Taylor & Francis
Taylor & Francis Group

Engineering Optimization

ISSN: 0305-215X (Print) 1029-0273 (Online) Journal homepage: www.tandfonline.com/journals/geno20

Handling integrated transportation and production scheduling via deep-Q-network-enhanced multi-objective quality–diversity algorithm

Ronghua Zou , Haoxiang Qin, Yi Xiang & Chunguo Wu

To cite this article: Ronghua Zou , Haoxiang Qin, Yi Xiang & Chunguo Wu (23 Mar 2026): Handling integrated transportation and production scheduling via deep-Q-network-enhanced multi-objective quality–diversity algorithm, Engineering Optimization, DOI: 10.1080/0305215X.2026.2637656

To link to this article: https://doi.org/10.1080/0305215X.2026.2637656

ENGINEERING OPTIMIZATION
https://doi.org/10.1080/0305215X.2026.2637656

Taylor & Francis
Taylor & Francis Group

Handling integrated transportation and production scheduling via deep-Q-network-enhanced multi-objective quality–diversity algorithm

Ronghua Zou[a,b], Haoxiang Qin [c], Yi Xiang [c,d] and Chunguo Wu[d]

[a]College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, People's Republic of China; [b]Finance Department, China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, People's Republic of China; [c]School of Software Engineering, South China University of Technology, Guangzhou, People's Republic of China; [d]Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun, People's Republic of China

ABSTRACT

Quality–diversity (QD) optimization, as a paradigm of evolutionary algorithms, aims to generate diverse and high-performing solutions. However, existing QD algorithms fail to leverage historical and domain sequential scheduling knowledge effectively, making it challenging for them to handle large-scale discrete problems with conflicting objectives. This article proposes an extension of QD optimization algorithms, called the Deep Reinforcement Learning Enhanced Multi-Objective MAP-Elites (DMOME) algorithm, to solve a real-world combinatorial optimization problem. The DMOME algorithm seamlessly combines the feature diversity of solutions, domain knowledge heuristics and deep reinforcement learning techniques by adding Pareto fronts to different cells. The results on real-world mechanical processing factory instances demonstrate that the DMOME algorithm outperforms state-of-the-art algorithms, such as the enhanced genetic algorithm, achieving improvements of 25.6% to 59.4% in performance metrics.

ARTICLE HISTORY
Received 4 September 2025
Accepted 21 February 2026

KEYWORDS
Quality–diversity optimization; evolutionary computation; knowledge-driven; integrated transportation and production scheduling

1. Introduction

Evolutionary Algorithms (EAs) are general optimization methods that yield a set of solutions, simulating natural evolution through iterative operations like selection and reproduction (Bäck 1996; Coello Coello, Van Veldhuizen, and Lamont 2007). Quality–Diversity (QD) EAs (Cully and Demiris 2018; Pugh, Soros, and Stanley 2016), a paradigm within EAs, aim to find a set of diverse and high-performing solutions (stored in the archive), rather than only the best one. One of the most well-known QD algorithms is the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) (Mouret and Clune 2015), which 'illuminates' the feature space by dividing it into a grid of descriptor niches, each maintaining its best solution. Specifically, MAP-Elites operates iteratively by selecting parent solutions from the archive, applying reproduction operators like crossover and mutation to create offspring, and updating the archive with the offspring.

MAP-Elites has been used in several fields such as robotics (Cully *et al.* 2015; Duarte *et al.* 2018; Vassiliades, Chatzilygeroudis, and Mouret 2018), games (Ecoffet *et al.* 2021; Fontaine *et al.* 2021; Khalifa *et al.* 2018), neural networks (Bossens, Mouret, and Tarapore 2020; Grillotti and Cully 2022;

CONTACT Haoxiang Qin 987352978@qq.com

Nilsson and Cully 2021) and combinatorial optimization (Dang *et al.* 2024; Nikfarjam, A. Neumann, and F. Neumann 2024; Urquhart and Hart 2018). Among them, QD studies on combinatorial optimization are particularly scarce. This study is conducted in the context of Integrated Transportation and Production Scheduling (ITPS), a complex real-world problem that has received limited attention (Huang *et al.* 2026; Qin *et al.* 2024, 2025). In ITPS, production and transportation tasks are carried out simultaneously (Chaudhry *et al.* 2022; Fu *et al.* 2025; He *et al.* 2022). A series of pre-defined jobs are assigned to a set of heterogeneous machines in a reasonable machine order to achieve the desired processing goals. Each job has the flexibility to choose a processing machine and is transported to the selected machine by equipment such as Automated Guided Vehicles (AGVs) (Meng *et al.* 2025). Recently, exploring solutions with diverse features has shown to be beneficial in ITPS applications (Qin *et al.* 2025). However, this study focuses solely on single-objective ITPS, overlooking scenarios with conflicting objectives. Additionally, it relies on basic Reinforcement Learning (RL) techniques for selecting search operators, with limited exploitation of domain knowledge in the solution process.

For this purpose, this article summarizes the reasons that QD is suitable for solving multi-objective ITPS, as follows.

- Prevalence of multi-objective requirements: real-world ITPS requires optimizing multiple conflicting objectives rather than a single goal. For example, the crane transportation processing problem minimizes makespan and energy consumption (J.-Q. Li *et al.* 2022), while the engine workshop of General Motors aims to reduce carbon emissions while ensuring timely completion (Mou *et al.* 2023). Similarly, machining plants seek to minimize both energy consumption and makespan (Dai *et al.* 2019; Pan, L. Wang, J. Wang, and Zhang 2024). Ignoring one objective leads to impractical solutions, necessitating Multi-Objective Evolutionary Algorithms (MOEAs). However, traditional MOEAs primarily optimize along a single Pareto frontier, often overlooking diversity, which can degrade performance and lead to local optima (Y. Wang, Xue, and Qian 2022; R.-J. Wang *et al.* 2023). QD overcomes these limitations by explicitly promoting diversity across the feature space while maintaining solution quality, making it well-suited for multi-objective ITPSs (Pierrot *et al.* 2022). However, most existing multi-objective QD studies focus on continuous optimization (Bossens and Tarapore 2022; Qian, Xue, and Wang 2024; Samuelsen and Glette 2018), limiting their direct applicability to discrete ITPS problems.
- Although existing QD methods attempt to maintain local Pareto fronts across different niches, they often fail to utilize problem-specific knowledge fully (Qin *et al.* 2024; Yao *et al.* 2025). For instance, in ITPS, factors such as unnecessary AGV transportation and excessive machine idling can lengthen completion times and increase energy consumption. However, most optimization algorithms rely on search or perturbation operators that directly manipulate the encoding vectors of solution, without considering machine idle time or AGV transport characteristics. This oversight of the knowledge can negatively impact algorithm performance (Bhosale and Pawar 2025; Cheng *et al.* 2025; Qin *et al.* 2026). Incorporating domain knowledge has been shown to improve performance significantly in multi-objective algorithms (Du *et al.* 2024; F. Zhang, Li, and Gong 2024; Pan, L. Wang, J. Wang, Yu, *et al.* 2024), suggesting its potential for enhancing QD optimization in specialized tasks.
- Insufficient utilization of historical and real-time data weakens decision-making in multi-objective QD optimization, particularly in complex tasks like transportation and production scheduling (Qin *et al.* 2023; R. Li *et al.* 2024; Y. Li *et al.* 2022). While methods like MOME (Pierrot *et al.* 2022) have achieved promising results, they rely on non-directed genetic operators. As a result, their adaptability in dynamic environments is limited (Janmohamed, Pierrot, and Cully 2023). Knowledge-driven strategies, such as AGV allocation and critical path rules, have shown effectiveness in ITPS (Han *et al.* 2024; W. Li *et al.* 2024). However, these strategies lack adaptive learning (Chen *et al.* 2024; Z. Zhang *et al.* 2023). Deep Q-Networks (DQNs) offer a solution by efficiently searching discrete decision spaces and generalizing from experience via neural networks, thus improving adaptability and stability over conventional Q-learning. Recent studies (Du *et al.* 2024;

F. Zhang, Li, and Gong 2024; X. Zhou *et al.* 2025) have demonstrated that integrating DQNs into evolutionary algorithms enhances learning efficiency. However, multi-objective QD methods have yet to leverage this advantage fully. Incorporating DQNs can significantly improve operator selection, leading to better exploration–exploitation balance and high-quality ITPS solutions.

To address the above issues, this article proposes a multi-objective QD optimization approach and applies it to a real-world ITPS. The problem is first modelled as a multi-objective QD optimization task. A Deep Reinforcement Learning Enhanced Multi-Objective MAP-Elites (DMOME) algorithm is then developed to solve it. the DMOME algorithm incorporates AGV transportation knowledge and leverages a DQN model to learn from historical data, enabling more informed decision-making for enhanced operator selection. Specifically, the DMOME algorithm maintains and updates local Pareto solution sets in each feature space niche, promoting diversity and high performance. This article evaluated the DMOME algorithm on various mechanical processing factory instances, confirming the suitability of modelling ITPS as a multi-objective QD optimization problem. Results demonstrate that the knowledge-driven enhanced strategies and DQN selection mechanism significantly improve performance.

2. Related work

2.1. Integrated transportation and production scheduling

With the gradual increase in the demand for AGVs in manufacturing plants, research related to AGV scheduling has received more and more attention (Bhosale and Pawar 2025; J. Wang, L. Wang, and Han 2025). If the scheduling of AGVs in ITPS is handled properly, transportation costs and time can be saved effectively. Bilge and Ulusoy (1995) formally established ITPS benchmarks for the first time, prompting studies on optimization strategies. Early studies primarily addressed machine scheduling and AGV scheduling as separate problems. Ulusoy and Bilge (1992) explored AGV return policies, highlighting the trade-off between scheduling simplicity and increased transportation time. Later, Ulusoy, Sivrikaya-Şerifoğlu, and Bilge (1997) proposed Genetic Algorithms (GAs) for simultaneous scheduling, though challenges like premature convergence persisted. To address this issue, researchers also introduced adaptive GAs (Jerald *et al.* 2006), chaos-based ant colony optimization (Mahdavi, Shirazi, and Sahebjamnia 2011) and constraint models (Novas and Henning 2014; Yao, Liu, *et al.* 2024), among other metaheuristics. Furthermore, Abdelmaguid *et al.* (2004) introduced operation-based encoding, while Hurink and Knust (2002) and Yao, Gui, *et al.* (2024) explored neighbourhood structures. Hybrid methods, such as Tabu Search (TS) (Y. Zheng, Xiao, and Seo 2014) differential evolution (Babu *et al.* 2010) and Variable Neighbourhood Search (VNS) (Abderrahim *et al.* 2022), have improved solution quality. However, most studies focus on search strategies rather than deeper problem analysis, leading to computational inefficiencies in large-scale instances (Yao *et al.* 2025). The research should explore domain-specific knowledge integration to enhance the efficiency of ITPS (Du *et al.* 2024; F. Zhang, Li, and Gong 2024; Pan, L. Wang, J. Wang, Yu, *et al.* 2024).

2.2. Quality–diversity optimization

Unlike traditional optimization methods that aim for a global optimum or Pareto set, QD optimization seeks to discover diverse and high-performing solutions across a feature space $\mathcal{B}$ (Cully and Demiris 2018; Nilsson and Cully 2021). Each solution is defined by its objective values f and a feature vector $\boldsymbol{b}_x$, representing task-specific properties. For example, in robotic locomotion, the feature vector may represent the robot's gait (Cully *et al.* 2015; Pierrot *et al.* 2022; Vassiliades, Chatzilygeroudis, and Mouret 2018), while in video game design, it might describe the number of enemy characters (Khalifa *et al.* 2018). QD uses local competition to retain only the best solution in each behavioural niche (Lim, Flageat, and Cully 2023). To handle conflicting objectives, multi-objective QD extends this idea by maintaining Pareto fronts within each niche. A representative method is MOME (Pierrot

et al. 2022), which incorporates new solutions only if they dominate existing ones in the same niche, thus balancing diversity and trade-off quality (Janmohamed, Pierrot, and Cully 2023; Janmohamed *et al.* 2024; Nickelson *et al.* 2023). To control memory and enable parallelism, MOME limits the size of the archive (Pierrot *et al.* 2022). Though promising in robotics, multi-objective QD methods remain unexplored in discrete real-world problems such as ITPS.

2.3. Reinforcement learning and deep reinforcement learning

RL (Cao, Lin, and M. Zhou 2021; R. Li, Gong, Lu, *et al.* 2023) and neural networks (Du *et al.* 2024; Yu and Liang 2001) have shown particular promise for combinatorial optimization. For example, R. Li, Gong, Lu, *et al.* (2023) proposed a mixed-integer linear programming formulation combined with an RL-based reference vector multi-agent system to solve the type-2 Fuzzy flexible Job shop Scheduling Problem (FJSP). Similarly, Yu and Liang (2001) introduced a hybrid model integrating neural networks with GAs, demonstrating improved scheduling efficiency. Moreover, Deep Reinforcement Learning (DRL) has become a popular strategy in scheduling research (Luo, Zhang, and Fan 2022; Song *et al.* 2023). DRL methods leverage neural networks to approximate value functions or policies, enabling agents to learn effective decision-making strategies from both historical data and real-time environments. This allows for more informed operator or rule selection, reducing reliance on manual heuristics and mitigating the inefficiencies of random choices (R. Li, Gong, L. Wang, *et al.* 2023; Song *et al.* 2023).

Despite its strengths, DRL-based methods continue to face a significant challenge: the lack of behavioural diversity. Traditional RL and DRL agents often converge to a limited set of high-reward policies, restricting exploration and compromising solution robustness, particularly in dynamic or stochastic environments. This limitation is especially evident in tasks requiring adaptive operator selection or diverse local search behaviours (Z. Zhang *et al.* 2023). Moreover, many RL-driven scheduling frameworks prioritize exploitation—refining high-quality solutions—while neglecting thorough exploration of alternative strategies or dispatching rules. To address this, researchers have increasingly explored hybrid approaches that integrate RL with evolutionary algorithms, enhancing adaptability and robustness without sacrificing solution quality (R. Li, Gong, L. Wang, *et al.* 2023). However, achieving an optimal balance between performance and behavioural diversity remains an open challenge.

3. Preliminaries

3.1. Problem description

In ITPS, three main subproblems must be addressed: (Sub-Problem 1) assigning machines to jobs; (Sub-Problem 2) sequencing the operations of jobs; and (Sub-Problem 3) allocating AGVs for transportation. Their interactions are visualized in Figure 1. For ITPS, consider a scenario where n distinct jobs need to be processed on m machines. Each job comprises w_i operations, which must be processed on the machines following a predefined sequence. There are v identical AGVs available to transport jobs between machines. Initially, an AGV transports a job from the depot to a machine for the first operation. Subsequently, the AGV transfers the job to other machines for further processing. Each AGV has three states: loaded; unloaded; and idle (when waiting for tasks). The processing time for the jth operation ($O_{i,j}$) of job i varies depending on the machine k, and the transportation time for an AGV to deliver job i to different machines k also differs.

ITPS has the following assumptions.

- At time zero, all jobs and AGVs are available and initially located at the depot.
- Each operation has the same processing priority and must be processed on a single machine at a time, with each machine handling only one operation simultaneously.

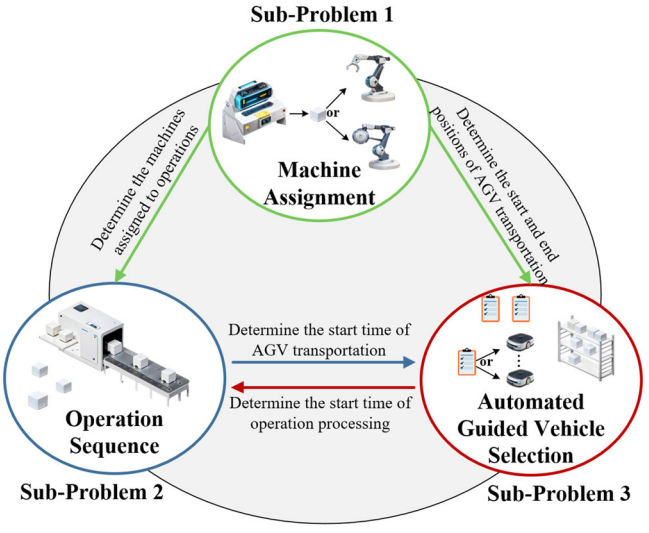


Figure 1. Relationship of the three tasks.

- Processing is non-preemptive, meaning that once an operation begins, it cannot be interrupted.
- The buffer capacity for each machine is assumed to be sufficient.
- Each job is assigned to a single AGV at any given time, and each AGV can transport only one job at a time.
- The AGVs operate under ideal conditions, without considering failures, road congestion or other disruptions.
- Once a transportation task begins, it must be completed without interruption.
- Upon completing a task, the AGV immediately proceeds to the next task; if the next task is at the same machine, it remains in place and waits for task completion.

In this study, a scheduling solution is encoded as a composite decision vector $x = (OS, MA, AS)$, where OS denotes the operation sequence, MA denotes the machine assignment, and AS denotes the AGV allocation sequence. These components jointly determine the start time, completion time and transportation decisions of all operations. The feasible decision space $\mathcal{D}$ consists of all decision vectors ($x = (OS, MA, AS)$) satisfying the precedence constraints, machine capacity constraints, AGV availability constraints and transportation timing constraints.

As in the work of Dai *et al.* (2019), the ITPS problem considers two optimization objectives: makespan and total energy consumption. The Multi-objective Optimization Problem (MOP) is formulated as

$$\min_{x\in\mathcal{D}} \boldsymbol{f}(x) = (f_1(x), f_2(x)).$$

The first objective minimizes the makespan:

$$f_1(x) = C_{\max}(x), \tag{1}$$

where $C_{\max}(x)$ denotes the maximum completion time under scheduling solution x.
The second objective minimizes the total energy consumption:

$$f_2(x) = TPE(x) + TIE(x) + TTE(x), \tag{2}$$

where all energy-related quantities are evaluated under the same feasible solution x.

Given a feasible scheduling solution x, the energy consumption components are computed as follows.

(a) Processing energy:

$$TPE(x) = \sum_{k \in M} (PP_k \times TP_k(x)), \tag{3}$$

where PP_k is the processing power of machine k, and $TP_k(x)$ is the total processing time assigned to machine k under solution x.

(b) Idle energy:

$$TIE(x) = \sum_{k \in M} (PI_k \times TI_k(x)), \tag{4}$$

where PI_k denotes the idle power of machine k, and $TI_k(x)$ represents its total idle time.

(c) Transportation energy:

$$TTE(x) = \sum_{a \in A} \left(PT_a \times \sum_{i \in I} \sum_{j \in J_i} TO_{i,j,a}(x) \right), \tag{5}$$

where PT_a is the transportation power of AGVa, and $TO_{i,j,a}(x)$ is the transportation time of operation $O_{i,j}$ executed by AGVa.

Traditional multi-objective optimization seeks a Pareto set in the objective space. In contrast, this work reformulates ITPS as a QD optimization problem, where high-performing solutions are discovered and maintained within different regions of a predefined feature space. Let $\boldsymbol{b}_x = (b_1(x), b_2(x), \ldots, b_r(x))$ denote the feature descriptor vector associated with solution x, where r is the feature-space dimensionality.

