Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.055244

ARTICLE

Q-Learning-Assisted Meta-Heuristics for Scheduling Distributed Hybrid Flow Shop Problems

Qianyao Zhu[1], Kaizhou Gao[1,*], Wuze Huang[1], Zhenfang Ma[1] and Adam Slowik[2]

[1]Institute of Systems Engineering, Macau University of Science and Technology, Macau, 99078, China

[2]Department of Electronics and Computer Science, Koszalin University of Technology, Koszalin, 75-453, Poland

*Corresponding Author: Kaizhou Gao. Email: kzgao@must.edu.mo

Received: 21 June 2024 Accepted: 13 August 2024 Published: 12 September 2024

ABSTRACT

The flow shop scheduling problem is important for the manufacturing industry. Effective flow shop scheduling can bring great benefits to the industry. However, there are few types of research on Distributed Hybrid Flow Shop Problems (DHFSP) by learning assisted meta-heuristics. This work addresses a DHFSP with minimizing the maximum completion time (Makespan). First, a mathematical model is developed for the concerned DHFSP. Second, four Q-learning-assisted meta-heuristics, e.g., genetic algorithm (GA), artificial bee colony algorithm (ABC), particle swarm optimization (PSO), and differential evolution (DE), are proposed. According to the nature of DHFSP, six local search operations are designed for finding high-quality solutions in local space. Instead of random selection, Q-learning assists meta-heuristics in choosing the appropriate local search operations during iterations. Finally, based on 60 cases, comprehensive numerical experiments are conducted to assess the effectiveness of the proposed algorithms. The experimental results and discussions prove that using Q-learning to select appropriate local search operations is more effective than the random strategy. To verify the competitiveness of the Q-learning assistedmeta-heuristics, they are compared with the improved iterated greedy algorithm (IIG), which is also for solving DHFSP. The Friedman test is executed on the results by five algorithms. It is concluded that the performance of four Q-learning-assisted meta-heuristics are better than IIG, and the Q-learning-assisted PSO shows the best competitiveness.

KEYWORDS

Distributed scheduling; hybrid flow shop; meta-heuristics; local search; Q-learning

1 Introduction

Distributed production and manufacturing affect the efficiency and competitiveness of enterprises and are important components of intelligent manufacturing systems [1]. Distributed flow shop scheduling is an important problem in distributed production and manufacturing. The study of distributed flow shop scheduling problems holds practical application value and significance [2]. Distributed flow shop scheduling refers to industries that have multiple workshops in different geographical locations. These workshops need to be managed to process and manufacture products. In the distributed flow shop scheduling problems, the workpieces are assigned to different workshops,

the resources are allocated for workpieces, and the workpieces are sequenced in each workshop, to optimize one or more production targets. Reasonable and efficient distributed scheduling can reduce cost, improve industrial competitiveness, and fully utilize resources [3].

The hybrid flow shop scheduling problem (HFSP) refers to a production facility that consists of multiple stages, each of which includes one or more parallel machines. HFSP is extensively present in many manufacturing industries, such as steel, textile, petrochemicals, and electronics and each factory represents a HFSP environment [4,5]. HFSP can be considered a combination of flow shop scheduling problem (FSP) and parallel machine scheduling [6]. To enhance the efficiency of the flow shop, in the traditional HFSP, jobs are processed by a single factory that utilizes one or more parallel machines for production at each stage. Therefore, numerous scholars have conducted extensive research on HFSP and have proposed various methods to address the issue, including the exact methods [7], heuristics [8], and meta-heuristics [9]. HFSP is a typical flow shop scheduling problem. It combines the characteristics of both classic flow shop and parallel machine scheduling and is an non-deterministic polynomial (NP-hard) problem [10].

The manufacturing problem with multiple hybrid flow plants is called DHFSP. DHFSP combines distributed production and HFSP, which is more complex and presents greater optimization challenges compared to HFSP. Compared to distributed production and HFSP, DHFSP has less studied. However, DHFSP is a common issue in manufacturing, particularly in semiconductor manufacturing. This issue holds significant research importance. Fig. 1 shows a semiconductor manufacturing process with two distinct workshops. All semiconductors begin with wafers. The raw materials used to make wafers are processed to produce finished wafers, followed by the application of oxidation protection. After completing the aforementioned steps, photo etching, etching, and thin film deposition are carried out on the wafer to create the circuit and micro-devices. Finally, the interconnection process is carried out to connect the circuits on the wafer, and the preliminary manufacturing of the semiconductor is completed.

