Math. Model. 41 (Jan.) (2016) 667–682.

[28] M. Pal, M.L. Mittal, G. Soni, S.S. Chouhan, M. Kumar, A multi-agent system for FJSP with setup and transportation times, Expert Syst. Appl. (ISSN: 0957-4174) 216 (2023) 119474.

[29] L. Berterottière, S. Dauzère-Pérès, C. Yugma, Flexible job-shop scheduling with transportation resources, European J. Oper. Res. (ISSN: 0377-2217) 312 (3) (2024) 890–909.

[30] M. Dai, D. Tang, A. Giret, M.A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, Robot. Comput. Integr. Manuf. 59 (2019) 143–157.

[31] A. Ham, Transfer-robot task scheduling in flexible job shop, J. Intell. Manuf. (3) (2020).

[32] J. Yan, Z. Liu, C. Zhang, T. Zhang, Y. Zhang, C. Yang, Research on flexible job shop scheduling under finite transportation conditions for digital twin workshop, Robot. Comput.-Integr. Manuf. 72 (2021) 102198.

[33] Y. Du, J. qing Li, C. Luo, L. lei Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, Swarm Evol. Comput. (ISSN: 2210-6502) 62 (2021) 100861.

[34] Q. Luo, Q. Deng, G. Gong, L. Zhang, K. Li, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, Expert Syst. Appl. 160 (2020) 113721.

[35] J. Li, Y. Han, K. Gao, X. Xiao, P. Duan, Bi-population balancing multi-objective algorithm for fuzzy flexible job shop with energy and transportation, IEEE Trans. Autom. Sci. Eng. (2023) 1–17, http://dx.doi.org/10.1109/TASE.2023.3300922.

[36] I.A. Chaudhry, A.A. Khan, A research survey: review of flexible job shop scheduling techniques, Int. Trans. Oper. Res. 23 (2016) 551–591, URL https://api.semanticscholar.org/CorpusID:42778359.

[37] J. Xie, L. Gao, K. Peng, X. Li, L. Haoran, Review on flexible job shop scheduling, IET Collab. Intell. Manuf. 1 (2019) http://dx.doi.org/10.1049/iet-cim.2018.0009.

[38] S. Dauzère-Pérès, J. Ding, L. Shen, K. Tamssaouet, The flexible job shop scheduling problem: A review, European J. Oper. Res. (ISSN: 0377-2217) (2023) http://dx.doi.org/10.1016/j.ejor.2023.05.017.

[39] Z.-Q. Zhang, F.-C. Wu, B. Qian, R. Hu, L. Wang, H.-P. Jin, A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation, Expert Syst. Appl. (ISSN: 0957-4174) 234 (2023) 121050, http://dx.doi.org/10.1016/j.eswa.2023.121050.

[40] J.K. Pugh, L.B. Soros, K.O. Stanley, Quality diversity: A new frontier for evolutionary computation, Front. Robot. Ai 3 (2016).

[41] J.-B. Mouret, J. Clune, Illuminating search spaces by mapping elites, 2015,

[42] Z. Pan, L. Wang, J. Wang, Q. Zhang, A bi-learning evolutionary algorithm for transportation-constrained and distributed energy-efficient flexible scheduling, IEEE Trans. Evol. Comput. (2024).

[43] R. Li, W. Gong, C. Lu, L. Wang, A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time, IEEE Trans. Evol. Comput. 27 (3) (2023) 610–620, http://dx.doi.org/10.1109/TEVC.2022.3175832.

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

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

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

[47] F. Zhao, Z. Wang, L. Wang, A reinforcement learning driven artificial bee colony algorithm for distributed heterogeneous no-wait flowshop scheduling problem with sequence-dependent setup times, IEEE Trans. Autom. Sci. Eng. 20 (4) (2023) 2305–2320.

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

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

[50] A. Cully, J. Clune, D. Tarapore, J.-B. Mouret, Robots that can adapt like animals, Nature (ISSN: 1476-4687) 521 (7553) (2015) 503–507, http://dx.doi.org/10.1038/nature14422.

[51] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, First return, then explore, Nature (ISSN: 1476-4687) 590 (7847) (2021) 580–586, http://dx.doi.org/10.1038/s41586-020-03157-9.

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

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

[54] B. Lim, M. Flageat, A. Cully, Understanding the synergies between quality-diversity and deep reinforcement learning, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1212–1220, http://dx.doi.org/10.1145/3583131.3590388.

[55] M. Flageat, B. Lim, L. Grillotti, M. Allard, S.C. Smith, A. Cully, Benchmarking quality-diversity algorithms on neuroevolution for reinforcement learning, 2022, arXiv:2211.02193.

[56] H. Mokhtari, A. Hasani, An energy-efficient multi-objective optimization for flexible job-shop scheduling problem, Comput. Chem. Eng. (ISSN: 0098-1354) 104 (2017) 339–352, http://dx.doi.org/10.1016/j.compchemeng.2017.05.004, URL https://www.sciencedirect.com/science/article/pii/S009813541730203X.

[57] X. Tao, Q. Pan, L. Gao, An iterated greedy algorithm with reinforcement learning for distributed hybrid FlowShop problems with job merging, IEEE Trans. Evol. Comput. (2024) 1.

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

[59] B. Ümit, U. Gündüz, A time window approach to simultaneous scheduling of machines and material handling system in an FMS, Oper. Res. (ISSN: 0030-364X) 43 (6) (1995) 1058–1070, http://dx.doi.org/10.1287/opre.43.6.1058.

[60] H. Qin, W. Bai, Y. Xiang, F. Liu, Y. Han, L. Wang, A self-adaptive collaborative differential evolution algorithm for solving energy resource management problems in smart grids, IEEE Trans. Evol. Comput. 28 (5) (2024) 1427–1441, http://dx.doi.org/10.1109/TEVC.2023.3312769.

[61] H. Qin, Y. Han, Y. Wang, Y. Liu, J. Li, Q. Pan, Intelligent optimization under blocking constraints: A novel iterated greedy algorithm for the hybrid flow shop group scheduling problem, Knowl.-Based Syst. (ISSN: 0950-7051) 258 (2022) 109962, http://dx.doi.org/10.1016/j.knosys.2022.109962.

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

[63] Y. Liang, H. Huang, Z. Cai, Z. Hao, Multiobjective evolutionary optimization based on fuzzy multicriteria evaluation and decomposition for image matting, IEEE Trans. Fuzzy Syst. 27 (5) (2019) 1100–1111.

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

[65] H. Huang, S. Yang, X. Li, Z. Hao, An embedded Hamiltonian graph-guided heuristic algorithm for two-echelon vehicle routing problem, IEEE Trans. Cybern. 52 (7) (2022) 5695–5707, http://dx.doi.org/10.1109/TCYB.2021.3108597.