For each feature niche $\boldsymbol{b} \in \mathcal{B}$, the QD optimization problem is defined as a conditional subproblem:

$$PS(\boldsymbol{b}) = \arg\min_{x \in \mathcal{D}} \boldsymbol{f}(x) \quad \text{s.t. } \boldsymbol{b}_x = \boldsymbol{b}. \tag{6}$$

Here, the QD formulation introduces an additional feature-based partitioning of the feasible search space.

Following W. Li *et al.* (2024), as shown in Tables 1 and 2, this work outlines the relevant notation and decision variables of ITPS.

The constraints of ITPS are listed as follows:

$$\sum_{k \in M_{i,j}} l_{i,j,k} = 1, \quad \forall\, i \in I,\ \forall\, j \in J_i \tag{7}$$

$$y_{i,j,i',j',k} + y_{i',j',i,j,k} \le l_{i,j,k}, \quad \forall\, i, i' \in I, \\ \forall\, j \in J_i,\ \forall\, j' \in J_{i'},\ i < i' \vee (i = i' \wedge j < j'),\ \forall\, k \in M_{i,j} \cap M_{i',j'} \tag{8}$$

$$y_{i,j,i',j',k} + y_{i',j',i,j,k} \ge l_{i,j,k} + x_{i',j',k} - 1, \quad \forall\, i, i' \in I, \\ \forall\, j \in J_i,\ \forall\, j' \in J_{i'},\ i < i' \vee (i = i' \wedge j < j'),\ \forall\, k \in M_{i,j} \cap M_{i',j'} \tag{9}$$

$$C_{i,j} \ge C_{i,j-1} + \sum_{k \in M_{i,j}} \left(l_{i,j,k} \times T_{i,j,k} \right), \quad \forall\, i \in I,\ \forall\, j \in J_i - \{1\} \tag{10}$$

$$C_{i',j'} \ge C_{i,j} + T_{i',j',k} + (y_{i,j,i',j',k} - 1) \times H, \\ \forall\, i, i' \in I,\ \forall\, j \in J_i,\ \forall\, j' \in J_{i'},\ i < i' \vee (i = i' \wedge j < j') \tag{11}$$

Table 1. ITPS notation.

Symbol	Description
n	Total number of jobs.
w_i	Number of operations of job i.
$n_{\max}$	Total number of operations of all jobs.
m	Total number of machines.
v	Total number of AGVs.
i, i'	Index of jobs.
j, j'	Index of operations.
k, k'	Index of machines.
a	Index of AGVs.
I	Set of jobs, $I = \{1, \ldots, i, \ldots, n\}$.
A	Set of AGVs, $A = \{1, \ldots, a, \ldots, v\}$.
J_i	Operation set of job i, $J_i = \{1, \ldots, j, \ldots, w_i\}$.
M	Set of machines, $M = \{1, \ldots, k, \ldots, m\}$.
$O_{i,j}$	Operation j of job i.
$M_{i,j}$	Available machines for $O_{i,j}$, $M_{i,j} \subseteq M$.
$T_{i,j,k}$	Processing time of $O_{i,j}$ on machine k.
$TR_{k,k'}$	Transport time from machines k to k'.
$TR_{0,k}$	Transport time from depot to machine k.
$TR_{k,0}$	Transport time from machine k to depot.
PP_k	Power of machine k when processing per unit time.
PI_k	Power of machine k when idle per unit time.
PT_a	Power of AGVa in transport per unit time.
H	A sufficiently large constant.

Table 2. Decision variables and performance metrics.

Symbol	Description
$l_{i,j,k}$	Binary variable; equals one if $O_{i,j}$ is assigned to machine k, otherwise zero.
$y_{i,j,i',j',k}$	Binary variable; equals one if $O_{i,j}$ precedes $O_{i',j'}$ on machine k, otherwise zero.
$w_{i,j,i',j',a}$	Binary variable; equals one if AGVa transports $i \rightarrow i'$ between $O_{i,j}$ and $O_{i',j'}$, otherwise zero. $w_{0,1,i',j'} = 1$ means i' is the first job transported by the AGV. $w_{i,j,0,1} = 1$ means i is the last job transported by the AGV.
$C_{\max}$	Makespan (completion time of last operation).
$C_{i,j}$	Completion time of operation $O_{i,j}$.
$S_{i,j}$	Arrival time of job i at the machine for $O_{i,j}$.
$TO_{i,j,a}$	Time for AGVa to transport job i to the machine for $O_{i,j}$.
TP_k	Total processing time on machine k.
TI_k	Total idle time on machine k.
TPE	Total energy consumed in processing.
TIE	Total idle energy consumption of machines.
TTE	Total transportation energy consumption by AGVs.

$$\sum_{i' \in I \cup \{0\}} \sum_{j' \in J_{i'}} w_{0,1,i',j',a} = 1 \tag{12}$$

$$\sum_{i \in I} \sum_{j \in J_i} w_{i,j,0,1,a} = 1, \quad \forall\, a \in A \tag{13}$$

$$C_{i,j} \geq S_{i,j} + \sum_{k \in M_{i,j}} \left(l_{i,j,k} \times T_{i,j,k} \right), \quad \forall\, i \in I,\ \forall\, j \in J_i,\ \forall\, k \in M_{i,j} \tag{14}$$

$$\begin{aligned} &S_{i,j} \geq C_{i,j-1} + TR_{k,k'} + (l_{i,j,k} + x_{i,j-1,k'} - w_{0,0,i,j,a} - 2) \times H, \\ &\quad \forall\, i \in I,\ \forall\, j \in J_i - \{1\},\ \forall\, k \in M_{i,j},\ \forall\, k' \in M_{i,j-1},\ k \neq k',\ \forall\, a \in A \end{aligned} \tag{15}$$

$$\begin{aligned} &TO_{i',j',a} \geq H \times (l_{i,j,k} + x_{i',j',k'} + x_{i',j'-1,k''} + w_{i,j,i',j',a} - 4) + TR_{k,k''} + TR_{k'',k'}, \\ &\quad \forall\, i \in I,\ \forall\, i' \in I,\ \forall\, j \in J_i,\ \forall\, j' \in J_{i'} - \{1\},\ i \neq i' \vee\ (i = i' \wedge j < j'), \end{aligned}$$

$$\forall k \in M_{i,j},\ \forall k' \in M_{i',j'},\ \forall k'' \in K_{i',j'-1},\ k' \neq k'',\ \forall a \in A \tag{16}$$

$$TO_{i',1,a} \geq TR_{k,0} + TR_{0,k'} + (l_{i,j,k} + x_{i',1,k'} + w_{i,j,i',1,a} - 3) \times H,$$
$$\forall i \in I,\ \forall i' \in I,\ \forall j \in J_i,\ i \neq i',\ \forall k \in M_{i,j},\ \forall k' \in M_{i',1},\ \forall a \in A \tag{17}$$

$$TO_{i',j',a} \geq H \times (x_{i',j',k'} + x_{i',j'-1,k''} + w_{0,1,i',j',a} - 3) + TR_{0,k''} + TR_{k'',k'},\quad \forall i' \in I,\ \forall j' \in J_{i'} - \{1\},$$
$$\forall k' \in M_{i',j'},\ \forall k'' \in M_{i',j'-1},\ k' \neq k'',\ \forall a \in A \tag{18}$$

$$TO_{i',1,a} \geq TR_{0,k'} + (x_{i',j,k'} + w_{0,1,i',1,a} - 2) \times H,$$
$$\forall i' \in I,\ \forall k' \in M_{i',1},\ \forall a \in A \tag{19}$$

$$S_{i',j'} \geq S_{i,j} + TO_{i',j',a} + (w_{i,j,i',j',a} - 1) \times H,$$
$$\forall i, i' \in I,\ \forall j \in J_i,\ \forall j' \in J_{i'},\ i \neq i' \vee (i = i' \wedge j < j'),\ \forall a \in A \tag{20}$$

$$S_{i',j'} \geq TO_{i',j',a} + (w_{0,1,i',j',a} - 1) \times H,$$
$$\forall i' \in I,\ \forall j' \in J_{i'},\ \forall a \in A \tag{21}$$

$$C_{\max} \geq C_{i,w_i},\quad \forall i \in I \tag{22}$$

$$TP_k = \sum_{i \in I} \sum_{j \in J_i} \left(l_{i,j,k} \times T_{i,j}\right),\quad \forall k \in M \tag{23}$$

$$TI_k \geq C_{i,j} - \sum_{i \in I} \sum_{j \in J_i} \left(l_{i,j,k} \cdot T_{i,j,k}\right) + (l_{i,j,k} - 1) \times H,$$
$$\forall i, i' \in I,\ \forall j \in J_i,\ \forall k \in M. \tag{24}$$

Constraint (7) indicates that $O_{i,j}$ is assigned to exactly one machine k. Constraints (8) and (9) establish the sequential processing order for operations $O_{i,j}$ and $O_{i',j'}$ on machine k, with $y_{i,j,i',j',k}$ and $y_{i',j',i,j,k}$ taking opposite values (one is '0', the other is '1'). Constraint (10) states that the completion time of $O_{i,j}$ is at least the sum of the completion and processing time of $O_{i,j-1}$. Constraint (11) defines the relationship between the completion and processing times of two consecutive operations. Constraints (12) and (13) ensure that there is exactly one transportation task that is both the first and last task for AGVa. Constraint (14) requires the completion time of $O_{i,j}$ to be no less than the sum of the arrival and processing times of the AGV. Constraint (15) denotes that, if an AGV is required to transport job i before processing operation $O_{i,j}$, the arrival time for $O_{i,j}$ must be no earlier than the completion time of the previous operation $O_{i,j-1}$, plus the transportation time between the two machines. Constraints (16–(19) define the required working time for AGVa to complete the transportation task in various scenarios, assuming AGVa is responsible for transporting a job before processing an operation. Constraint (20) ensures that, if $O_{i,j}$ and $O_{i',j'}$ are consecutive tasks for AGV a, the processing time of $O_{i,j}$ is at least the sum of the processing time of $O_{i',j'}$ and the working time of AGVa. Constraint (21) ensures that, if $O_{i',j'}$ is the first task for AGVa, its processing time must be at least the working time of AGVa. Constraint (22) denotes the relationship between the completion time and the makespan. Constraint (23) calculates the TP_k. Constraint (24) represents the relationship among TI_k, completion time and processing time.

3.2. Definition of the objective and feature space

See Dai *et al.* (2019), both makespan and energy consumption are defined as optimization objectives. Feature space $\mathcal{B}$ is defined using two key features: the number of machine idles (NI) and the number of AGV transfers (NT). These features are chosen because, along with processing time, they significantly impact energy consumption and makespan (Qin *et al.* 2024). Specifically, excessive machine idling

Table 3. The processing time of operations on machines.

		Available machines					Available machines		
Job	Operation	M1	*M2*	M3	Job	Operation	*M1*	M2	M3
J_1	$O_{1,1}$	8	10	9	J_4	$O_{4,1}$	6	6	–
	$O_{1,2}$	–	2	3		$O_{4,2}$	3	2	3
	$O_{1,3}$	4	–	4		$O_{4,3}$	–	5	7
J_2	$O_{2,1}$	7	10	6		$O_{4,4}$	2	4	5
	$O_{2,2}$	5	8	–	J_5	$O_{5,1}$	12	6	8
J_3	$O_{3,1}$	4	4	4		$O_{5,2}$	5	3	3
	$O_{3,2}$	–	–	5		$O_{5,3}$	2	2	–
	$O_{3,3}$	–	3	3					

Table 4. AGV transportation time between machines.

Layout	Depot	M1	M2	M3
Depot	0	2	3	4
M1	2	0	2	4
M2	3	2	0	2
M3	4	4	2	0

leads to unnecessary delays, while more job transfers introduce additional lag, both of which contribute to inefficient resource utilization (Pan *et al.* 2022). Additionally, concentrating all operations on a single machine without transfers could leave other machines idle, leading to an unreasonably prolonged makespan. Thus, balancing machine idle times and job transfers is critical for optimizing ITPS. If the feature space is unbounded, the computational burden becomes prohibitively high. To avoid the computational burden of an infinite feature space, the lower bound (lb) for both NI and NT is set to zero. The upper bound (ub) for NI corresponds to the total number of operations ($n_{\max}$), representing the maximum possible number of machine idles. For NT, the upper bound ub_t is set to $2 \times n_{\max}$, based on the assumption that each AGV transport involves a round trip to and from the depot. Both NI and NT are constrained within the range $[lb, ub]$. Consequently, the feature space is discretized into $(ub_i - lb) \times (ub_t - lb)$ niches, with the goal of identifying the PS for each niche. These bounds can be adjusted based on the problem scale or decision-maker preferences.

3.3. Illustrative example of ITPS

The following example further illustrates ITPS and identifies the corresponding objective function values, decision space and feature space in the context of multi-objective QD optimization.

Consider a factory with m machines, where the set of machines is $M = \{1, 2, 3\}$. In this example, there are $n = 5$ jobs, denoted as $I = \{1, 2, 3, 4, 5\}$, each consisting of a set of w_i operations: $J_1 = \{O_{1,1}, O_{1,2}, O_{1,3}\}$, $J_2 = \{O_{2,1}, O_{2,2}\}$, $J_3 = \{O_{3,1}, O_{3,2}, O_{3,3}\}$, $J_4 = \{O_{4,1}, O_{4,2}, O_{4,3}, O_{4,4}\}$, $J_5 = \{O_{5,1}, O_{5,2}, O_{5,3}\}$. Each operation $O_{i,j}$ must be processed on one of the available machine set. In this factory, there are two AGVs whose task is to transport all jobs from the depot to different machines and to change the machines for processing. It is assumed that, when the machines are idle, their energy consumption is 1 kW per unit time, while the AGVs consume 3 kW per unit time during transportation, and the machines consume 6 kW per unit time during operation. The processing time of $O_{i,j}$ on different machines is predetermined. Similarly, the transportation time of the AGV to various machines is also fixed. These values are presented in Tables 3 and 4.

As shown in Figure 2, the operations within each job are represented consistently using identical labels . The AGV in a no-load state is indicated by the corresponding annotation in the figure. For machines M1, M2 and M3, the blank intervals represent their idle times. The objective functions optimized in this example are makespan $C_{\max}$ and total energy consumption (TEC). After computation,

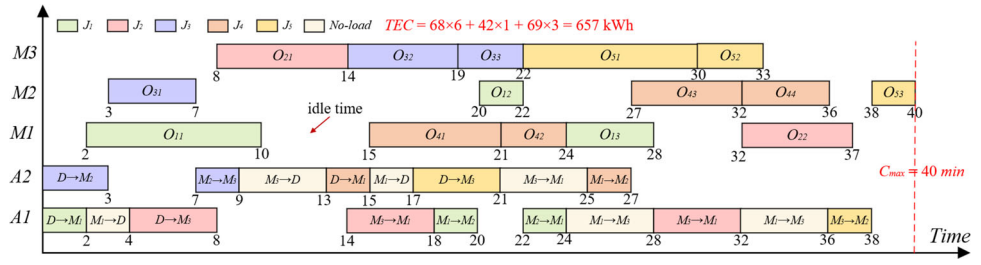


Figure 2. Gantt chart of the example.

their values are 40 minutes and 657 kWh, respectively. The decision space consists of all possible combinations of operation sequence, machine assignment and AGV allocation vectors. The feature space is defined by two features: the number of machine idle times and the number of job transportations. From the Gantt chart, it can be observe that the total machine idle time is 8, and the total number of job transportations is 11. Therefore, the position $\boldsymbol{b}_x$ in the feature space is represented as (11, 8).

3.4. Critical path

The critical path refers to the longest sequence of operations in the entire processing schedule (R. Li, Gong, L. Wang, *et al.* 2023; Y. Wang *et al.* 2023). Operations along this path are termed critical operations, and no idle time is allowed between adjacent critical operations. In this work, the length of the critical path is equivalent to the makespan. To reduce the makespan in ITPS, the length of the critical path can be minimized by adjusting the order of items in FA, OS and MS vectors.

4. Knowledge-driven multi-objective MAP-elites

The DMOME algorithm maintains a multi-dimensional archive, where each niche in the feature space $\mathcal{B}$ stores its corresponding Pareto solution set. An overview of the workflow is illustrated in Figure 3, and the subsequent subsections detail its implementation.

4.1. Initialization and selection

A grid $\mathcal{A}$ of size $(ub_i\text{-}lb) \times (ub_t\text{-}lb)$ is created. As described in Section 3, $\mathcal{A}$ is a two-dimensional matrix where all niches are initially empty, indicating that no solutions are stored. *ps* solutions are randomly generated and added to $\mathcal{A}$, ensuring their features lie within the bounds defined by the lower and upper limits. This work still uses the initialization strategy of traditional QDs, whose performance relies heavily on the proposed strategy. Each solution x is represented by the concatenation of three encoded vectors: (1) operation sequence (OS); (2) machine assignment (MA); and (3) AGV allocation (AS); each having a length equal to the total number of operations, $n_{\max}$. The detailed encoding rules are provided in the appendix. The proposed DMOME algorithm employs two complementary strategies for selecting solutions to undergo variation. One solution is chosen from the global Pareto set in $\mathcal{A}$ to incorporate high-quality, non-dominated solutions. The other is randomly selected from a niche within $\mathcal{A}$ to encourage exploration of diverse and under-explored regions.

4.2. Evaluation and update of feature space

The framework of the DMOME algorithm is presented in Algorithm 1. The algorithm first initializes an empty feature–performance grid $\mathcal{A}$ containing the performance archive $\mathcal{P}$ and the solution archive $\mathcal{X}$ (Line 1). Then, *ps* solutions are randomly generated and added into grid $\mathcal{A}$ (Line 2). Subsequently, two neural networks $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$ are created using the input hyperparameters,

Algorithm 1: The framework of the DMOME algorithm

Input: *batch*, α, γ, ϵ, S_E and *Epoch*
Output: *grid* $\mathcal{A}$ ($\mathcal{P}$ *and* $\mathcal{X}$)
1: $\mathcal{P} \leftarrow \emptyset$; $\mathcal{X} \leftarrow \emptyset$
// *Create* $\mathcal{A}$ *with performances* $\mathcal{P}$ *and solution set* $\mathcal{X}$
2: Randomly initialize *ps* solutions and add them into grid $\mathcal{A}$
3: Create networks $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$ using input parameters
4: **while** *the termination condition is not met* **do**
// *The innovations presented in this article are as follows:*
5: **for** $i \leftarrow 1$ **to** *batch* **do**
6: Find the global Pareto solution set Γ in $\mathcal{X}$
7: $x \leftarrow$ random_solution(Γ)
8: $x' \leftarrow$ random_solution($\mathcal{X}$)
9: $\{y, z\} \leftarrow$ Variation(x, x')
10: Select the higher-quality solution between y and z as x
11: $\{\mathcal{P}, \mathcal{X}, *\} \leftarrow$ Add_to_grid($\mathcal{P}, \mathcal{X}, x$) // *Algorithms 2and3*
12: x serves as state s_t and is input to $Q_{\text{val}}(\theta_1)$
13: **if** *rand* $< \epsilon$ **then**
14: Select action a_t with the max value from $Q_{\text{val}}(\theta_1)$
15: **else**
16: Randomly select an action a_t from all possible actions
17: **end if**
18: Execute an enhanced operator according to a_t, x'' is generated // *Algorithm 4*
19: x'' serves as the next state s_{t+1}
20: $\{\mathcal{P}, \mathcal{X}, r_t\} \leftarrow$ Add_to_grid($\mathcal{P}, \mathcal{X}, x''$)
21: Store transaction (s_t, a_t, r_t, s_{t+1}) in experience pool
22: Train the $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$ // *Algorithm 5*
23: $\epsilon \leftarrow \max(\epsilon_{\min}, \lambda_\epsilon \cdot \epsilon)$
24: **end for**
25: **end while**
return *grid* $\mathcal{A}$ ($\mathcal{P}$ *and* $\mathcal{X}$)

including the learning rate α, discount factor γ, exploration rate ϵ, replay buffer S_E and the number of epochs for training (Line 3). The main evolutionary process is executed within a loop that continues until the predefined termination condition is reached (Line 4). Here, the termination criterion is controlled by the hyperparameter *Max_ter*, which denotes the maximum number of solution evaluations, set as $Max\_Iter = 10 \times m \times \sum_1^n w_i$, where m is the number of machines and $\sum_1^n w_i$ is the total number of operations. After selecting two solutions, variation is applied (Line 9, Algorithm 1). This work introduces the Precedence Operation Crossover (POX) operator for the OS vector and the Uniform Crossover (UX) operator for the MA vector (R. Li, Gong, L. Wang, *et al.* 2023). Unlike previous studies, this article also designs POX and UX operators for adjusting the AS vector, similar to the adjustments made for OS and MA. Owing to space restrictions and the fact that this strategy is not the core focus of this article, its details are not provided. Readers who are interested may refer to the original article.