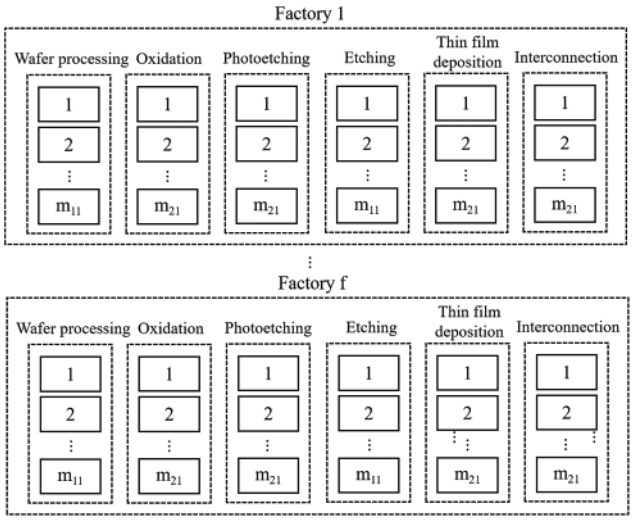

Figure 1: The diagram of DHFSP in semiconductor manufacturing

In the recent development, DHFSP has garnered significant thinking and has yielded numerous results [11–15]. Meta-heuristics are widely utilized to solve DHFSP. The artificial bee colony algorithm (ABC) is widely used to resolve the discrete harmony search and firefly algorithm (DHFSP).

Li et al. [16] improved the ABC (IABC) minimization algorithm to solve DHFSP, the IABC uses a two-stage encoding approach and a machine-selected decoding approach. The hybrid search strategy combines the benefits of simulated annealing (SA) and retention mechanisms to enhance the performance of ABC. In [17], a hybrid ABC method with mixed domain operation and a multiple critical plant exchange strategy is proposed. The self-adaptive ABC (SABC) is introduced to address the DRCHFS, the algorithm considers resource constraints and machine allocation in decoding and introduces a new initialization strategy that considers the maximum completion time of the work piece [18]. There are also numerous studies utilizing other meta-heuristic algorithms. Hao et al. found an improved crossover operator based on Partial Mapping Crossover (PMX) to enhance the performance of the brain storm optimization Algorithm (BSO) [19]. A hybrid multi-objective iterated greedy with a new integration initialization strategy by incorporating four heuristic rules is introduced by Lu et al. [20] to solve an energy-aware problem of DHFSP. Li et al. [21] classified groups to implement different evolutionary strategies becomes a multi-group cooperative evolutionary mechanism, increasing the diversity of solutions. In [22], Cai et al. introduced a novel shuffled frog-leaping algorithm (FLA) with memeplex quality designed, selecting new memeplex by evaluating the quality of each memeplex. Wu utilizes particle swarm optimization (PSO) in conjunction with the leapfrog algorithm to address production management issues in manufacturing facilities [23]. It combines the variation and crossover ideas of genetic algorithms.

With the wide application of reinforcement learning (RL), there is an increasing amount of research to solve DHFSP using reinforcement learning. Using RL and an effective solution selection strategy based on decomposition can help in selecting appropriate improved operators [24], which is advantageous for both the convergence and diversity of solutions. Q-learning, as a kind of RL, is often used to solve the distributed flow shop scheduling problem. Zhang et al. [25] proposed a meta-reinforcement learning-based meta-heuristic (MRLM). The search operator is trained to construct the initial learning model, and then Q-learning is utilized to learn and assimilate feedback for selecting the search operator. In order to expedite the convergence of the algorithm, it is common to design multiple domain structures. Currently, many studies combine Q-learning to select the domain structure more effectively [26,27]. In literature [28], Luo et al. used Q-learning to select the most effective strategy among the six domain structures designed to accelerate convergence. A training algorithm based on genetic algorithms is proposed by Liu et al. [29]. Combined with a genetic algorithm, a target evaluation strategy for each workshop state is proposed, and the Deep Q-Network (DQN) is enhanced to ensure stability during training. Zhao et al. [30] proposed a knowledge-driven cooperative scatter search algorithm. To enhance the exploration ability and search efficiency of the algorithm, Q-learning is design to select disturbance strategies. The aforementioned research demonstrates that combining Q-learning to enhance algorithm performance is feasible and effective. There is few existing research on applying Q-learning to solve DHFSP. It is a challenge that design Q-learning to effectively enhance the performance of algorithms to solve the DHFSP. This paper introduces meta-heuristics integrates Q-learning to solve DHFSP.