The evaluation step calculates the makespan and energy consumption of the newly generated solution and records its corresponding coordinates (NT, NI) in $\mathcal{A}$. As shown in Algorithm 2, x represents the solution to be evaluated, concatenated by vectors OS, MA and AS. $f_1(x)$ and $f_2(x)$ represent the objectives to be optimized (makespan and energy consumption). NI denotes the number of AGV transportation tasks, while NI also represents the machine idle times. For clarity, this article also defines five additional symbols: $Start_{i,j}$, P_k, AT_a, AP_a and $U_{i,j}$. $Start_{i,j}$ refers to the start time of operation $O_{i,j}$, P_k represents the availability set of machine k, AT_a denotes the cumulative transportation time of AGVa, AP_a indicates the current location of AGVa, and $U_{i,j}$ represents the machine processing operation $O_{i,j}$. Lines 3–4 extract operations and assign machines and AGVs. Lines 6–13 calculate the AGV energy consumption for transporting the first operation, as well as the machine idle

Algorithm 2: *Evaluation* (x)

Input: x
Output: $\{f_1(x), f_2(x), \boldsymbol{b}_x\}$
1: $NI \leftarrow 0;\ NT \leftarrow 0$
2: **for** $pos \leftarrow 1$ **to** $n_{\max}$ **do**
3: $AT_a \leftarrow 0;\ AP_a \leftarrow 0; O_{i,j} \leftarrow\ OS_{pos}$
4: Extract $U_{i,j}$, a from MS and AS that OS_{pos} is assigned
5: **if** $O_{i,j}$ is the first operation **then**
6: $AT_a \leftarrow TR_{AP_a,0} + TR_{0,U_{i,j}} + AT_a$
7: Calculate the NT and the energy consumption of the AGV
8: $AP_a \leftarrow U_{i,j}$
9: **if** $AT_a > P_{U_{i,j}}$ **then**
10: Calculate the energy consumption of the idle machine
11: $NI \leftarrow NI + 1$
12: **end if**
13: $AT_a \leftarrow max(AT_a, P_{U_{i,j}});\ Start_{i,j} \leftarrow AT_a$
14: **else**
15: $AT_a \leftarrow max(AT_a + TR_{AP_a,U_{i,j-1}},\ C_{i,j-1})$
16: $AT_a \leftarrow AT_a + TR_{U_{i,j-1},U_{i,j}}$
17: Calculate the NT and energy consumption of the AGV
18: **if** $AT_a > P_{U_{i,j}}$ **then**
19: Calculate the energy consumption of the idle machine
20: $NI \leftarrow NI + 1$
21: **end if**
22: $Start_{i,j} \leftarrow max(AT_a, P_{U_{i,j}});\ AP_a \leftarrow U_{i,j}$
23: **end if**
24: $C_{i,j} \leftarrow Start_{i,j} + T_{i,j,U_{i,j}};\ P_{U_{i,j}} \leftarrow C_{i,j}$
25: Calculate the processing energy consumption
26: **end for**
27: Calculate $f_1(x)$ and $f_2(x)$
28: $\boldsymbol{b}_x \leftarrow (NT, NI)$
return $\{f_1(x), f_2(x), \boldsymbol{b}_x\}$

energy consumption. Lines 15–22 compute the AGV energy consumption and machine idle energy consumption for subsequent operations. Line 25 calculates the energy consumption for machine processing. Line 27 determines the final completion time $f_1(x)$ and total energy consumption $f_2(x)$. Finally, Line 28 records the coordinates $\boldsymbol{b}_x$ of solutions in the feature space.

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

Input: $\mathcal{P}, \mathcal{X}, x$
Output: feature-performance grid $(\mathcal{P}, \mathcal{X})$ and reward r_t
1: $\{f_1(x), f_2(x), \boldsymbol{b}_x\} \leftarrow$ *Evaluation* (x) // *Algorithm 2*
2: **if** $\mathcal{P}(\boldsymbol{b}_x) = \emptyset$ **then**
3: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow \{f_1(x), f_2(x)\};\ \mathcal{X}(\boldsymbol{b}_x) \leftarrow x;\ r_t \leftarrow 1$
4: **else**
5: **if** $(\mathcal{X}(\boldsymbol{b}_x) \prec x$ *or* $\mathcal{X}(\boldsymbol{b}_x) \prec_d x)$ *and* $|\mathcal{X}(\boldsymbol{b}_x)| < \delta$ **then**
6: $\mathcal{P}(\boldsymbol{b}_x) \leftarrow \{f_1(x), f_2(x)\};\ \mathcal{X}(\boldsymbol{b}_x) \leftarrow x;\ r_t \leftarrow 1$
7: **else if** $(\mathcal{X}(\boldsymbol{b}_x) \prec x$ *or* $\mathcal{X}(\boldsymbol{b}_x) \prec_d x)$ *and* $|\mathcal{X}(\boldsymbol{b}_x)| \geq \delta$ **then**
8: Find the solution with the minimum HV value and record it as $\kappa;\ \mathcal{X}^{[\kappa]}(\boldsymbol{b}_x) \leftarrow x;\ r_t \leftarrow 0.8$ // $[\kappa]$ represents the κth solution that should be replaced
9: **else**
10: $r_t \leftarrow 0$
11: **end if**
12: **end if**
return $\{\mathcal{P}, \mathcal{X}, r_t\}$

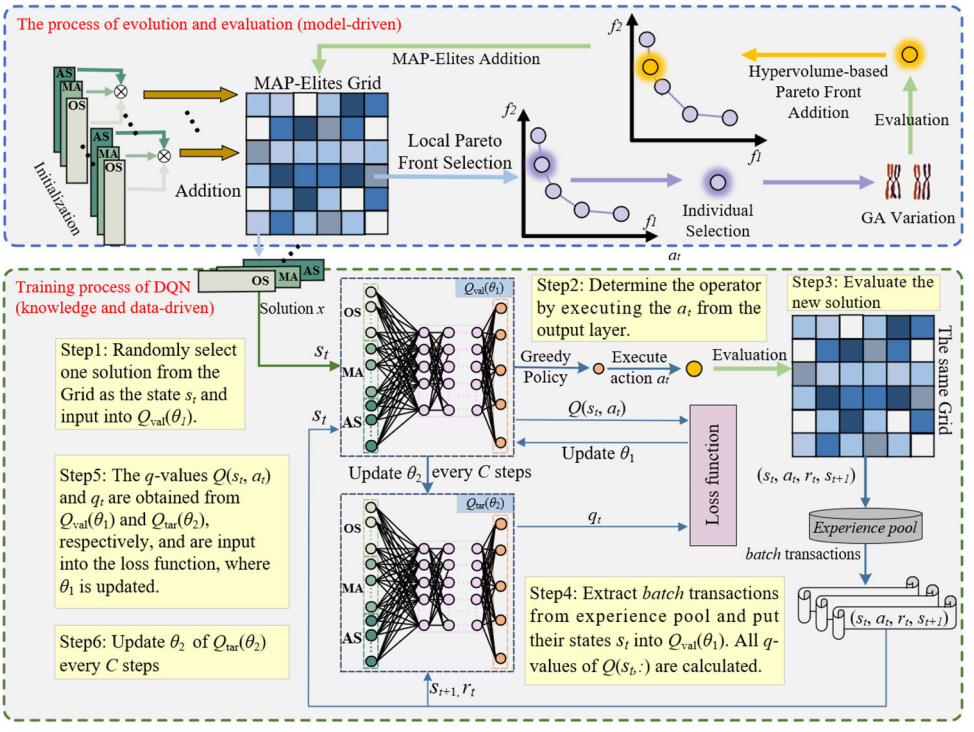


Figure 3. *Workflow of the DMOME algorithm.* The upper part of the figure depicts the optimization process driven by the QD optimization model. The lower part of the figure indicates the DQN training process, which includes both the knowledge-driven enhanced search strategy and the DQN selection mechanism. The evolutionary process consists of the following steps: initialization; selection and variation; evaluation and update. The DQN training process involves several stages. First, solutions are fed into the network. Next, an enhanced operator is selected for execution. The newly generated solution is then evaluated, and the grid is updated accordingly. Transactions are extracted from the experience pool and, finally, the loss is minimized by reducing the difference between the valuation network and the target network Q-values.

After the evaluation, the next step requires determining whether the solution should be added to $\mathcal{A}$. Algorithm 3 outlines the update procedure in detail. In Line 3, if $\mathcal{P}(\boldsymbol{b_x})$ is empty, the solution x is added to $\mathcal{A}$, and the reward r_t (used for DQN training) will be set to one. If $\mathcal{P}(\boldsymbol{b_x})$ is occupied, the number of solutions in $\boldsymbol{b_x}$ is evaluated. If x either dominates (denoted by $\prec$) or is non-dominated (denoted by $\prec_d$) compared to all existing solutions in $\mathcal{A}$, and the number of solutions in $\boldsymbol{b_x}$ is less than P, x is added to $\mathcal{A}$ with a reward of one (Line 6). If the number of solutions in $\boldsymbol{b_x}$ exceeds P, all solutions within $\boldsymbol{b_x}$ are evaluated using the Hypervolume (HV) metric. The solution with the lowest HV contribution is replaced by x, and a reward of 0.8 is assigned (Line 8). Otherwise, a reward of zero is given (Line 10), indicating that the existing solutions in $\boldsymbol{b_x}$ dominate the newly generated solution x. This process is illustrated more intuitively in the upper part of Figure 3.

4.3. Knowledge-driven enhanced search strategy

This article incorporates the knowledge of AGV transportation and machine idling into the evolutionary process to exploit the problem characteristics more effectively. In addition to utilizing domain knowledge in multi-objective QD optimization, the proposed strategy also incorporates heuristic rules specifically tailored to AGV transportation. The results demonstrate that prioritizing operations with the longest transportation times for adjustment significantly improves performance. This prioritization is applied before making changes to the OS, MA and AS vectors. Search operators that

Algorithm 4: Knowledge-driven enhanced search strategy

Input: solution x and performance grid $\mathcal{P}$
Output: new solution x'
1: Encode x as OS, MA and AS
2: Calculate the critical path *ope_path*
3: Randomly select a job i from *ope_path*
4: Sort all operations within i in descending order of AGV transportation time
5: Identify the operation $O_{i,j}$ with the longest AGV transportation time
6: **if** the MA vector requires adjustment **then**
7: Change the machine for $O_{i,j}$ // *ES*2
8: **else if** the OS vector requires adjustment **then**
9: Randomly select an operation and swap it with $O_{i,j}$ // *ES*3
10: **else if** the AS vector requires adjustment **then**
11: Change the AGV for $O_{i,j}$ // *ES*4
12: **else if** a reordering of the OS vector is required **then**
13: Randomly select a job, find the operation with the longest transportation time, and swap it with $O_{i,j}$ // *ES*5
14: **else**
15: Select $O_{i,j}$ from all jobs and change its machine // *ES*1
16: **end if**
17: Generate a new solution x'
return x'

employ this rule outperform those that do not. The following elaborates on these enhanced search operators.

The process begins by selecting a critical job from the critical path and identifying the operation within the job that has the longest transportation time. Then, the machine assignment, AGV allocation, or position of this critical operation is adjusted. The intrinsic difference of the proposed ES from existing search strategies is the incorporation of AGV transportation heuristic rules. Unlike random selection, the adjustments are guided by the transportation time of the AGV. The proposed *ES*2–*ES*5 are unified into a single framework that handles perturbations of the OS, MA and AS vectors. Additionally, *ES*1 operates independently of the critical path but still incorporates AGV transportation heuristic rules.

As outlined in Algorithm 4, adjustments to the MA, OS and AS vectors follow a uniform process. Each step begins by selecting a job i from the critical path (Line 3) and identifying the operation within the job with the longest AGV transportation time (Lines 4 and 5). Once the operation $O_{i,j}$ is identified, the ES is chosen based on the action a_t determined by the DQN (Lines 6–15). Details of the DQN's selection and training process are provided in the next subsection. If a vector is updated, it is concatenated with the other two vectors to generate a new solution x'.

4.4. Deep-Q-networks selection mechanism

While enhanced search operators effectively utilize domain-specific knowledge, they face limitations such as blind and unpredictable operator selection, relying on static heuristics without learning from historical or current data. This often leads to suboptimal decisions and reduces efficiency in dynamic or complex environments. To address this issue, this work proposes integrating a DQN selection mechanism into the selection process. By leveraging RL, the DQN adapts operator selection based on accumulated learning and real-time feedback, reducing randomness and improving search efficiency and convergence in multi-objective QD optimization. In DQNs, a transaction (s_t, a_t, r_t, s_{t+1}) is composed of the current state s_t, action a_t, reward r_t and the next state s_{t+1}. Their specific definitions are as follows.

- *State.* This article defines a complete solution (concatenated by the OS, MA and AS vectors) as the state s_t and puts it into the input layer of the DQN.

- *Action.* The DQN selects the appropriate action to execute based on historical experience and the current environment. Actions are defined as the enhanced search operators, and in each iteration the DQN determines which action to perform to modify the encoded vector. To balance exploration and exploitation, the policy is executed using an epsilon-greedy strategy: a higher epsilon is adopted in the early stages to encourage sufficient exploration of the ES operators, while epsilon gradually decays as learning progresses so that the policy can be increasingly exploited once the Q-network becomes more reliable.
- *Reward.* As outlined in the addition for the feature space $\mathcal{B}$, the corresponding feedback rewards differ based on specific conditions. Notably, r_t is not calculated during the evaluation and update of the $\mathcal{B}$ when generating initial solutions or applying the *V ariation* strategy, as the DQN is not involved in these situations. When the *V ariation* strategy generates a new solution x, the reward is produced only when x is added to the feature grid. If the feature cell corresponding to $\boldsymbol{b}_x$ is empty, the solution x is directly inserted and a reward of $r_t = 1$ is assigned. If the cell already contains solutions but is not yet full, and x either dominates or is non-dominated while providing an improvement based on the dominance relation, then x replaces the previous content of that cell and again yields $r_t = 1$. When the cell is full, but x still provides an improvement according to the dominance criterion, the solution with the minimum hypervolume contribution in that cell is replaced by x, and a smaller reward of $r_t = 0.8$ is assigned. In contrast, if x does not improve the feature cell under any of the above conditions, no replacement occurs and the reward is set to $r_t = 0$.
- *Transition.* After the execution of action a_t, a new solution (state s_{t+1}) is generated. The state completes the transition.
- *Network structure.* Following R. Li, Gong, L. Wang, *et al.* (2023), the valuation network $Q_{\text{val}}(\theta_1)$ and the target network $Q_{\text{tar}}(\theta_2)$ adopt the same architecture. Both networks consist of six fully connected layers, with Rectified Linear Unit (ReLU) activation functions between adjacent layers. $n^{[k]}$ represents the input dimension of the kth layer, while $m^{[k]}$ denotes its output dimension. The network is structured as follows: $n^{[1]}$: *In*, $m^{[1]}$: *128*; $n^{[2]}$: *128*, $m^{[2]}$: *256*; $n^{[3]}$: *256*, $m^{[3]}$: *128*; $n^{[4]}$: *128*, $m^{[4]}$: *64*; $n^{[5]}$: *64*, $m^{[5]}$: *32*; and $n^{[6]}$: *32*, $m^{[6]}$: *Out*. Here, *In* refers to the total dimensionality of the OS, MA and AS vectors, while *Out* represents the number of enhanced search operators.
- *Training of the DQN.* The inputs include the number of transactions extracted *batch*, the valuation network $Q_{\text{val}}(\theta_1)$, the target network $Q_{\text{tar}}(\theta_2)$, and the parameters γ , *EP* and *Epoch*. The outputs are the updated $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$. *batch* transactions, including the corresponding s_t, a_t, r_t and s_{t+1}, are extracted from the experience pool. Then, the q-values for $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$ are computed, with q_t for $Q_{\text{tar}}(\theta_2)$ calculated using Equation (25):

$$q_t = r_t + \gamma \times \max Q\left(s_{t+1}, :\right). \tag{25}$$

  At time step t, r_t represents the immediate reward. The target q-value q_t is computed using the discount factor γ , which balances immediate and future rewards to determine the significance of long-term outcomes in current decision-making. $\max Q(s_{t+1}, :)$ denotes the maximum expected return over all possible actions in the subsequent state s_{t+1}.
  Subsequently, the $Q(s_t, a_t)$ value of $Q_{\text{val}}(\theta_1)$ is computed. Next, the difference between q_t and $Q(s_t, a_t)$ is calculated using the loss function defined in Equation (26):

$$J\left(\theta_1\right) = \frac{1}{batch} \times \sum_{t=1}^{batch} \left(Q\left(s_t, a_t\right) - q_t\right)^2. \tag{26}$$

  The goal of training the DQN is to minimize the loss function $J(\theta_1)$. $J(\theta_1)$ measures the average squared difference between $Q(s_t, a_t)$ and q_t across a batch of training samples. The calculations for $Q(s_t, a_t)$ and q_t have been provided earlier.
  After calculating the loss, the parameters θ_1 in $Q_{\text{val}}(\theta_1)$ are updated. When the learning steps exceed the hyperparameter *EP*, θ_2 in $Q_{\text{tar}}(\theta_2)$ is replaced by θ_1.

For clarity, several symbols are defined as follows: the term *batch* denotes the minibatch size sampled from the replay buffer; α represents the learning rate of the optimizer; γ is the discount factor in reinforcement learning; ϵ is the exploration probability used in the ϵ-greedy action selection strategy; S_E denotes the replay memory that stores past transitions (s_t, a_t, r_t, s_{t+1}); and the hyperparameter *Epoch* indicates the number of gradient-update iterations performed during each training call of the DQN. The pseudo-code for DQN training is given by Algorithm 5.