The main contributions of this study are shown as follows:

(1) A mathematical model is developed for solving the DHFSP.

(2) Six local search schemes are designed based on the nature of DHFSP to improve the performance of four meta-heuristics.

(3) A learning strategy is proposed to assist the algorithms find the best local search strategy during iterations.

The rest of this study consists of the following. In Section 2, we introduce the DHFSP and establish a mathematical model. In Section 3, the proposed algorithms and the improvement strategies are presented. In Section 4, experiments are conducted to validate the effectiveness of the proposed strategies. Finally, Section 5 summarizes this work, and future directions are given.

2 Problem Description and Model

In this section, the specific process of DHFSP is introduced. There is a set of jobs that need to be processed in factories. Each factory has stages. Each job must include all processing steps. Each stage has one or more machines. The denote the number of machines at stage in factory. When a machine is available on the production stage, the next job can be processed on this machine. The optimization objective is to minimize the maximum completion time among all factories (makespan). The makespan is crucial for optimizing resource utilization and achieving high productivity.

Parameters:

F:	Number of factories.
s:	Number of stages in each factory.
n:	Number of jobs.
f:	Index for factories. $f = \{1, 2, \ldots, F\}$.
g:	Index for stages. $g = \{1, 2, \ldots, s\}$.
j:	Index for jobs. $j = \{1, 2, \ldots, n\}$.
i:	Index for machines.
m_{gf}:	The number of machines at stage g in factory f.
B_{jg}:	The beginning time of job j in stage g.
P_{jg}:	The processing time of job j in stage g.
C_{jg}:	The completion time of job j in stage g.
C_{max}:	The makespan of a schedule.
x_{jf}:	1 if job j is allocated to factory f and 0 otherwise.
y_{jigf}:	1 if job j is processed on machine i at stage g in factory f and 0 otherwise.
$z_{jj'gf}$:	1 if job j is precedes j' at stage g in factory f and 0 otherwise.

The target of the problem is to minimize the makespan as follows:

$$\min \mathrm{C}_{\max} = \max_{\mathrm{j}=1,\ldots,\mathrm{n};\mathrm{g}=1,\ldots,\mathrm{s}} \left\{\mathrm{C}_{\mathrm{jg}}\right\} = \max_{\mathrm{j}=1,\ldots,\mathrm{n};\mathrm{g}=1,\ldots,\mathrm{s}} \{\mathrm{B}_{\mathrm{jg}} + \mathrm{P}_{\mathrm{jg}}\} \tag{1}$$

On the basis of the notations described, the mathematical model of DHFSP is as follows:

$$\sum_{\mathrm{f}=1}^{\mathrm{F}} \mathrm{x}_{\mathrm{jf}} = 1 \ \forall \mathrm{j} = 1, 2, \ldots, \mathrm{n} \tag{2}$$

$$\mathrm{C}_{\mathrm{jg}+1} - \mathrm{P}_{\mathrm{jg}+1} \geq \mathrm{C}_{\mathrm{jg}} \forall \mathrm{j} = 1, 2, \ldots, \mathrm{n}\ \mathrm{g} = 1, 2, \ldots, \mathrm{s} - 1 \tag{3}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{m_{gf}}} \mathrm{y}_{\mathrm{jigf}} = \mathrm{x}_{\mathrm{jf}} \forall \mathrm{f} = 1, 2, \ldots, \mathrm{F}\ \mathrm{g} = 1, 2, \ldots, \mathrm{s}\ \mathrm{j} = 1, 2, \ldots, \mathrm{n} \tag{4}$$