Algorithm 5: The training process of the DQN

Input: *batch*, $Q_{\text{val}}(\theta_1)$, $Q_{\text{tar}}(\theta_2)$, γ , *EP* and *Epoch*
Output: $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$
for *epoch* $\leftarrow 1$ **to** *Epoch* **do**
  Randomly sample *batch* transactions from S_E
  Extract s_t, a_t, r_t, s_{t+1} from the transactions
  Compute target q-value q_t using $Q_{\text{tar}}(\theta_2)$ (Equation 25)
  Compute predicted $Q(s_t, a_t)$ using $Q_{\text{val}}(\theta_1)$
  Calculate the loss (Equation 26)
  Update θ_1 by minimizing the loss with the *Adam* optimizer
  if *step count* $>$ *EP* **then**
    $\theta_2 \leftarrow \theta_1$
  end if
end for

4.5. Computational complexity analysis of the DMOME algorithm

The time complexity of the DMOME algorithm can be analysed by examining the key operations within its iterative process. The main loop (Lines 4–24, Algorithm 1) runs until a termination condition is met, which is typically dependent on the number of evaluations and iterations. Within each batch iteration, the algorithm performs several computationally intensive operations. The Add_to_Grid function (Algorithm 3) involves decoding the solution, evaluating objective functions and updating the grid. The worst-case scenario includes finding and replacing the solution with the minimum HV value, which requires scanning the grid. If the grid size is bounded by G, this step contributes $O(G)$ complexity. The knowledge-driven enhanced search strategy (Algorithm 4) introduces additional complexity owing to the critical path calculation, sorting operations by AGV transportation time, and various job or machine adjustments. Sorting operations in descending order of AGV transportation time contribute $O(nlogn)$, where n is the number of operations. The machine and AGV modifications introduce additional constant-time operations, making the overall complexity of this search strategy approximately $O(nlogn)$. Given that the termination condition is based on evaluations, the worst-case complexity depends on the total number of function evaluations E, leading to an estimated upper bound of $O(E \times (G + nlogn))$.

5. Experiments

This section evaluates the proposed method in a real-world setting: a mechanical processing workshop in Nanjing, People's Republic of China (Dai *et al.* 2019). This article tested 18 problem instances with varying job and machine configurations. Each instance was independently run 20 times. The termination criterion was set to $Max\_Iter = 10 \times m \times \sum_1^n w_i$, where m is the number of machines and w_i is the number of operations in job i. The job count range was $\{20, 30, 40, 50, 80, 100\}$ and the machine count range was $\{5, 8, 10\}$. Each job contained five operations with processing times uniformly drawn from $[5, 40]$, and AGV transportation times from $[1, 5]$. These values remained fixed across all runs. The energy consumptions were as follows: the per-unit processing energy per machine

was {20, 15, 6, 12, 10, 5.5, 7.5, 3, 5.5, 10}, the idle energy per unit time was one, and the AGV transportation energy was 3.5. All experiments were conducted on a 64-bit Windows® 11 system with an Intel® Core™ i7-13790F @2.10 GHz and 16 GB RAM, implemented in Python® using PyCharm 2023.

5.1. Evaluation metrics

Following R. Li, Gong, L. Wang, *et al.* (2023) and Y. Wang *et al.* (2023), this article uses three multi-objective optimization metrics to evaluate the performance of comparative algorithms: Generational Distance (GD), Inverted Generational Distance (IGD) and Hypervolume (HV). Since there is a significant difference between the values of makespan and TEC, both metrics are normalized before applying. It is worth noting that, since the problem is a large-scale NP-hard problem, the true Pareto optimal set is unknown. Therefore, in this article, the Pareto front P^* obtained by all algorithms is used as a reference ideal Pareto optimal set.

GD is a commonly used metric for evaluating the convergence of the solution set. Specifically, it measures the average minimum distance from each point a in the solution set P_g of algorithm g to its nearest point in the reference set P^*. Smaller values of GD denote better convergence performance.

$$GD_g\left(P_g, P^*\right) = \frac{\sqrt{\sum_{\zeta \in P_g} \min_{\xi \in P^*} dis\left(\zeta, \xi\right)^2}}{|P_g|}, \tag{27}$$

where $dis(\zeta, \xi)^2$ denotes the square of the minimum Euclidean distance from ζ to ξ. $|P_g|$ represents the number of solutions in P_g.

IGD functions as a comprehensive metric that assesses both the convergence and distribution of the solution set P_g. Specifically, it is defined as the average Euclidean distance from each reference point ξ in P^* to its closest solution ζ in P. A smaller IGD value indicates superior overall performance of the algorithm.

$$IGD_g\left(P_g, P^*\right) = \frac{\sum_{\xi \in P^*} \min_{\zeta \in P_g} dis\left(\zeta, \xi\right)}{|P^*|}, \tag{28}$$

where $dis(\zeta, \xi)$ indicates the minimum Euclidean distance from ζ to ξ. $|P^*|$ represents the number of solutions in P^*.

HV provides a comprehensive evaluation of both the convergence and distribution of P_g. It quantifies the volume of the objective space enclosed by the non-dominated solutions produced by algorithm g and a reference point re. This article sets re as (1.1, 1.1). A higher value of HV_g indicates that the solution set generated by algorithm g demonstrates better overall performance.

$$HV_g\left(P_g, re\right) = \mathcal{L}\left(\bigcup_{\zeta \in P_g} \{\xi | \zeta \prec \xi \prec re\}\right), \tag{29}$$

where $\mathcal{L}(\cdot)$ indicates the Lebesgue measure of P_g, and ζ refers to an element in P_g. $\bigcup_{\zeta \in P_g} \{\xi | \zeta \prec \xi \prec re\}$ signifies that ξ is strictly greater than ζ and ξ is strictly less than the reference point re.

Given the large difference between the two objective values, normalization is needed. This process changes the objective values into a standard range between zero and one.

$$f_i^{\text{norm}}(x) = \frac{f_i(x) - \min\left\{f_i\right\}}{\max\left\{f_i\right\} - \min\left\{f_i\right\}}, \quad i = 1, 2, \tag{30}$$

where $\max\{f_i\}$ and $\min\{f_i\}$ represent the maximum and minimum values of the ith objective across all solutions.

To ensure a fair comparison, this work uses the number of evaluations as the termination criterion instead of runtime. Indeed, the introduction of the DQN has increased the runtime of QD. The preliminary tests show that the DQN takes approximately 200–250 seconds on average across all instances, which is three to five times longer than traditional evolutionary algorithms—*e.g.* 50–60 seconds for INSGA-II (X. Zhou *et al.* 2023) and 60–70 seconds for MOEAD (Q. Zhang and Li 2007). However, it is still faster than other DQN-enhanced EAs, such as the state-of-the-art DQN-based Memetic Algorithm (DQNMA) (F. Zhang, Li, and Gong 2024), which requires around 250–300 seconds. Overall, the times are within acceptable limits.

5.2. Parameter calibration

This study employs the Taguchi method (R. Li, Gong, L. Wang, *et al.* 2023) to calibrate five parameters: batch size (*batch*), learning rate (α), discount factor (γ), greedy factor (ϵ) and experience pool size (*EP*). Four levels were considered for each parameter. $batch \in \{3, 5, 7, 9\}$, $\alpha \in \{0.001, 0.005, 0.01, 0.015\}$, $\gamma \in \{0.8, 0.85, 0.9, 0.95\}$, $\epsilon \in \{0.8, 0.85, 0.9, 0.95\}$ and $EP \in \{10, 30, 50, 70\}$. The Taguchi method uses an orthogonal array $L_{16}(4^5)$, which consists of 16 different parameter combinations. In addition, for all tests, the algorithm generates 100 initial solutions. Figure 4 shows the factor level trend plots for the five parameters, which illustrate the effect of different values on the performance of the DMOME algorithm. As can be seen from Figure 4, the optimal parameter values for the DMOME algorithm were found to be $batch = 5$, $\alpha = 0.01$, $\gamma = 0.85$, $\epsilon = 0.85$ and $EP = 50$. The reasons may be as follows.

- *batch*: the batch size determines how many transitions are sampled from the experience pool and how many solutions are updated in each iteration. For the DMOME algorithm, the optimal batch size is five. A smaller batch size may lead to unstable training owing to insufficient sample diversity, while an excessively large batch size may slow down convergence by reducing the frequency of parameter updates.
- α: the learning rate controls the magnitude of weight updates during training. The optimal value is 0.01. A too small learning rate results in slow convergence, whereas a too large value may cause oscillation or divergence.
- γ: the discount factor determines the relative importance of future rewards. An optimal value of 0.85 is adopted in the DMOME algorithm. A large γ places excessive emphasis on long-term rewards and may slow down learning, while a small γ focuses overly on immediate rewards and may limit long-term performance.
- ϵ: the greedy factor balances exploration and exploitation. A larger ϵ encourages exploration and increases the diversity of visited states, whereas a smaller ϵ promotes exploitation and accelerates convergence. In the DMOME algorithm, an annealing strategy is adopted so that early exploration is gradually reduced, allowing the policy to converge toward more stable action selection in later stages.
- *EP*: the experience pool size determines the amount of past transitions retained for training and influences the distribution of sampled experiences across different stages of the learning process. A larger pool introduces the issue of sample 'age'—older samples promote diversity and generalization by exposing the policy to rarely visited states, whereas recent samples tend to accelerate convergence by focusing on actions aligned with the current policy. In contrast, a smaller pool emphasizes recent transitions, which speeds up convergence but may reduce sample diversity. In the DMOME algorithm, a pool size of 50 is adopted as a compromise that balances sufficient historical diversity with limited influence from outdated experiences.

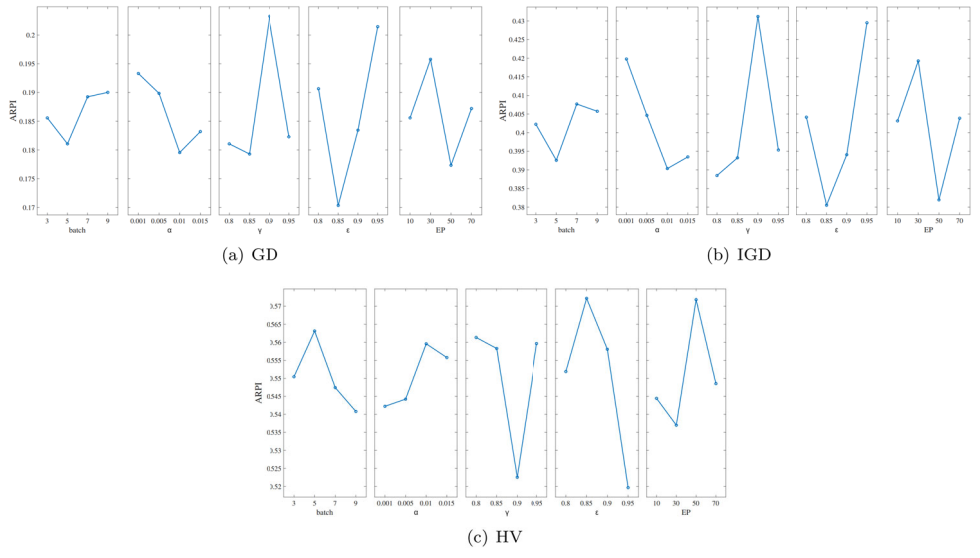


Figure 4. Comparison of variants under GD, IGD and HV metrics.

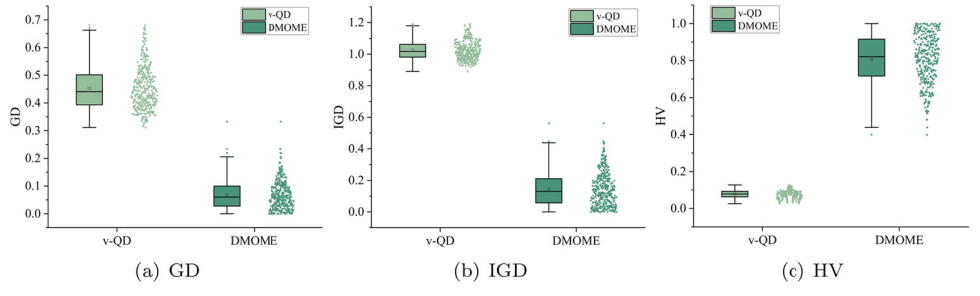


Figure 5. Comparison of the GD, IGD and HV metrics with and without QD optimization models.

5.3. Verification of the QD framework

To assess the improvements in diversity and convergence, this article compared the proposed DMOME algorithm with a baseline multi-objective optimization framework (v-QD) that ignores behavioural features and uses traditional population storage instead of an elite archive. As shown in Figures 5(a)–5(c), the DMOME algorithm outperforms v-QD across all three metrics, with improvements ranging from 84.40% to 90.10%. Both the mean and median values of the DMOME algorithm are consistently better (HV values are opposite).

QD emphasizes both solution quality and diversity, enabling comprehensive exploration of the feature space. This is especially beneficial for large-scale problems, as it identifies high-quality solutions across various niches. By incorporating behavioural features and dynamically adapting the search process, QD improves convergence and resource efficiency. Its flexibility also allows better response to changing problem requirements, making it well-suited for real-world applications like ITPS.

5.4. Verification of the knowledge-driven strategies

The essential difference between the proposed knowledge-driven enhanced search strategy and traditional search strategies lies in the incorporation of AGV transportation heuristic rules to perturb

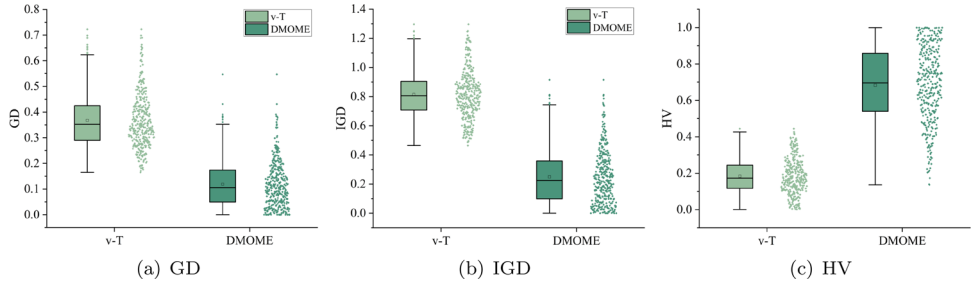


Figure 6. Comparison of the GD, IGD and HV metrics with and without AGV transportation heuristics.

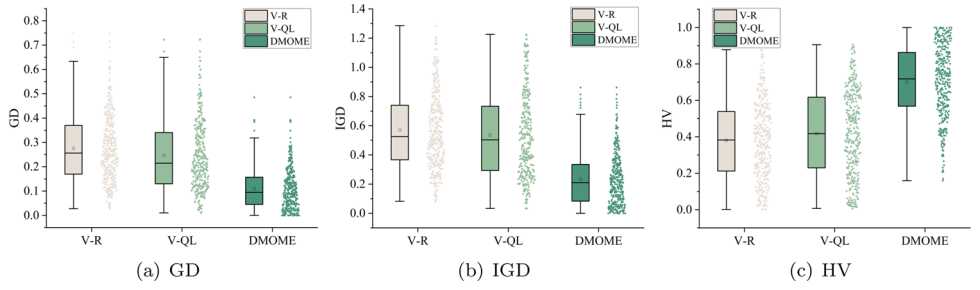


Figure 7. Comparison of the GD, IGD and HV metrics with different selection mechanisms.

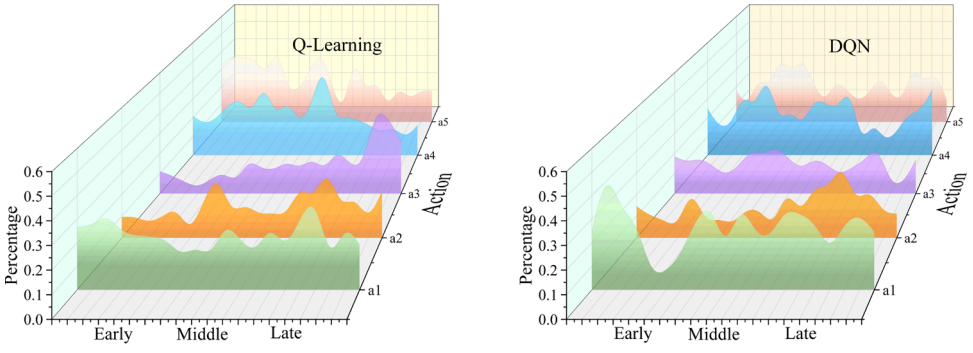
the ordering of the OS, MA and AS vectors. Therefore, to validate the effectiveness of this strategy, this article compares the results of the DMOME algorithm with a version that does not use the AGV transportation heuristic rule (denoted as v-T) across all test instances. As shown in Figure 6, the solutions obtained by the DMOME algorithm outperform those of v-T by approximately 67.6%–73.5% across all metrics.

The knowledge-driven enhanced search strategy significantly improves optimization performance by leveraging domain-specific heuristics, such as AGV transportation rules. These heuristics help exploit problem structures more effectively, guiding the search toward higher-quality solutions and reducing unproductive exploration. By influencing optimization objectives and perturbing the order of solution vectors (OS, MA, AS), the strategy enables more refined adjustments, enhancing both convergence and diversity. Experimental results show that the DMOME algorithm outperforms its variant without the AGV heuristic across GD, IGD and HV metrics, highlighting the value of domain knowledge in improving multi-objective optimization efficiency.

5.5. Verification of the DQN

To demonstrate the effectiveness of the DQN in learning from historical data and guiding the evolution process, this article compares it with widely used Q-learning (Qin *et al.* 2023) (a non-neural network but widely used method) based selection (denoted as v-QL) and random selection (denoted as v-R). It is important to note that, aside from the selection strategy, all other components of the algorithms remain identical across the different variants. As shown in Figure 7, compared to random selection, the DQN-based selection outperforms it by 45.7%–60.7% in terms of the GD, IGD and HV metrics. Furthermore, compared to Q-learning, the DQN-based selection outperforms it by 40%–56.6% on the same metrics.

Figure 8 presents the action selection percentages for Q-learning and the DQN at different evolutionary stages in a larger-scale instance (80_8). These results reveal distinct patterns between the two methods. Early in the evolution, both Q-learning and the DQN show relatively high probabilities



(a) Q-Learning (b) DQN

Figure 8. Percentage of each action selected in different evolutionary stages.

for actions 1 and 4 (*ES*1 and *ES*4). However, as learning progresses, Q-learning increasingly favours actions 2 and 3, while the DQN continues to maintain a higher proportion of actions 1 and 4. In the later stages, actions 1 and 4 dominate in the DQN, while action 3 becomes the primary choice for Q-learning.

This behaviour indicates that the DQN exhibits greater stability in selecting effective search operators, consistently favouring the more beneficial ones—actions 1 and 4—throughout the evolutionary process. In contrast, Q-learning tends to shift toward actions 2 and 3 as the search progresses. The continued preference of the DQN for actions 1 and 4 suggests its ability to exploit prior learning effectively and maintain focus on optimal search strategies. Specifically, actions 1 and 4 correspond to adjustments in machine selection and AGV transportation allocation, respectively. The consistent selection of these actions implies that it successfully leverages knowledge of machine availability and AGV scheduling to guide the search. This targeted behaviour highlights the importance of integrating domain knowledge into the QD optimization process, particularly in terms of efficient machine and AGV allocation, which contributes to generating high-quality solutions.

Following the observed operator selection patterns in Figure 8, additional experiments are conducted to assess the contributions of ES1 and ES4 quantitatively. Specifically, two algorithmic variants were constructed by disabling ES1 and ES4, denoted as N-ES1 and N-ES4, respectively, while keeping all other components unchanged. Table 5 reports the GD, IGD and HV results of these variants across all benchmark instances. The results show that removing either ES1 or ES4 leads to a clear degradation in performance. In most instances, both GD and IGD increase. Meanwhile, HV decreases compared with the DMOME algorithm. This degradation is also reflected in the average results over all instances. These observations provide numerical evidence that ES1 and ES4 play critical roles in improving convergence accuracy and maintaining solution diversity.

Additionally, this work further illustrate the online training process of the DQN on the 80_8 instance by plotting the variation curves for makespan and energy consumption. The results for these two optimization objectives after approximately 8500 training steps are shown in Figures 9(a) and 9(b), respectively. From the training curves, it is clear that both makespan and energy consumption consistently decrease as the number of training steps increases. This indicates that the use of the DQN to select knowledge-driven enhanced search operators effectively improves the performance of multi-objective QD optimization.

The superior performance of the DQN compared to both random selection and Q-learning can be attributed to its ability to learn effectively from historical data and provide feedback to the current environment. Unlike random selection, which lacks any learned knowledge and fails to exploit previous experiences, the DQN makes more informed decisions based on accumulated experience. This

Table 5. Performance impact of removing ES1 and ES4 across all instances.

	GD			IGD			HV		
Instance	N-ES1	N-ES4	DMOME	N-ES1	N-ES4	DMOME	N-ES1	N-ES4	DMOME
20J5M	0.25	0.26	**0.02**	0.15	0.43	**0.00**	0.78	0.43	**0.99**
20J8M	0.19	0.10	**0.04**	0.37	0.20	**0.03**	0.32	0.52	**0.81**
20J10M	0.19	0.47	**0.00**	0.40	0.83	**0.00**	0.46	0.08	**0.96**
30J5M	0.45	0.27	**0.02**	0.77	0.48	**0.00**	0.13	0.38	**0.98**
30J8M	0.24	0.13	**0.00**	0.47	0.29	**0.01**	0.23	0.57	**0.91**
30J10M	0.24	0.20	**0.05**	0.50	0.36	**0.00**	0.28	0.44	**0.83**
40J5M	0.29	0.19	**0.00**	0.61	0.40	**0.03**	0.12	0.37	**0.86**
40J8M	0.23	0.18	**0.01**	0.43	0.43	**0.00**	0.29	0.31	**0.83**
40J10M	0.60	0.21	**0.00**	1.05	0.34	**0.00**	0.02	0.54	**0.98**
50J5M	0.15	0.22	**0.00**	0.43	0.23	**0.01**	0.24	0.44	**0.78**
50J8M	0.39	0.23	**0.00**	0.85	0.53	**0.00**	0.08	0.33	**0.97**
50J10M	0.18	0.12	**0.02**	0.40	0.19	**0.00**	0.33	0.60	**0.92**
80J5M	0.21	0.23	**0.00**	0.68	0.60	**0.00**	0.06	0.12	**0.81**
80J8M	0.19	0.42	**0.01**	0.61	0.70	**0.06**	0.15	0.05	**0.63**
80J10M	0.80	0.33	**0.00**	1.23	0.62	**0.00**	0.00	0.26	**0.99**
100J5M	0.25	0.17	**0.00**	0.58	0.44	**0.02**	0.17	0.31	**0.84**
100J8M	0.26	0.19	**0.01**	0.47	0.44	**0.00**	0.17	0.22	**0.77**
100J10M	0.21	0.14	**0.02**	0.66	0.36	**0.30**	0.03	0.38	**0.72**
Mean	0.30	0.22	**0.01**	0.59	0.44	**0.03**	0.22	0.35	**0.87**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

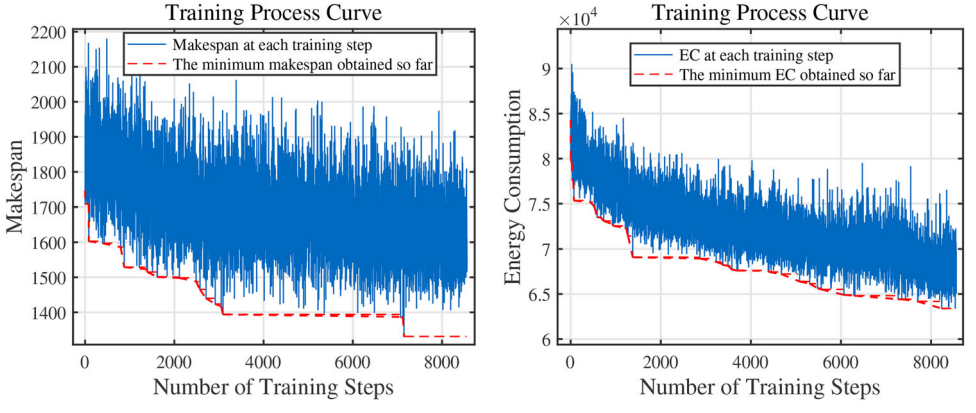


Figure 9. The training process curve of the DQN: (a) makespan at each training step; (b) energy consumption at each training step.

enables better exploration and exploitation, allowing the DQN to optimize long-term rewards. Compared to Q-learning, the DQN benefits further from its deep learning architecture, which enables it to handle large and complex state spaces and generalize more effectively. As a result, the DQN demonstrates improved performance across the GD, IGD and HV metrics.

It is worth noting that the action distribution learned by the DQN is meaningful rather than an artefact of stochasticity. Although different solutions may benefit from different local search operators, they exhibit strong structural regularities. In large instances, many solutions share similar characteristics—such as machine bottlenecks, AGV transportation delays and critical-path patterns—which makes certain operators consistently more effective. Specifically, ES1 and ES4 tend to alleviate these dominant bottlenecks across broad regions of the search space. As a result, the DQN learns a stable preference for these operators, and the observed action trends reflect underlying problem structure rather than random fluctuation. This explains why the action-selection patterns remain coherent throughout evolution, supporting the validity of the analysis presented in this section.

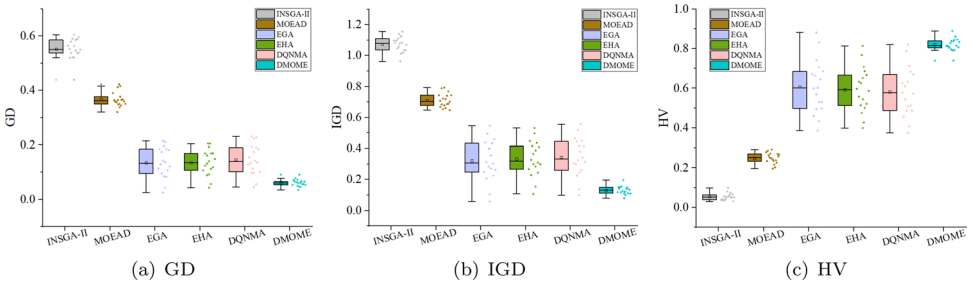


Figure 10. Box plots and data distribution diagrams of GD, IGD and HV for all algorithms.

5.6. Performance comparison and analysis of algorithms

To evaluate the whole performance of the DMOME algorithm further, this article compared it with four widely-used multi-objective optimization algorithms for ITPS, along with one state-of-the-art DQN-based evolutionary algorithm. These include INSGA-II (X. Zhou *et al.* 2023), MOEAD (Q. Zhang and Li 2007), Enhanced GA (EGA) (Dai *et al.* 2019), EHA (Xu, Bao, and Zhang 2023) and the DQNMA (F. Zhang, Li, and Gong 2024). INSGA-II was chosen because it is an improved version of the classic NSGA-II and has been applied to solve ITPS. MOEAD was selected owing to its established reputation in multi-objective optimization. EGA and EHA were included as they are also used for solving multi-objective ITPS. Finally, the DQNMA was selected as it not only solves multi-objective ITPS problems but also represents one of the most innovative algorithms currently, combining DQNs and Memetic Algorithms (MAs). To ensure a fair comparison, all algorithms were calibrated on the same dataset, and their parameter settings were optimized before conducting the performance evaluation.

Table 6 lists the numerical results of all comparison algorithms for the GD, IGD and HV metrics. The last row, labelled 'Mean', represents the average value of these metrics across all test instances. Additionally, a Wilcoxon rank-sum test was conducted to compare the algorithms with the DMOME algorithm. The symbol '†' denotes a statistically significant difference between the comparison algorithm and the DMOME algorithm, with a significance level of 0.05. The best results are highlighted in bold. Figure 10 displays box plots and corresponding data distribution plots for the GD, IGD and HV metrics across all algorithms, illustrating the Pareto front sets obtained.

Based on the GD, IGD and HV values presented in Table 6, it is evident that the DMOME algorithm outperforms all other algorithms, followed by EHA, DQNMA, EGA, MOEAD and INSGA-II. In over 90% of pairwise comparisons, the DMOME algorithm shows a statistically significant advantage over the other algorithms. For the 'Mean' GD and IGD values, the DMOME algorithm consistently outperforms other methods, being approximately 2.2–9.2 times smaller. For the 'Mean' HV value, the DMOME algorithm achieves superior diversity, with values 1.34–16.4 times greater than its counterparts. The above results demonstrate that the proposed algorithm significantly outperforms the comparison algorithms, exhibiting superior convergence and diversity. Figure 10 provides a statistical representation of the overall results, offering a more intuitive comparison that highlights the superior performance of the DMOME algorithm over the other algorithms.

Table 7 presents the results of the Friedman statistical test with a confidence level of $\alpha = 0.050$, including the algorithm rankings and the p-values. The results clearly show that the DMOME algorithm ranks first, followed by EHA, DQNMA, EGA, MOEAD and INSGA-II. The p-values for the GD, IGD and HV metrics are all significantly lower than 0.05, indicating significant differences between the DMOME algorithm and the comparison algorithms.

To provide a more intuitive illustration of the distribution of the global Pareto front across all algorithms, this article considerately chose representative instances of small, medium and large size and plotted their final Pareto fronts (Figure 11). However, these plots only display the global Pareto

Table 6. Numerical results of all comparison algorithms.

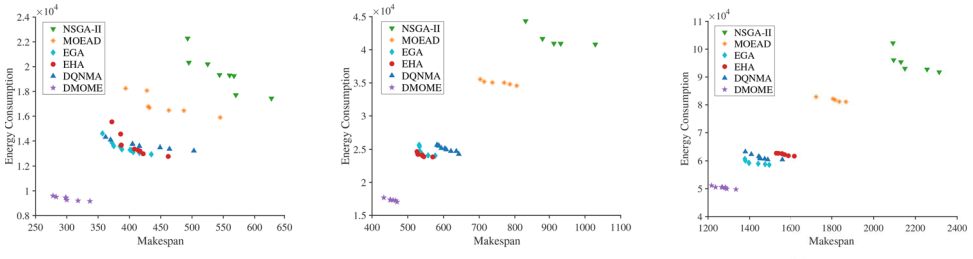
	GD						IGD						HV					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
20J5M	0.44†	0.34†	0.20†	0.19†	0.22†	**0.05**	0.96†	0.72†	0.55†	0.53†	0.56†	**0.11**	0.10†	0.24†	0.39†	0.40†	0.38†	**0.84**
20J8M	0.52†	0.35†	0.17†	0.16†	0.17†	**0.03**	1.01†	0.66†	0.40†	0.39†	0.42†	**0.08**	0.08†	0.29†	0.53†	0.54†	0.51†	**0.89**
20J10M	0.56†	0.42†	0.21†	0.20†	0.23†	**0.09**	1.08†	0.79†	0.49†	0.50†	0.52†	**0.20**	0.06†	0.20†	0.43†	0.43†	0.41†	**0.74**
30J5M	0.52†	0.36†	0.18†	0.17†	0.19†	**0.06**	1.08†	0.74†	0.43†	0.42†	0.44†	**0.12**	0.05†	0.23†	0.50†	0.51†	0.49†	**0.84**
30J8M	0.55†	0.38†	0.18†	0.17†	0.19†	**0.06**	1.06†	0.72†	0.44†	0.44†	0.45†	**0.13**	0.06†	0.25†	0.50†	0.50†	0.49†	**0.84**
30J10M	0.59†	0.41†	0.21†	0.20†	0.23†	**0.05**	1.12†	0.77†	0.46†	0.45†	0.47†	**0.10**	0.04†	0.22†	0.46†	0.48†	0.46†	**0.86**
40J5M	0.54†	0.36†	0.14†	0.14†	0.15†	**0.07**	1.02†	0.68†	0.36†	0.37†	0.38†	**0.15**	0.07†	0.27†	0.57†	0.56†	0.54†	**0.79**
40J8M	0.60†	0.42†	0.18†	0.17†	0.18†	**0.07**	1.14†	0.79†	0.40†	0.41†	0.41†	**0.14**	0.04†	0.20†	0.53†	0.52†	0.51†	**0.83**
40J10M	0.59†	0.37†	0.14†	0.13†	0.14†	**0.06**	1.05†	0.69†	0.30†	0.31†	0.33†	**0.12**	0.05†	0.25†	0.60†	0.60†	0.58†	**0.82**
50J5M	0.54†	0.36†	0.11†	0.11†	0.11†	**0.06**	1.04†	0.70†	0.31†	0.33†	0.34†	**0.15**	0.06†	0.24†	0.60†	0.59†	0.57†	**0.79**
50J8M	0.58†	0.37†	0.13†	0.12†	0.14†	**0.08**	1.10†	0.71†	0.27†	0.28†	0.30†	0.15	0.05†	0.26†	0.66†	0.65†	0.63†	**0.80**
50J10M	0.59†	0.36†	0.12†	0.12†	0.13†	**0.06**	1.12†	0.74†	0.28†	0.30†	0.31†	**0.11**	0.04†	0.23†	0.65†	0.63†	0.61†	**0.85**
80J5M	0.52†	0.35†	0.09†	0.09†	0.10†	**0.07**	1.03†	0.65†	0.23†	0.24†	0.25†	**0.14**	0.07†	0.29†	0.71†	0.69†	0.68†	**0.80**
80J8M	0.55†	0.36†	0.10†	0.11†	0.10†	**0.07**	1.11†	0.69†	0.21†	0.23†	0.22†	**0.14**	0.04†	0.27†	0.74†	0.71†	0.71†	**0.81**
80J10M	0.60†	0.39†	0.11†	0.15†	0.13†	**0.07**	1.16†	0.74†	0.26†	0.30†	0.30†	**0.15**	0.03†	0.23†	0.67†	0.63†	0.63†	**0.81**
100J5M	0.56†	0.34†	0.08†	0.09†	0.10†	**0.05**	1.08†	0.68†	0.25†	0.27†	0.26†	**0.13**	0.04†	0.26†	0.69†	0.67†	0.67†	**0.81**
100J8M	0.54†	0.32†	**0.04**	0.06	0.06	0.05	1.07†	0.65†	**0.10**	0.15	0.14	0.12	0.05†	0.28†	**0.83**	0.77	0.79	**0.83**
100J10M	0.55†	0.35†	**0.02**	0.04	0.05	0.05	1.10†	0.67†	**0.06**	0.11	0.10	0.11	0.04†	0.27†	**0.88**	0.81	0.82	0.81
Mean	0.55	0.37	0.13	0.14	0.15	**0.06**	1.07	0.71	0.32	0.33	0.34	**0.13**	0.05	0.25	0.61	0.59	0.58	**0.82**

Notes: Bold font represents the best value.The symbol '†' denotes a statistically significant difference between the comparison algorithm and the DMOME algorithm, with a significance level of 0.05.

Table 7. The results of Friedman test for all algorithms (confidence level $\alpha = 0.05$).

	GD		IGD		HV	
Algorithm	Rank	*p*-Value	Rank	*p*-Value	Rank	*p*-Value
INSGA-II	6.00	7.38E−17	6.00	2.65E−16	5.83	7.38E−16
MOEAD	5.00		5.00		4.47	
EGA	3.67		3.61		4.17	
EHA	2.39		2.47		2.00	
DQNMA	2.86		2.78		3.53	
The DMOME algorithm	**1.08**		**1.14**		**1.00**	

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.



(a) 20J5M (b) 40J10M (c) 80J5M

Figure 11. Pareto fronts obtained by all algorithms under different scales.

front in the objective space. In fact, the DMOME algorithm also maintains a set of high-quality and diverse local Pareto fronts within different small niches in the feature space. Although the global Pareto front obtained by the DMOME algorithm dominates the local Pareto fronts, the overall quality of the Pareto front is superior to that of the comparison algorithms. This superiority is evident when considering the diversity of behavioural features and the set of solutions available for user selection.

To provide a more comprehensive evaluation, this article further compare the computational time of all algorithms, as shown in Table 8. The results indicate that the DMOME algorithm maintains a good balance between solution quality and efficiency. Compared with the DQNMA—which also uses DQN-based online training—the DMOME algorithm requires less runtime on almost all instances. Although online DQN training makes the DMOME algorithm slower than traditional algorithms such as NSGA-II and MOEAD, the extra cost is acceptable because of the clear improvements in GD, IGD and HV reported earlier. The additional computation mainly arises from network inference and periodic updates, yet the QD mechanism effectively offsets part of this cost by guiding the algorithm toward more promising regions. Overall, the DMOME algorithm runs faster than the DQNMA, remains moderately slower than lightweight baselines, and achieves superior optimization results. This confirms that the additional time cost is reasonable and well-justified.

To investigate whether the proposed algorithm can outperform other methods under the same time budget, additional experiments are conducted under equal computational time. Specifically, a time-based termination criterion was adopted. The time budget was set to 20 seconds for the smallest instance and was increased by 20 seconds for each subsequent problem scale. Table 9 reports the GD, IGD and HV results obtained by different algorithms under these equal time budgets. The results show that the proposed DMOME algorithm consistently attains the lowest average GD and IGD values while achieving the highest HV across all instances. These results demonstrate that the advantage of the DMOME algorithm does not rely on longer execution time. Even under time-critical settings, the proposed method remains robust and consistently outperforms the comparison algorithms in terms of overall solution quality.

Table 8. The computational time of all algorithms.