$$\mathrm{z}_{\mathrm{jj'gf}} + \mathrm{z}_{\mathrm{j'jgf}} \leq 1 \forall \mathrm{j}, \mathrm{j}' = 1, 2, \ldots, \mathrm{n}\ \mathrm{g} = 1, 2, \ldots, \mathrm{s}\ \mathrm{f} =, 1, 2, \ldots, \mathrm{F} \tag{5}$$

$$\sum_{j=1}^{n} y_{jigf} \leq 1 \; \forall f = 1, 2, \ldots, F \; g = 1, 2, \ldots, s \tag{6}$$

$$B_{jg} \geq 0 \; \forall j = 1, 2, \ldots, n \; g = 1, 2, \ldots, s \tag{7}$$

$$\sum_{j=1}^{n} y_{jigf} \geq \sum_{j=1}^{n} y_{ji+1gf} \; \forall f = 1, 2, \ldots, F \; g = 1, 2, \ldots, s \tag{8}$$

Constraint (2) states that each job assigned to one factory cannot be assigned to another. Constraint (3) indicates that the current operation can only be performed after its previous operation is complete. Constraint (4) indicates that all jobs must go through all operations, and when a machine starts an operation, the operation cannot be assigned to other machines. Constraints (5) and (6) describe that a task can only be processed on one machine, and when a machine starts an operation, it cannot perform other operations. Constraint (7) stipulates that the start time of each operation will not be less than 0. Constraint (8) means that when multiple parallel machines are idle, the previous machine is given priority.

3 The Proposed Algorithms

In this section, we present the encoding and decoding methods, four meta-heuristics, local search operations, and Q-learning. Then, we propose a method for selecting local search operations using Q-learning during iterations and introduce the framework of the proposed algorithms.

3.1 Encoding and Decoding

DHFSP can be divided into the following steps: (1) assign jobs to several factories, (2) assign jobs to multiple machines within each factory, and (3) sort jobs on machines. There are widely used encoding methods for HFSP in [31]. In this study, the solution of DHFSP is encoded and decoded by the method in [32]. Set a solution $\pi = [\pi_1, \pi_2, \ldots, \pi_f]$, where $\pi_i, i = 1, 2, \ldots, f$ is the sequence of tasks in factory i. During the decoding process for factory i, the processing order of jobs in the first stage is the same as the order in π_i. In the later stages, the jobs are processed in sequence according to the completion order in the previous stage. When jobs enter the next stage at the same time, they are processed according to their processing order in π_i. For example, 4 jobs are processed in 2 factories. Factory 1 has two stages, Stage 1 with 1 machine, and Stage 2 with 2 machines. In Factory 2, Stage 1 has two machines, and another has 1 machine. The processing time for each task is shown in Table 1. Suppose $\pi = [[3, 2], [1, 4]]$ as a feasible solution. Assign Job 2 and Job 3 to Factory 1, while Job 1 and Job 4 to Factory 2 according to π. In the first stage of Factory 1, Job 3 is processed first, followed by Job 2. In the later stage, priority is given to Job 3 with the least completion time from the previous stage. The same principle applies to Factory 2. Therefore, the completion times of two factories are $C_1 = 6+4+6 = 16$, $C_2 = 3+5+4 = 12$, $C_{max} = 16$. The Gantt chart for decoding is shown in Fig. 2.

Table 1: Processing times

Factory 1	Job 2	Job 3	Factory 2	Job 1	Job 4
Stage 1	4	6	Stage 1	5	3
Stage 2	6	5	Stage 2	4	5