	Computational time (seconds)					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
20J5M	5.25	4.92	3.92	5.65	17.77	13.42
20J8M	8.56	12.52	8.01	9.38	29.74	28.13
20J10M	10.55	15.06	8.91	10.43	40.04	37.36
30J5M	11.15	22.4	7.66	11.46	38.27	33.21
30J8M	14.55	27.1	10.62	14.32	55.55	46.53
30J10M	15.8	13.19	11.53	15.1	125.06	107.9
40J5M	15.59	85.45	9.14	21.68	104.88	91.76
40J8M	26.04	29.39	13.81	26.06	178.69	159.03
40J10M	45.38	47.13	17.45	31.03	228.71	197.48
50J5M	21.92	47.24	10.17	28.25	125.32	126.86
50J8M	40.9	52.45	17.69	42.1	257.53	229.42
50J10M	53.12	179.75	22.34	52.8	220.86	197.18
80J5M	68.85	55.28	15.93	80.12	267.31	172.88
80J8M	94.37	93.99	26.31	94.21	304.22	281.01
80J10M	122.92	123.97	33.87	123.02	436.51	340.51
100J5M	94.41	94.13	20.7	95.13	303.64	265.04
100J8M	139.79	139.57	31.95	139.63	751.78	587.53
100J10M	198.83	198.45	41.5	199.02	1212.83	864.79
Mean	54.888	68.999	17.306	58.455	261.039	210.002

By modelling ITPS as a multi-objective QD optimization problem, the DMOME algorithm aligns well with real-world applications that require balancing multiple objectives. Choosing the number of AGV transportation events and machine idle times as key features in the feature space helps leverage domain-specific knowledge, enabling the DMOME algorithm to more efficiently find high-quality and diverse solutions. Unlike traditional optimization methods that focus on a single Pareto front, the DMOME algorithm generates diverse solutions across multiple objectives, ensuring both high-quality results and a wide range of solution behaviours. These features are directly related to the objective values, allowing the algorithm to focus on optimizing critical aspects that impact both transportation and production processes. Additionally, the DMOME algorithm integrates domain-specific knowledge, such as the relationship between AGV transportation and machine idling in the ITPS, enabling better exploitation of the feature space. It also leverages the DQN decision models, which allow it to adapt to dynamic environments and optimize performance over time. These combined features make the DMOME algorithm more flexible and efficient than traditional optimization algorithms, particularly in large-scale and complex real-world scenarios.

To assess the differences between the algorithms from multiple perspectives and gain a more comprehensive understanding of the result reliability, this work categorized the test instances into small (denoted by s, where $J = \{20, 30\}$), medium (denoted by m, $J = \{40, 50\}$), and large-scale (denoted by l, $J = \{80, 100\}$) solution sets. Error bar charts with 95% confidence intervals and win rate plots were generated. As shown in Figure 12, each scale was evaluated using GD, IGD and HV metrics, and the win rates were statistically analysed. the DMOME algorithm consistently outperforms the other algorithms across all scales. In both the GD and IGD metrics, the DMOME algorithm exhibits lower mean and median values compared to the other algorithms. In contrast, for the HV metric, the DMOME algorithm shows higher mean and median values than the comparison algorithms. When the DMOME algorithm is excluded, EGA, EHA and the DQNMA achieve the highest win rates across the different metrics. However, with the DMOME algorithm included, the proposed algorithm achieves a win rate of over 50% across all three metrics at each scale. This further highlights the effectiveness of the DMOME algorithm.

Table 9. Numerical results under equal time budgets across different problem scales.

	GD						IGD						HV					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
20J5M	0.45	0.32	0.14	0.17	0.18	**0.07**	1.09	0.84	0.44	0.43	0.53	**0.24**	0.04	0.17	0.49	0.49	0.41	**0.81**
20J8M	0.58	0.42	0.22	0.15	0.15	**0.05**	1.15	0.94	0.38	0.37	0.44	**0.16**	0.01	0.29	0.61	0.54	0.48	**0.72**
20J10M	0.62	0.68	0.17	0.23	0.28	**0.05**	1.25	0.91	0.52	0.54	0.51	**0.23**	0.01	0.21	0.71	0.47	0.39	**0.84**
30J5M	0.49	0.69	0.12	0.27	0.18	**0.09**	1.17	0.93	0.36	0.42	0.39	**0.19**	0.02	0.20	0.66	0.48	0.52	**0.85**
30J8M	1.33	0.62	0.30	0.61	0.40	**0.08**	1.34	1.25	0.59	0.63	0.67	**0.27**	0.00	0.21	0.63	0.48	0.36	**0.77**
30J10M	0.91	0.53	0.24	0.26	0.24	**0.10**	1.31	1.08	0.50	0.59	0.60	**0.23**	0.00	0.25	0.62	0.55	0.43	**0.74**
40J5M	0.56	0.57	0.29	0.24	0.33	**0.07**	1.23	1.11	0.51	0.63	0.61	**0.25**	0.01	0.24	0.72	0.54	0.43	**0.86**
40J8M	0.71	1.01	0.21	0.38	0.42	**0.08**	1.21	1.03	0.60	0.66	0.70	**0.26**	0.01	0.25	0.65	0.65	0.54	**0.81**
40J10M	0.71	1.05	0.29	0.68	0.36	**0.10**	1.24	1.08	0.66	0.71	0.61	**0.18**	0.00	0.24	0.78	0.62	0.50	**0.83**
50J5M	0.56	0.43	0.33	0.28	0.26	**0.11**	1.08	0.94	0.54	0.60	0.56	**0.25**	0.03	0.29	0.83	0.60	0.36	**0.87**
50J8M	0.93	0.56	0.35	0.44	0.36	**0.13**	1.30	1.23	0.70	0.74	0.72	**0.28**	0.00	0.21	0.79	0.72	0.45	**0.85**
50J10M	0.65	0.72	0.29	0.37	0.55	**0.15**	1.29	1.17	0.61	0.80	0.75	**0.34**	0.00	0.22	0.84	0.71	0.61	**0.86**
80J5M	0.61	0.60	0.32	0.32	0.32	**0.12**	1.23	1.06	0.78	0.77	0.84	**0.36**	0.01	0.26	0.82	0.73	0.68	**0.87**
80J8M	0.60	0.72	0.34	0.48	0.36	**0.16**	1.22	1.07	0.67	0.82	0.81	**0.32**	0.01	0.25	0.77	0.78	0.79	**0.83**
80J10M	0.66	0.54	0.39	0.43	0.63	**0.14**	1.29	1.19	0.91	0.92	0.88	**0.38**	0.00	0.22	0.73	0.82	0.84	**0.85**
100J5M	0.86	0.79	0.42	0.40	0.42	**0.37**	1.22	1.10	0.89	0.95	0.98	**0.34**	0.01	0.23	0.85	0.81	0.78	**0.88**
100J8M	0.71	0.76	**0.31**	0.48	0.48	0.36	1.18	1.03	**0.76**	0.92	0.94	0.85	0.02	0.26	0.82	**0.85**	0.83	0.84
100J10M	1.32	0.86	0.47	**0.44**	0.76	0.64	1.35	1.21	**1.00**	1.04	1.01	1.03	0.03	0.25	**0.87**	0.84	0.76	0.85
Mean	0.74	0.66	0.29	0.37	0.37	**0.16**	1.23	1.06	0.63	0.70	0.70	**0.34**	0.01	0.24	0.73	0.65	0.57	**0.83**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

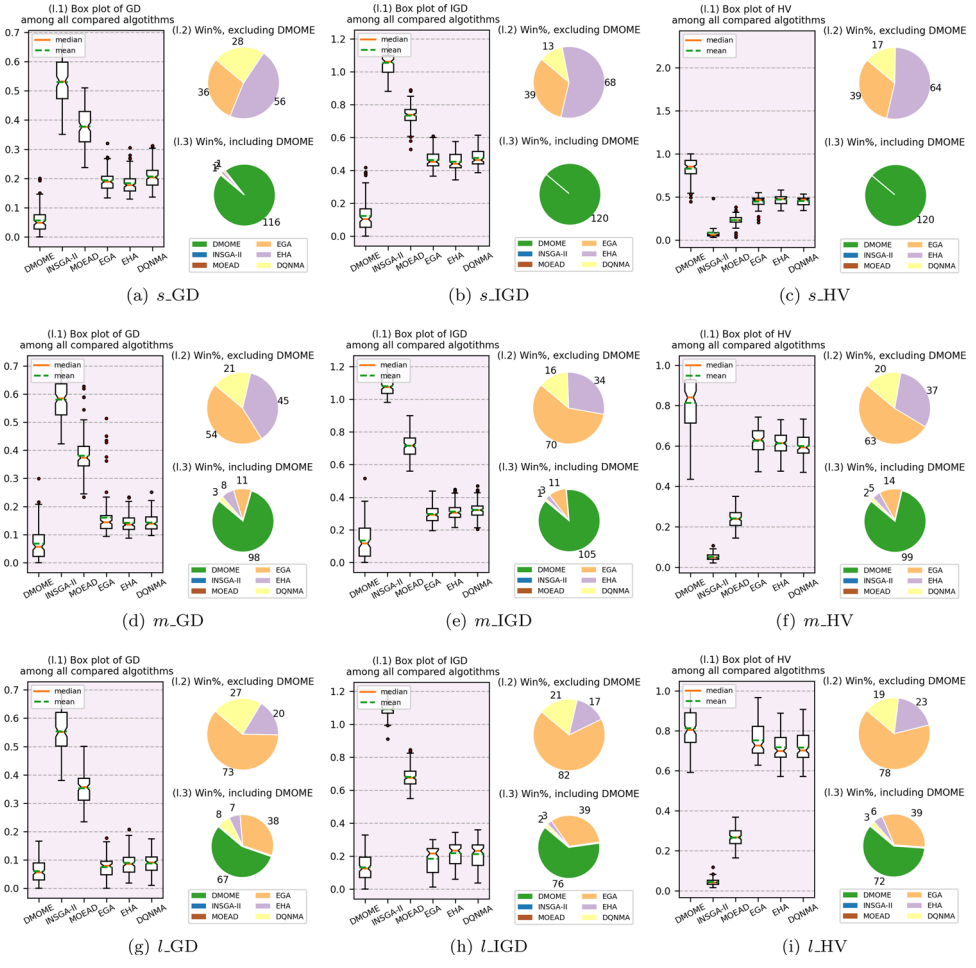


Figure 12. Comparative analysis of the DMOME algorithm with other algorithms under various scales.

5.7. Ablation experiment

To evaluate the effectiveness of different components in the DMOME algorithm, this article compares the proposed algorithm with two variants: (1) V-ES: the variant without knowledge-driven Enhanced Search Strategy; (2) V-DQN: the variant without the DQN selection mechanism. Table 10 lists the GD, IGD and HV results of the proposed algorithm and its variants. For each metric, the last row shows the mean value across all test instances for each algorithm. Additionally, the results from 360 runs are displayed as box plots (Figure 13) to provide a more intuitive view of the solution distribution.

As shown in Table 10, the DMOME algorithm achieved the best results across all test instances, while both V-ES and V-DQN obtained no best values. Based on the mean values of the three evaluation metrics, the DMOME algorithm outperforms V-ES by 62.9%–66.2%, and V-DQN by 47.1%–60.7%. This demonstrates that the use of both the Enhanced Search Strategy (ES) and the DQN selection mechanism significantly improves the performance of QD optimization. Figure 13 further supports this, showing that the DMOME algorithm achieved the best average and median values across 360 runs, highlighting its superior performance. The results indicate that incorporating these strategies helps the DMOME algorithm to explore and exploit the feature space effectively, leading to more optimal outcomes.

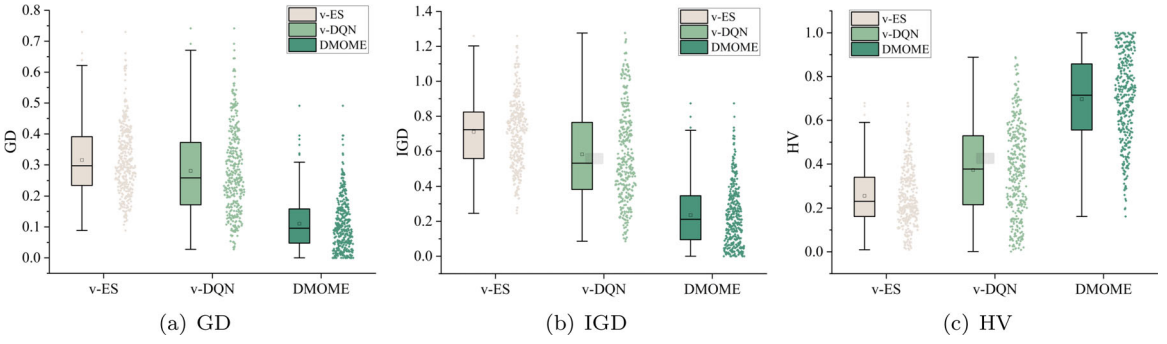


Figure 13. Parameter calibration under GD, IGD and HV metrics.

Table 10. Comparison results between different variants.

	GD			IGD			HV		
Instance	V-ES	V-DQN	DMOME	V-ES	V-DQN	DMOME	V-ES	V-DQN	DMOME
20J5M	0.30	0.29	**0.08**	0.85	0.62	**0.19**	0.15	0.33	**0.73**
20J8M	0.31	0.29	**0.06**	0.66	0.57	**0.13**	0.28	0.38	**0.81**
20J10M	0.36	0.32	**0.16**	0.86	0.60	**0.34**	0.15	0.34	**0.59**
30J5M	0.35	0.21	**0.11**	0.83	0.43	**0.23**	0.17	0.52	**0.73**
30J8M	0.37	0.30	**0.12**	0.80	0.64	**0.24**	0.19	0.34	**0.71**
30J10M	0.46	0.20	**0.07**	0.89	0.45	**0.16**	0.15	0.49	**0.78**
40J5M	0.31	0.29	**0.13**	0.74	0.57	**0.28**	0.23	0.36	**0.64**
40J8M	0.41	0.32	**0.15**	0.87	0.66	**0.30**	0.16	0.33	**0.66**
40J10M	0.32	0.28	**0.12**	0.69	0.53	**0.22**	0.24	0.39	**0.68**
50J5M	0.21	0.23	**0.09**	0.58	0.52	**0.25**	0.32	0.40	**0.66**
50J8M	0.26	0.20	**0.12**	0.56	0.45	**0.25**	0.36	0.50	**0.69**
50J10M	0.30	0.26	**0.12**	0.71	0.58	**0.24**	0.24	0.37	**0.70**
80J5M	0.20	0.21	**0.11**	0.48	0.48	**0.23**	0.43	0.46	**0.68**
80J8M	0.37	0.31	**0.12**	0.78	0.63	**0.26**	0.21	0.33	**0.65**
80J10M	0.38	0.44	**0.13**	0.82	0.83	**0.28**	0.19	0.19	**0.66**
100J5M	0.27	0.28	**0.10**	0.62	0.63	**0.23**	0.31	0.31	**0.70**
100J8M	0.23	0.33	**0.10**	0.51	0.70	**0.23**	0.41	0.28	**0.70**
100J10M	0.26	0.29	**0.10**	0.53	0.57	**0.19**	0.42	0.40	**0.77**
Mean	0.32	0.28	**0.11**	0.71	0.58	**0.24**	0.26	0.37	**0.70**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

The success of combining the knowledge-driven ES and the DQN selection mechanism in multi-objective QD optimization lies in their complementary strengths. ES leverages domain-specific knowledge to guide the search towards promising regions of the feature space, improving efficiency and accelerating convergence. This is particularly effective in complex problems where random search methods may not perform well. On the other hand, the DQN adapts the search strategy based on past experiences, learning which search operators are most effective in different regions. This adaptive mechanism enhances exploration, avoiding stagnation in local optima while ensuring a diverse search. Together, the ES and the DQN enable efficient exploration, dynamic adaptation and a more robust search process, making the algorithm well-suited to tackle complex multi-objective problems.

6. Conclusion and future work

Multi-objective Quality–Diversity (QD) optimization extends Evolutionary Algorithms (EAs) by combining solution diversity with multi-objective optimization, enabling the generation of diverse high-performing solutions across multiple niches in a feature space. This paradigm enhances EAs' ability to address conflicting objectives while maintaining adaptability in complex real-world

scenarios. This article proposed the DMOME algorithm, a novel multi-objective QD approach that integrates knowledge-driven strategies, such as AGV transportation heuristics, to refine the search process and improve Pareto solution quality. Additionally, it employs a DQN-based selection mechanism to adapt operator choices dynamically, learning from historical data and real-time feedback. This combination ensures efficient exploration of the solution space and demonstrates the potential of multi-objective QD optimization for solving complex ITPS problems.

The results on real-world mechanical processing factory instances demonstrate the effectiveness of the proposed approach, with the DMOME algorithm achieving 25.6% to 59.4% improvements compared to the best performing EGA. These findings confirm the advantages of integrating QD modelling with domain knowledge. Nonetheless, this work has certain limitations. At the problem level, the current formulation does not consider distributed factory scenarios or assembly-oriented production environments, which may involve more complex inter-factory transportation or assembly constraints. At the algorithmic level, the initialization relies mainly on random generation, and incorporating knowledge-driven heuristic initialization may further enhance population quality and diversity.

These limitations point to promising directions for future work, where extending the DMOME algorithm to richer industrial settings and developing more informed initialization strategies may yield additional performance gains. Specifically, one potential direction is to expand the feature space by incorporating additional features specific to different scheduling environments, which could enhance the model's adaptability. Another area worth exploring is the scalability of QD optimization algorithms to tackle more complex multi-objective problems. For instance, integrating QD with advanced techniques such as knowledge-driven strategies and feedback mechanisms could further enhance the performance of QD optimization algorithms.

Author contributions

CRediT: **Ronghua Zou:** Conceptualization, Data curation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing; **Haoxiang Qin:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing; **Yi Xiang:** Formal analysis, Funding acquisition, Project administration, Resources, Supervision; **Chunguo Wu:** Supervision, Writing – review & editing

Disclosure statement

No potential conflict of interests was reported by the authors.

Funding

This work is supported by Guangdong Basic and Applied Basic Research Foundation [No. 2024A1515030022]; the Fundamental Research Funds for the Central Universities, Jilin University [No. 93K172024K03, 2024ZYGXZR097]; the National Natural Science Foundation of China [No. 61906069, 62566012]; the Natural Science Research Project of the Education Department of Guizhou Province [QJJ2023061].

Ethics approval and consent to participate

Not applicable. This study does not involve human participants or animals.

Data availability and access

The data used to support the findings of this study is available from the corresponding author, Haoxiang Qin, upon request.

ORCID

Haoxiang Qin http://orcid.org/0000-0002-7373-9055
Yi Xiang http://orcid.org/0000-0003-2118-4825

References

Abdelmaguid, T. F., A. O. Nassef, B. A. Kamal, and M. F. Hassan. 2004. "A Hybrid GA/Heuristic Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles." *International Journal of Production Research* 42 (2): 267–281. https://doi.org/10.1080/0020754032000123579.

Abderrahim, M., A. Bekrar, D. Trentesaux, N. Aissani, and K. Bouamrane. 2022. "Bi-Local Search Based Variable Neighborhood Search for Job-Shop Scheduling Problem with Transport Constraints." *Optimization Letters* 16 (1): 255–280. https://doi.org/10.1007/s11590-020-01674-0.

Babu, A. G., J. Jerald, A. N. Haq, V. M. Luxmi, and T. P. Vigneswaralu. 2010. "Scheduling of Machines and Automated Guided Vehicles in FMS Using Differential Evolution." *International Journal of Production Research* 48 (16): 4683–4699. https://doi.org/10.1080/00207540903049407.

Bäck, T. 1996. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press.

Bhosale, K. C., and P. J. Pawar. 2025. "Investigations into Effect of Waiting Time in Integrated Machine Scheduling and Automated Guided Vehicles Scheduling." *International Journal of Interactive Design and Manufacturing* 19:6707–6724. https://doi.org/10.1007/s12008-025-02248-z.

Bilge, U., and G. Ulusoy. 1995. "A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS." *Operations Research* 43 (6): 1058–1070. https://doi.org/10.1287/opre.43.6.1058.

Bossens, D. M., J.-B. Mouret, and D. Tarapore. 2020. "Learning Behaviour—Performance Maps with Meta-Evolution." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '20)*, 49–57. New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/3377930.3390181.

Bossens, D. M., and D. Tarapore. 2022. "Quality–Diversity Meta-Evolution: Customizing Behavior Spaces to a Meta-Objective." *IEEE Transactions on Evolutionary Computation* 26 (5): 1171–1181. https://doi.org/10.1109/TEVC.2022.3152384.

Cao, Z., C. Lin, and M. Zhou. 2021. "A Knowledge-Based Cuckoo Search Algorithm to Schedule a Flexible Job Shop with Sequencing Flexibility." *IEEE Transactions on Automation Science and Engineering* 18 (1): 56–69. https://doi.org/10.1109/TASE.2019.2945717.

Chaudhry, I. A., A. F. Rafique, I. Elbadawi, M. Aichouni, M. Usman, M. Boujelbene, and A. Boudjemline. 2022. "Integrated Scheduling of Machines and Automated Guided Vehicles (AGVs) in Flexible Job Shop Environment Using Genetic Algorithms." *International Journal of Industrial Engineering Computations* 13 (3): 343–362.

Chen, R., B. Wu, H. Wang, H. Tong, and F. Yan. 2024. "A Q-Learning Based NSGA-II for Dynamic Flexible Job Shop Scheduling with Limited Transportation Resources." *Swarm and Evolutionary Computation* 90:101658. https://doi.org/10.1016/j.swevo.2024.101658.

Cheng, W., L. Meng, B. Zhang, K. Gao, and H. Sang. 2025. "Imitation Learning-Assisted Evolutionary Algorithm for Energy-Efficient Flexible Job Shop Scheduling Problem with Automated Guided Vehicles." *IEEE Transactions on Evolutionary Computation* 30 (1): 171–185. https://doi.org/10.1109/TEVC.2025.3540105.

Coello Coello, C. A., D. A. Van Veldhuizen, and G. B. Lamont. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd ed. Vol. 5. New York: Springer.

Cully, A., J. Clune, D. Tarapore, and J.-B. Mouret. 2015. "Robots That Can Adapt Like Animals." *Nature* 521 (7553): 503–507. https://doi.org/10.1038/nature14422.

Cully, A., and Y. Demiris. 2018. "Quality and Diversity Optimization: A Unifying Modular Framework." *IEEE Transactions on Evolutionary Computation* 22 (2): 245–259. https://doi.org/10.1109/TEVC.2017.2704781.

Dai, M., D. Tang, A. Giret, and M. A. Salido. 2019. "Multi-Objective Optimization for Energy-Efficient Flexible Job Shop Scheduling Problem with Transportation Constraints." *Robotics and Computer-Integrated Manufacturing* 59:143–157. https://doi.org/10.1016/j.rcim.2019.04.006.

Dang, D.-C., A. Neumann, F. Neumann, A. Opris, and D. Sudholt. 2024. "Theoretical Analysis of Quality Diversity Algorithms for a Classical Path Planning Problem." *arXiv Preprint arXiv:2412.11446*. https://arxiv.org/abs/2412.11446.

Du, Y., J. Li, C. Li, and P. Duan. 2024. "A Reinforcement Learning Approach for Flexible Job Shop Scheduling Problem with Crane Transportation and Setup Times." *IEEE Transactions on Neural Networks and Learning Systems* 35 (4): 5695–5709. https://doi.org/10.1109/TNNLS.2022.3208942.

Duarte, M., J. Gomes, S. M. Oliveira, and A. L. Christensen. 2018. "Evolution of Repertoire-Based Control for Robots with Complex Locomotor Systems." *IEEE Transactions on Evolutionary Computation* 22 (2): 314–328. https://doi.org/10.1109/TEVC.2017.2722101.

Ecoffet, A., J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. 2021. "First Return, Then Explore." *Nature* 590 (7847): 580–586. https://doi.org/10.1038/s41586-020-03157-9.

Fontaine, M. C., R. Liu, J. Togelius, A. K. Hoover, and S. Nikolaidis. 2021. "Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network." In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 5922–5930. Palo Alto, CA: AAAI Press.

Fu, Y., Z. Zhang, M. Huang, X. Guo, and L. Qi. 2025. "Multi-Objective Integrated Energy-Efficient Scheduling of Distributed Flexible Job Shop and Vehicle Routing by Knowledge-and-Learning-Based Hyper-Heuristics." *IEEE*

Transactions on Emerging Topics in Computational Intelligence 9 (3): 2137–2150. https://doi.org/10.1109/TETCI.2025.3540422.

Grillotti, L., and A. Cully. 2022. "Unsupervised Behavior Discovery with Quality–Diversity Optimization." *IEEE Transactions on Evolutionary Computation* 26 (6): 1539–1552. https://doi.org/10.1109/TEVC.2022.3159855.

Han, X., W. Cheng, L. Meng, B. Zhang, K. Gao, C. Zhang, and P. Duan. 2024. "A Dual Population Collaborative Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem with AGV." *Swarm and Evolutionary Computation* 86:101538. https://doi.org/10.1016/j.swevo.2024.101538.

He, L., R. Chiong, W. Li, G. S. Budhi, and Y. Zhang. 2022. "A Multiobjective Evolutionary Algorithm for Achieving Energy Efficiency in Production Environments Integrated with Multiple Automated Guided Vehicles." *Knowledge-Based Systems* 243:108315. https://doi.org/10.1016/j.knosys.2022.108315.

Huang, Y., Y. Guo, H. Wei, J. Xin, and S. Yang. 2026. "Collaboration–Competition Estimation of Distribution Algorithm for Flexible Job Shop Co-Scheduling with Multiload AGVs." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 56 (1): 321–335. https://doi.org/10.1109/TSMC.2025.3624888.

Hurink, J., and S. Knust. 2002. "A Tabu Search Algorithm for Scheduling a Single Robot in a Job-Shop Environment." *Discrete Applied Mathematics* 119 (1–2): 181–203. https://doi.org/10.1016/S0166-218X(01)00273-6.

Janmohamed, H., T. Pierrot, and A. Cully. 2023. "Improving the Data Efficiency of Multi-Objective Quality-Diversity through Gradient Assistance and Crowding Exploration." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*, 163–173. New York: Association for Computing Machinery (ACM).

Janmohamed, H., M. Wolinska, S. Surana, T. Pierrot, A. Walsh, and A. Cully. 2024. "Multi-Objective Quality-Diversity for Crystal Structure Prediction." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24)*, 1273–1281. New York: Association for Computing Machinery (ACM).

Jerald, J., P. Asokan, R. Saravanan, and G. Prabaharan. 2006. "Simultaneous Scheduling of Parts and Automated Guided Vehicles in an FMS Environment Using Adaptive Genetic Algorithm." *International Journal of Advanced Manufacturing Technology* 29 (5): 584–589. https://doi.org/10.1007/s00170-005-2529-9.

Khalifa, A., S. Lee, A. Nealen, and J. Togelius. 2018. "Talakat: Bullet Hell Generation through Constrained MAP-Elites." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*, 1047–1054. New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/3205455.3205470.

Li, J.-Q., Y. Du, K.-Z. Gao, P.-Y. Duan, D.-W. Gong, Q.-K. Pan, and P. N. Suganthan. 2022. "A Hybrid Iterated Greedy Algorithm for a Crane Transportation Flexible Job Shop Problem." *IEEE Transactions on Automation Science and Engineering* 19 (3): 2153–2170. https://doi.org/10.1109/TASE.2021.3062979.

Li, R., W. Gong, C. Lu, and L. Wang. 2023. "A Learning-Based Memetic Algorithm for Energy-Efficient Flexible Job-Shop Scheduling with Type-2 Fuzzy Processing Time." *IEEE Transactions on Evolutionary Computation* 27 (3): 610–620. https://doi.org/10.1109/TEVC.2022.3175832.

Li, R., W. Gong, L. Wang, C. Lu, and C. Dong. 2023. "Co-Evolution with Deep Reinforcement Learning for Energy-Aware Distributed Heterogeneous Flexible Job Shop Scheduling." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54 (1): 1–11. https://doi.org/10.1109/TSMC.2023.3305541.

Li, R., L. Wang, W. Gong, and F. Ming. 2024. "An Evolutionary Multitasking Memetic Algorithm for Multi-Objective Distributed Heterogeneous Welding Flow Shop Scheduling." *IEEE Transactions on Evolutionary Computation* 29 (6): 2287–2298. https://doi.org/10.1109/TEVC.2024.3393620.

Li, W., H. Li, Y. Wang, and Y. Han. 2024. "Optimizing Flexible Job Shop Scheduling with Automated Guided Vehicles Using a Multi-Strategy-Driven Genetic Algorithm." *Egyptian Informatics Journal* 25:100437. https://doi.org/10.1016/j.eij.2023.100437.

Li, Y., W. Gu, M. Yuan, and Y. Tang. 2022. "Real-Time Data-Driven Dynamic Scheduling for Flexible Job Shop with Insufficient Transportation Resources Using Hybrid Deep Q Network." *Robotics and Computer-Integrated Manufacturing* 74:102283. https://doi.org/10.1016/j.rcim.2021.102283.

Lim, B., M. Flageat, and A. Cully. 2023. "Understanding the Synergies between Quality–Diversity and Deep Reinforcement Learning." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*, 1212–1220. New York: Association for Computing Machinery (ACM).

Luo, S., L. Zhang, and Y. Fan. 2022. "Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning." *IEEE Transactions on Automation Science and Engineering* 19 (4): 3020–3038. https://doi.org/10.1109/TASE.2021.3104716.

Mahdavi, I., B. Shirazi, and N. Sahebjamnia. 2011. "Development of a Simulation-Based Optimisation for Controlling Operation Allocation and Material Handling Equipment Selection in FMS." *International Journal of Production Research* 49 (23): 6981–7005. https://doi.org/10.1080/00207543.2010.534826.

Meng, L., W. Cheng, C. Zhang, K. Gao, B. Zhang, and Y. Ren. 2025. "Novel CP Models and CP-Assisted Meta-Heuristic Algorithm for Flexible Job Shop Scheduling Benchmark Problem with Multi-AGV." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 55 (11): 8455–8468. https://doi.org/10.1109/TSMC.2025.3604355.

Mou, J., K. Gao, P. Duan, J. Li, A. Garg, and R. Sharma. 2023. "A Machine Learning Approach for Energy-Efficient Intelligent Transportation Scheduling Problem in Real-World Dynamic Circumstances." *IEEE Transactions on Intelligent Transportation Systems* 24 (12): 15527–15539. https://doi.org/10.1109/TITS.2022.3183215.

Mouret, J.-B., and J. Clune. 2015. "Illuminating Search Spaces by Mapping Elites." *arXiv Preprint arXiv:1504.04909*. https://arxiv.org/abs/1504.04909.

Nickelson, A., N. Zerbel, G. Dixit, and K. Tumer. 2023. "Shaping the Behavior Space with Counterfactual Agents in Multi-Objective MAP-Elites." In *Proceedings of the 15th International Joint Conference on Computational Intelligence (IJCCI 2023)*, 41–52. Setúbal, Portugal: SciTePress Digital Library, Science and Technology Publications, LDA. https://doi.org/10.5220/0012164800003595.

Nikfarjam, A., A. Neumann, and F. Neumann. 2024. "On the Use of Quality Diversity Algorithms for the Travelling Thief Problem." *ACM Transactions on Evolutionary Learning and Optimization* 4 (2): 1–22. https://doi.org/10.1145/3641109.

Nilsson, O., and A. Cully. 2021. "Policy Gradient Assisted MAP-Elites." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*, 866–875. New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/3449639.3459304.

Novas, J. M., and G. P. Henning. 2014. "Integrated Scheduling of Resource-Constrained Flexible Manufacturing Systems Using Constraint Programming." *Expert Systems with Applications* 41 (5): 2286–2299. https://doi.org/10.1016/j.eswa.2013.09.026.

Pan, Z., L. Wang, J. Wang, Y. Yu, and R. Li. 2024. "Distributed Energy-Efficient Flexible Manufacturing with Assembly and Transportation: A Knowledge-Based Bi-Hierarchical Optimization Approach." *IEEE Transactions on Automation Science and Engineering* 22:7463–7479. https://doi.org/10.1109/TASE.2024.3396474.

Pan, Z., L. Wang, J. Wang, and Q. Zhang. 2024. "A Bi-Learning Evolutionary Algorithm for Transportation-Constrained and Distributed Energy-Efficient Flexible Scheduling." *IEEE Transactions on Evolutionary Computation* 29 (1): 232–246. https://doi.org/10.1109/TEVC.2024.3354850.

Pan, Z., L. Wang, J. Zheng, J.-F. Chen, and X. Wang. 2022. "A Learning-Based Multi-Population Evolutionary Optimization for Flexible Job Shop Scheduling Problem with Finite Transportation Resources." *IEEE Transactions on Evolutionary Computation* 27 (6): 1590–1603. https://doi.org/10.1109/TEVC.2022.3219238.

Pierrot, T., G. Richard, K. Beguir, and A. Cully. 2022. "Multi-Objective Quality Diversity Optimization." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/3512290.3528823.

Pugh, J. K., L. B. Soros, and K. O. Stanley. 2016. "Quality Diversity: A New Frontier for Evolutionary Computation." *Frontiers in Robotics and AI* 3, Article 40:1–17. https://doi.org/10.3389/frobt.2016.00040.

Qian, C., K. Xue, and R.-J. Wang. 2024. "Quality–Diversity Algorithms Can Provably Be Helpful for Optimization." In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024)*, 6994–7002. Santa Clara, CA: International Joint Conferences on Artificial Intelligence Organisation (IJCAI). https://doi.org/10.24963/ijcai.2024/773.

Qin, H., Y. Han, Q. Chen, L. Wang, Y. Wang, J. Li, and Y. Liu. 2023. "Energy-Efficient Iterative Greedy Algorithm for the Distributed Hybrid Flow Shop Scheduling with Blocking Constraints." *IEEE Transactions on Emerging Topics in Computational Intelligence* 7 (5): 1442–1457. https://doi.org/10.1109/TETCI.2023.3271331.

Qin, H., Y. Xiang, Y. Han, Y. Wang, J. Li, and Q. Pan. 2026. "A Knowledge Region Selection Enhanced Quality–Diversity Algorithm for Real-World Flexible Job Shop Scheduling with Automated Guided Vehicles Transportation." *Engineering Applications of Artificial Intelligence* 164:113352. https://doi.org/10.1016/j.engappai.2025.113352.

Qin, H., Y. Xiang, Y. Han, and X. Yan. 2024. "Optimizing Energy-Efficient Flexible Job Shop Scheduling with Transportation Constraints: A Q-learning Enhanced Quality–Diversity Algorithm." In *Proceedings of the 6th IEEE International Conference on Data-Driven Optimization of Complex Systems (DOCS)*, 373–378. Piscataway, NJ: The Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/DOCS63458.2024.10704469.

Qin, H., Y. Xiang, F. Liu, Y. Han, and Y. Wang. 2025. "Enhancing Quality–Diversity Algorithm by Reinforcement Learning for Flexible Job Shop Scheduling with Transportation Constraints." *Swarm and Evolutionary Computation* 93:101849. https://doi.org/10.1016/j.swevo.2025.101849.

Samuelsen, E., and K. Glette. 2018. "Multi-Objective Analysis of MAP-Elites Performance." *arXiv Preprint arXiv:1803.05174*. https://arxiv.org/abs/1803.05174.

Song, W., X. Chen, Q. Li, and Z. Cao. 2023. "Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning." *IEEE Transactions on Industrial Informatics* 19 (2): 1600–1610. https://doi.org/10.1109/TII.2022.3189725.

Ulusoy, G., and Ü. Bilge. 1992. "Simultaneous Scheduling of Machines and Material Handling System in an FMS." *IFAC Proceedings Volumes* 25 (8): 15–25. https://doi.org/10.1016/S1474-6670(17)54042-2.

Ulusoy, G., F. Sivrikaya-Şerifolu, and Ü. Bilge. 1997. "A Genetic Algorithm Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles." *Computers & Operations Research* 24 (4): 335–351. https://doi.org/10.1016/S0305-0548(96)00061-5.

Urquhart, N., and E. Hart. 2018. "Optimisation and Illumination of a Real-World Workforce Scheduling and Routing Application (WSRP) via Map-Elites." In *Parallel Problem Solving from Nature (PPSN XV)*, edited by A. Auger, C. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Vol. 11101 of the book series *Lecture Notes in Computer Science*, 488–499. Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-99253-2_39.

Vassiliades, V., K. Chatzilygeroudis, and J.-B. Mouret. 2018. "Using Centroidal Voronoi Tessellations to Scale up the Multidimensional Archive of Phenotypic Elites Algorithm." *IEEE Transactions on Evolutionary Computation* 22 (4): 623–630. https://doi.org/10.1109/TEVC.4235.

Wang, J., L. Wang, and H. Han. 2025. "A Knowledge-Driven Cooperative Coevolutionary Algorithm for Integrated Distributed Production and Transportation Scheduling Problem." *IEEE Transactions on Automation Science and Engineering* 22:7435–7448. https://doi.org/10.1109/TASE.2024.3422473.

Wang, R.-J., K. Xue, H. Shang, C. Qian, H. Fu, and Q. Fu. 2023. "Multi-Objective Optimization-Based Selection for Quality–Diversity by Non-Surrounded-Dominated Sorting." In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2023)*. Santa Clara, CA: International Joint Conferences on Artificial Intelligence Organisation (IJCAI). https://api.semanticscholar.org/CorpusID:260857729.

Wang, Y., Y. Han, Y. Wang, Q.-K. Pan, and L. Wang. 2023. "Sustainable Scheduling of Distributed Flow Shop Group: A Collaborative Multi-Objective Evolutionary Algorithm Driven by Indicators." *IEEE Transactions on Evolutionary Computation* 28 (6): 1794–1808. https://doi.org/10.1109/TEVC.2023.3339558.

Wang, Y., K. Xue, and C. Qian. 2022. "Evolutionary Diversity Optimization with Clustering-Based Selection for Reinforcement Learning." In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*, 15226–15242. OpenReview. https://api.semanticscholar.org/CorpusID:251648121.

Xu, G., Q. Bao, and H. Zhang. 2023. "Multi-Objective Green Scheduling of Integrated Flexible Job Shop and Automated Guided Vehicles." *Engineering Applications of Artificial Intelligence* 126:106864. https://doi.org/10.1016/j.engappai.2023.106864.

Yao, Y., L. Gui, X. Li, and L. Gao. 2024. "Tabu Search Based on Novel Neighborhood Structures for Solving Job Shop Scheduling Problem Integrating Finite Transportation Resources." *Robotics and Computer-Integrated Manufacturing* 89:102782. https://doi.org/10.1016/j.rcim.2024.102782.