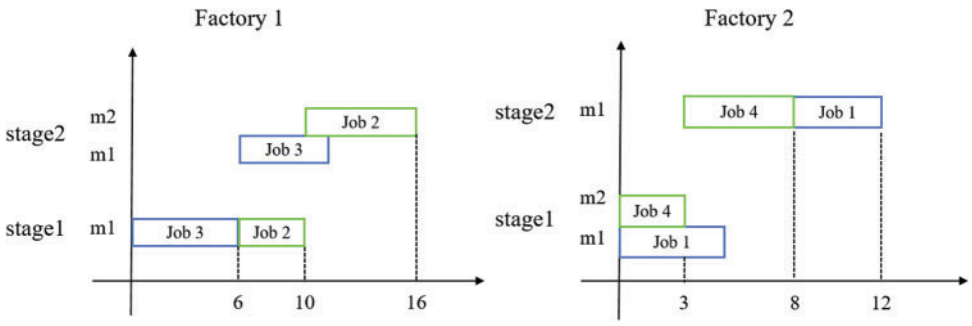

Figure 2: The Gantt chart of one solution for the example is in Table 1

3.2 Meta-Heuristics

Many meta-heuristics are utilized to solve production scheduling problems. In this study, we select four algorithms: ABC, differential evolution (DE), PSO, and genetic algorithm (GA). These four algorithms are most commonly used to solve DHFSP problems, and most studies have confirmed that they have better performance in solving DHFSP. Fig. 3 illustrates the four algorithms' framework. First, the initial population is randomly generated, and the mass of the initial population is calculated, which represents the current optimal solution by default. Then enter the iteration, and update the optimal solution by comparing the fitness values according to meta-heuristics respective population updating strategies. The fitness value is used to assess the quality of the current solutions generated by the four meta-heuristics. In the DHFSP, the fitness value is makespan.

3.2.1 GA

The GA is a computational model that simulates the biological evolution process by mimicking natural selection and the genetic mechanisms of Darwinian evolution. GA is now commonly used in optimizing problems across various engineering fields. In GA, each solution is considered a chromosome. In each generation, new chromosomes are created through three operations crossover, mutation, and selection, using the parent chromosome from the previous generation.

3.2.2 PSO

The PSO simulates the process of hunting birds and fish in nature. Its principle of seeking the global optimal solution to a problem through group collaboration is now widely applied in optimization problems across various engineering fields.

3.2.3 DE

The DE is based on population evolution. The fundamental operations of DE include mutation, crossover, and selection. The DE generates new individuals through the mutation operation based on the difference. Compared with the GA, DE retains the global search strategy based on population, utilizes real number coding, employs simple variation operations based on differences, and implements a one-to-one competitive survival strategy to simplify genetic operations.

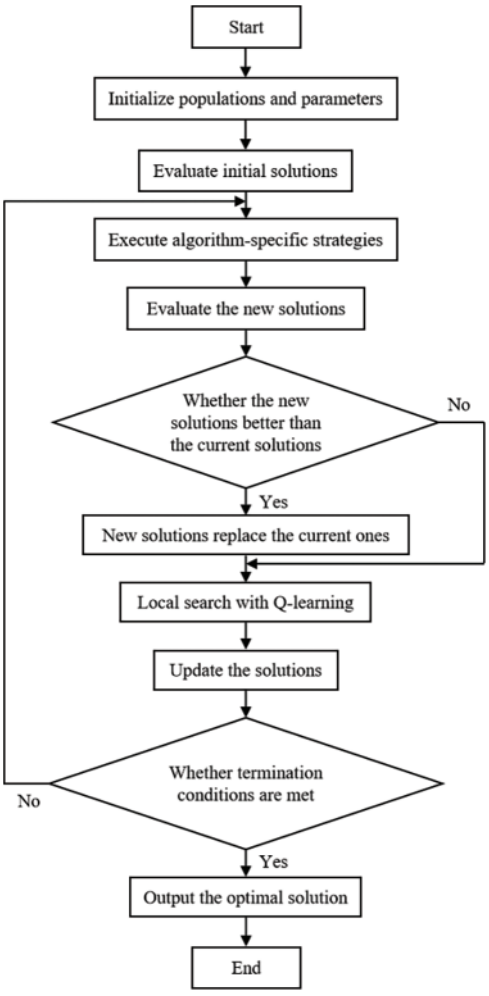

Figure 3: Algorithm flow framework

3.2.4 ABC

The ABC simulates the behavior of bees searching for honey sources in nature and seeks the optimal solution through the division of labor and information sharing. A nectar source stands for a feasible solution, and the quantity of nectar in each source reflects the fitness of that solution. ABC exhibits strong global search ability and local optimization capability, making it well-suited for function optimization problems. Compared with other heuristic algorithms, it has fewer control parameters and higher robustness.