Yao, Y., Q. Liu, L. Fu, X. Li, Y. Yu, L. Gao, and W. Zhou. 2024. "A Novel Mathematical Model for the Flexible Job-Shop Scheduling Problem with Limited Automated Guided Vehicles." *IEEE Transactions on Automation Science and Engineering* 22:7449–7462. https://doi.org/10.1109/TASE.2024.3356255.

Yao, Y., C. Wang, X. Li, and L. Gao. 2025. "A Knowledge-Driven Hybrid Algorithm for Solving the Integrated Production and Transportation Scheduling Problem in Job Shop." *IEEE Transactions on Intelligent Transportation Systems* 26 (2): 2707–2720. https://doi.org/10.1109/TITS.2024.3511998.

Yu, H., and W. Liang. 2001. "Neural Network and Genetic Algorithm-Based Hybrid Approach to Expanded Job-Shop Scheduling." *Computers & Industrial Engineering* 39 (3): 337–356. https://doi.org/10.1016/S0360-8352(01)00010-9.

Zhang, F., R. Li, and W. Gong. 2024. "Deep Reinforcement Learning-Based Memetic Algorithm for Energy-Aware Flexible Job Shop Scheduling with Multi-AGV." *Computers & Industrial Engineering*189:109917. https://doi.org/10.1016/j.cie.2024.109917.

Zhang, Q., and H. Li. 2007. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." *IEEE Transactions on Evolutionary Computation*11 (6): 712–731. https://doi.org/10.1109/TEVC.2007.892759.

Zhang, Z., F. Wu, B. Qian, R. Hu, L. Wang, and H. Jin. 2023. "A Q-Learning-Based Hyper-Heuristic Evolutionary Algorithm for the Distributed Flexible Job-Shop Scheduling Problem with Crane Transportation." *Expert Systems with Applications* 234:121050. https://doi.org/10.1016/j.eswa.2023.121050.

Zheng, Y., Y. Xiao, and Y. Seo. 2014. "A Tabu Search Algorithm for Simultaneous Machine/AGV Scheduling Problem." *International Journal of Production Research* 52 (19): 5748–5763. https://doi.org/10.1080/00207543.2014.910628.

Zhou, X., F. Wang, N. Shen, and W. Zheng. 2023. "A Green Flexible Job-Shop Scheduling Model for Multiple AGVs Considering Carbon Footprint." *Systems* 11 (8): 1–23. https://doi.org/10.3390/systems11080427.

Zhou, X., F. Wang, B. Wu, Y. Li, and N. Shen. 2025. "Deep Reinforcement Learning-Based Memetic Algorithm for Solving Dynamic Distributed Green Flexible Job Shop Scheduling Problem with Finite Transportation Resources." *Swarm and Evolutionary Computation* 94:101885. https://doi.org/10.1016/j.swevo.2025.101885.

Appendix

A.1. Implementation details of comparison algorithms

To ensure consistency and reproducibility across all comparative experiments, all algorithms were implemented under a unified three-layer encoding scheme, where OS, MA and AS, respectively, denote the operation sequence, machine assignment and AGV selection. All methods use the same decoding procedure described in Algorithm 2 to evaluate makespan and energy consumption. The only difference is that the comparative algorithms output only these two objective values, whereas the DMOME algorithm additionally derives feature-space coordinates required by the QD method. Within this unified representation and decoding framework, each algorithm was faithfully reproduced in accordance with its original design.

INSGA-II followed its canonical implementation, including non-dominated sorting, crowding-distance-based selection, and its original crossover and mutation operators. MOEA/D retained its decomposition strategy and

Table A1. Parameter settings of the comparison algorithms.

Algorithm	Parameter	Value
NSGA-II	Population size, *PS*	100
	Crossover probability, P_{cross}	0.9
	Mutation probability, P_m	0.2
EGA	Population size, *PS*	100
	Crossover probability, P_{cross}	0.8
	Learning factors	(0.3, 0.3)
	Annealing rate coefficient	2
	Iteration number for temperature	10
	Temperature threshold	50
MOEA/D	Population size, *PS*	200
	Number of neighbours	20
	Crossover probability, P_{cross}	1
	Mutation probability, P_m	0.2
	Penalty factor, θ	5
EHA	Population size, *PS*	200
	Crossover probability, P_{cross}	1
	Mutation probability, P_m	0.3
DQNMA	Population size, *PS*	100
	Batch size, *bs*	16
	Learning rate, *lr*	0.001
	Pool size	512
	ϵ	0.9
	γ	0.9

neighbourhood-based update mechanism, but its parent selection, POX crossover, and sequence-adjustment mutation operators were aligned with those used in the DMOME algorithm to ensure compatibility with the ITPS encoding while maintaining fairness. EGA and EHA were implemented strictly according to their original specifications, as both were designed for scheduling problems highly similar to ITPS. For EGA, this included the original learning-factor mechanism, annealing-based acceptance criterion, temperature-schedule control and genetic operators, while for EHA, the crossover, mutation and critical-path-based local search strategy were fully reconstructed as described in the original article. The DQNMA was also implemented following its original framework, including its random initialization procedure, knowledge-driven neighbourhood structures, and the DQN architecture and training process. The only modification concerned hyperparameter values, which were recalibrated on the ITPS dataset to ensure fair comparison. The final parameter settings are provided in the tables in this appendix.

A.2. Partial experimental data supplement

This subsection is complementary to the experimental data in Section 5 . Table A1 lists the parameter settings of all comparison algorithms. Table A2 lists the comparison experimental data for validating the effectiveness of the multi-objective QD optimization model. Table A3 shows the comparison experimental data for validating the effectiveness of the knowledge-driven AGV heuristic strategy. Table A4 presents the comparison experimental data using different selection mechanisms. The results all confirm that the proposed methods significantly enhance the performance of the DMOME algorithm. For detailed analysis, please refer to the corresponding sections of the article.

A.3. Case study

To verify further the effectiveness and generalizability of the DMOME algorithm in related problems, this article applies it to another real-world case: the production workshop of structural parts for China Coal Machinery (The layout of the factory is described in Yao, Liu, *et al.* 2024). The workshop consists of eleven machines, including six different types of processing equipment. Operations are transported to different machines for processing by two AGVs. The specific functions of the six different machine types are shown in the Table A5. Other relevant settings, such as the processing time of the operation and the AGV transportation time, can be found in the original literature.

This article evaluated all algorithms in the real-world instances. As shown in Tables A6, A7 and A8, the DMOME algorithm outperformed all other algorithms in every single one of the 20 tests, achieving the best results across all instances. In terms of GD, IGD and HV, the DMOME algorithm showed superior performance with significantly lower values compared to the other algorithms, including INSGA-II, MOEAD, EGA, EHA and DQNMA. For GD, the DMOME algorithm outperformed the other algorithms by approximately 73.1% to 85.7%. For IGD, the DMOME algorithm outperformed the other algorithms by approximately 74.6% to 84.5%. For HV, the values of the DMOME algorithm were 2.35–8.89 times higher than the other compared algorithms.

These results clearly demonstrate the effectiveness of the DMOME algorithm in handling the multi-objective optimization problem, achieving better convergence and diversity in the solutions compared to the alternative algorithms. This consistent superiority indicates that the proposed methods, including the enhanced search strategy and the DQN-based selection mechanism, significantly enhance the performance of the algorithm in real-world scenarios.

Table A2. Validation of the QD optimization model.

	GD		IGD		HV	
Instance	V-QD	DMOME	V-QD	DMOME	V-QD	DMOME
20J5M	0.42	**0.05**	0.98	**0.13**	0.09	**0.81**
20J8M	0.41	**0.04**	0.96	**0.09**	0.11	**0.88**
20J10M	0.47	**0.09**	0.99	**0.20**	0.09	**0.73**
30J5M	0.48	**0.06**	0.98	**0.13**	0.10	**0.84**
30J8M	0.45	**0.06**	0.98	**0.13**	0.10	**0.83**
30J10M	0.56	**0.05**	1.12	**0.12**	0.04	**0.84**
40J5M	0.41	**0.08**	1.02	**0.17**	0.07	**0.77**
40J8M	0.48	**0.08**	1.05	**0.17**	0.07	**0.79**
40J10M	0.46	**0.07**	1.01	**0.13**	0.08	**0.79**
50J5M	0.40	**0.06**	1.00	**0.16**	0.08	**0.78**
50J8M	0.46	**0.09**	1.03	**0.17**	0.08	**0.79**
50J10M	0.46	**0.07**	1.06	**0.14**	0.06	**0.81**
80J5M	0.42	**0.07**	0.98	**0.15**	0.09	**0.78**
80J8M	0.40	**0.07**	1.00	**0.14**	0.09	**0.80**
80J10M	0.52	**0.08**	1.15	**0.17**	0.04	**0.78**
100J5M	0.44	**0.06**	1.07	**0.15**	0.05	**0.79**
100J8M	0.44	**0.06**	0.99	**0.12**	0.09	**0.83**
100J10M	0.49	**0.07**	1.08	**0.12**	0.06	**0.85**
Mean	0.45	**0.07**	1.02	**0.14**	0.08	**0.81**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

Table A3. Validation of the AGV-based heuristics for the enhanced search strategy.

	GD		IGD		HV	
Instance	V-T	DMOME	V-T	DMOME	V-T	DMOME
20J5M	0.37	**0.08**	0.89	**0.19**	0.12	**0.73**
20J8M	0.33	**0.06**	0.75	**0.13**	0.23	**0.82**
20J10M	0.40	**0.16**	0.92	**0.35**	0.12	**0.57**
30J5M	0.40	**0.11**	0.87	**0.23**	0.15	**0.72**
30J8M	0.41	**0.12**	0.90	**0.23**	0.14	**0.71**
30J10M	0.38	**0.09**	0.83	**0.19**	0.18	**0.75**
40J5M	0.35	**0.12**	0.81	**0.27**	0.18	**0.66**
40J8M	0.39	**0.14**	0.86	**0.30**	0.16	**0.67**
40J10M	0.37	**0.14**	0.78	**0.25**	0.17	**0.64**
50J5M	0.26	**0.11**	0.65	**0.27**	0.28	**0.65**
50J8M	0.39	**0.15**	0.81	**0.27**	0.21	**0.68**
50J10M	0.46	**0.17**	0.88	**0.32**	0.12	**0.61**
80J5M	0.30	**0.12**	0.71	**0.26**	0.24	**0.64**
80J8M	0.37	**0.13**	0.77	**0.27**	0.22	**0.65**
80J10M	0.40	**0.13**	0.87	**0.27**	0.16	**0.67**
100J5M	0.32	**0.09**	0.79	**0.22**	0.19	**0.71**
100J8M	0.36	**0.10**	0.79	**0.23**	0.20	**0.70**
100J10M	0.35	**0.13**	0.78	**0.22**	0.23	**0.73**
Mean	0.37	**0.12**	0.81	**0.25**	0.18	**0.68**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

Table A4. Validation of the DQN selection mechanism.

	GD			IGD			HV		
Instance	V-R	V-QL	DMOME	V-R	V-QL	DMOME	V-R	V-QL	DMOME
20J5M	0.24	0.22	**0.07**	0.50	0.51	**0.16**	0.43	0.43	**0.78**
20J8M	0.28	0.17	**0.06**	0.57	0.36	**0.13**	0.39	0.57	**0.82**
20J10M	0.35	0.29	**0.18**	0.67	0.59	**0.38**	0.29	0.37	**0.54**
30J5M	0.19	0.23	**0.10**	0.37	0.48	**0.20**	0.57	0.47	**0.76**
30J8M	0.28	0.29	**0.11**	0.62	0.57	**0.23**	0.36	0.41	**0.72**
30J10M	0.24	0.27	**0.09**	0.53	0.57	**0.19**	0.42	0.38	**0.74**
40J5M	0.30	0.37	**0.13**	0.58	0.83	**0.29**	0.35	0.19	**0.64**
40J8M	0.31	0.25	**0.14**	0.65	0.56	**0.30**	0.33	0.42	**0.67**
40J10M	0.27	0.21	**0.11**	0.51	0.43	**0.21**	0.41	0.49	**0.70**
50J5M	0.24	0.20	**0.10**	0.52	0.50	**0.25**	0.40	0.43	**0.67**
50J8M	0.20	0.21	**0.12**	0.44	0.50	**0.24**	0.50	0.46	**0.68**
50J10M	0.25	0.27	**0.12**	0.56	0.56	**0.23**	0.38	0.39	**0.71**
80J5M	0.23	0.22	**0.11**	0.52	0.50	**0.24**	0.43	0.42	**0.66**
80J8M	0.29	0.25	**0.11**	0.58	0.56	**0.24**	0.38	0.39	**0.69**
80J10M	0.42	0.32	**0.13**	0.80	0.62	**0.27**	0.20	0.34	**0.67**
100J5M	0.28	0.16	**0.10**	0.63	0.44	**0.23**	0.29	0.49	**0.68**
100J8M	0.30	0.28	**0.09**	0.64	0.62	**0.21**	0.32	0.34	**0.73**
100J10M	0.29	0.24	**0.10**	0.57	0.44	**0.19**	0.40	0.52	**0.77**
Mean	0.28	0.25	**0.11**	0.57	0.53	**0.23**	0.38	0.42	**0.70**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

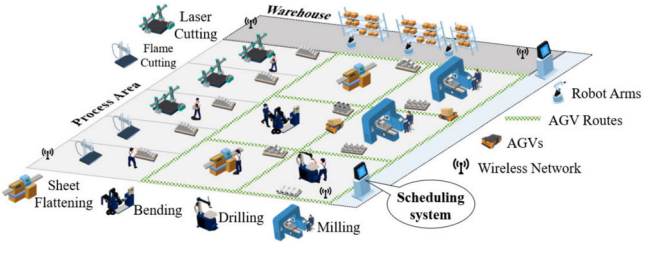


Figure 14. The layout of the real-world case (Yao, Liu, *et al.* 2024).

Table A5. The numbers and functions corresponding to the different types of machine.

Type	Function	Available machine
1	Laser cutting	M1, M2, M3
2	Flame cutting	M4, M5
3	Sheet flattening	M6, M7
4	Bending	M8
5	Milling	M9, M10
6	Drilling	M11

Table A6. GD results for all algorithms in the real-world case.

	GD					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
Test1	0.52	0.37	0.34	0.34	**0.21**	**0.21**
Test2	0.49	0.30	0.27	0.27	0.21	**0.09**
Test3	0.38	0.42	0.25	0.25	0.28	**0.13**
Test4	0.47	0.38	0.33	0.33	0.28	**0.07**
Test5	0.50	0.45	0.25	0.25	0.27	**0.09**
Test6	0.57	0.40	0.29	0.29	0.23	**0.04**
Test7	0.53	0.29	0.26	0.26	0.23	**0.09**
Test8	0.49	0.37	0.22	0.24	0.28	**0.12**
Test9	0.41	0.29	0.30	0.26	0.29	**0.11**
Test10	0.56	0.31	0.34	0.34	0.26	**0.06**
Test11	0.50	0.31	0.20	0.20	0.22	**0.03**
Test12	0.51	0.45	0.25	0.25	0.24	**0.10**
Test13	0.55	0.37	0.22	0.22	0.28	**0.06**
Test14	0.45	0.36	0.25	0.25	0.27	**0.04**
Test15	0.37	0.46	0.34	0.34	0.30	**0.00**
Test16	0.56	0.39	0.21	0.21	0.27	**0.05**
Test17	0.50	0.36	0.27	0.27	0.30	**0.02**
Test18	0.41	0.33	0.28	0.28	0.22	**0.03**
Test19	0.60	0.46	0.30	0.30	0.32	**0.01**
Test20	0.50	0.40	0.31	0.31	0.32	**0.11**
Mean	0.49	0.37	0.27	0.27	0.26	**0.07**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

Table A7. IGD results for all algorithms in the real-world case.

	IGD					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
Test1	0.88	0.81	0.58	0.58	0.59	**0.41**
Test2	0.92	0.85	0.59	0.59	0.60	**0.20**
Test3	0.95	0.74	0.58	0.58	0.63	**0.26**
Test4	0.98	0.85	0.57	0.57	0.57	**0.12**
Test5	0.99	0.78	0.60	0.60	0.59	**0.21**
Test6	0.99	0.79	0.58	0.58	0.60	**0.12**
Test7	0.98	0.76	0.58	0.58	0.59	**0.16**
Test8	0.93	0.74	0.59	0.59	0.59	**0.23**
Test9	0.98	0.76	0.59	0.59	0.57	**0.23**
Test10	0.93	0.79	0.57	0.57	0.61	**0.16**
Test11	0.98	0.74	0.58	0.58	0.59	**0.05**
Test12	0.99	0.78	0.63	0.63	0.59	**0.19**
Test13	0.92	0.72	0.58	0.58	0.58	**0.10**
Test14	0.94	0.80	0.62	0.62	0.60	**0.10**
Test15	1.04	0.79	0.59	0.59	0.59	**0.00**
Test16	0.97	0.77	0.58	0.58	0.58	**0.10**
Test17	1.00	0.78	0.58	0.58	0.59	**0.04**
Test18	0.96	0.70	0.60	0.60	0.58	**0.06**
Test19	1.04	0.78	0.61	0.61	0.62	**0.04**
Test20	0.96	0.80	0.62	0.62	0.63	**0.20**
Mean	0.97	0.78	0.59	0.59	0.59	**0.15**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.

Table A8. HV results for all algorithms in the real-world case.

	HV					
Instance	INSGA-II	MOEAD	EGA	EHA	DQNMA	DMOME
Test1	0.13	0.19	0.35	0.35	0.35	**0.49**
Test2	0.10	0.16	0.34	0.34	0.34	**0.73**
Test3	0.11	0.21	0.34	0.34	0.32	**0.66**
Test4	0.08	0.16	0.35	0.35	0.35	**0.83**
Test5	0.09	0.19	0.35	0.35	0.35	**0.73**
Test6	0.08	0.18	0.35	0.35	0.34	**0.83**
Test7	0.08	0.22	0.35	0.35	0.35	**0.77**
Test8	0.11	0.22	0.34	0.34	0.35	**0.68**
Test9	0.08	0.22	0.35	0.35	0.36	**0.68**
Test10	0.11	0.20	0.35	0.35	0.32	**0.77**
Test11	0.08	0.22	0.36	0.36	0.36	**0.93**
Test12	0.08	0.20	0.31	0.31	0.34	**0.73**
Test13	0.10	0.24	0.36	0.36	0.36	**0.85**
Test14	0.10	0.19	0.33	0.33	0.33	**0.87**
Test15	0.07	0.18	0.35	0.35	0.34	**1.00**
Test16	0.09	0.20	0.36	0.36	0.35	**0.84**
Test17	0.10	0.19	0.34	0.34	0.34	**0.96**
Test18	0.09	0.25	0.35	0.35	0.36	**0.91**
Test19	0.06	0.19	0.33	0.33	0.32	**0.97**
Test20	0.09	0.19	0.34	0.34	0.33	**0.71**
Mean	0.09	0.20	0.34	0.34	0.34	**0.80**

Note: Boldface values are used to highlight the best-performing results, facilitating quick comparison across algorithms.