3.3 Local Search

The meta-heuristics are easy to implement and converge quickly. However, the algorithms may easily fall into a situation where they reach a local optimum during the iteration process. To enable the algorithm to escape from a local optimal solution, this paper develops six local search methods based on the nature of the concerned problems. In DHFSP, set the factory with the maximum C_{max} as

critical factory f^*, and randomly select a non-critical factory as f_{other}. Figs. 4–9 show the six types of local search.

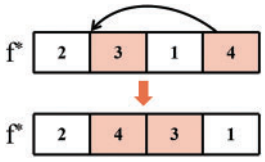

Figure 4: Critical factory insertion

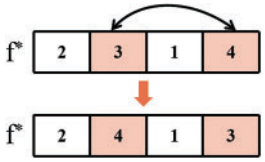

Figure 5: Critical factory swap

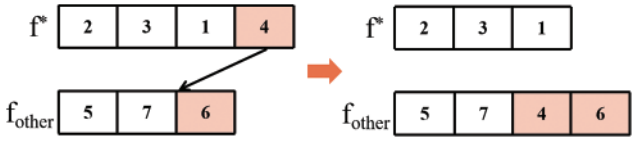

Figure 6: Critical factory and other factory insertion

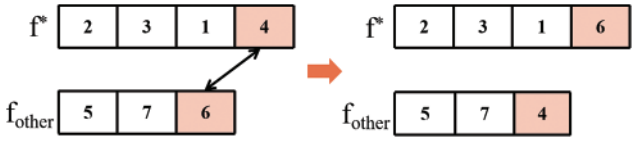

Figure 7: Critical factory and other factory swap

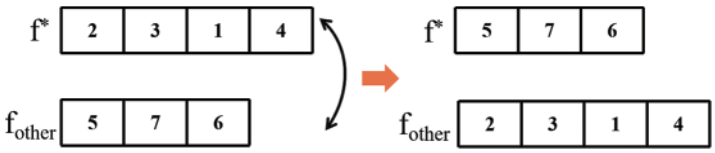

Figure 8: Critical factory and other factory sequence exchange

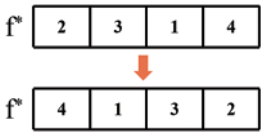

Figure 9: Critical factory opposite sequence

Critical Factory Insertion (Fig. 4): Select a job in the f^* and insert it into random location in the f^*.

Critical Factory Swap (Fig. 5): Select two jobs randomly in the f^* and swap their locations.

Critical Factory and Other Factory Insertion (Fig. 6): Select a job at random in both the f^* and f_{other}, then insert the task in the f^* in front of the job in f other.

Critical Factory and Other Factory Swap (Fig. 7): Select a job at random in both the f^* and f_{other}, then swap their locations.

Critical Factory and Other Factory Sequence Exchange (Fig. 8): Exchange job sequences for f^* and f_{other}.

Critical Factory Opposite Sequence (Fig. 9): Reverse the sequence of jobs in f^*.

The f^* is the factory with the maximum C_{max}. It indicates that adjusting the jobs in the f^* can effectively reduce the maximum C_{max}, and optimize the optimal solution.

3.4 Q-Learning

RL is a field of machine learning. In RL, agents acquire tactics to optimize rewards or accomplish particular objectives through their engagements with the environment. RL focuses on how agents make decisions in an environment to maximize cumulative rewards. Learners will take actions in the environment and receive rewards based on their actions. Through feedback, the agent will eventually obtain the optimal policy. The policy aims to maximize his reward for actions and interactions with the environment. The framework of RL is depicted in Fig. 10.

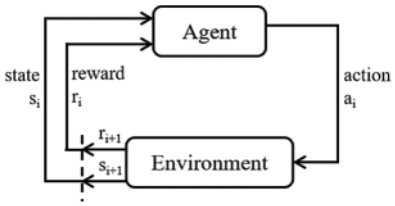

Figure 10: The framework of reinforcement learning

Q-learning is a form of RL. In Q-learning, positive behavior is rewarded, while negative behavior is punished. Q-learning introduces new components within the framework of reinforcement learning. Q-values represent the value of acting in its current state. Q-learning uses Q-values to determine the optimal action. The updated formula for Q-values is as follows:

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * \max(Q(s',a')) - Q(s,a)) \quad (9)$$

where $Q(s,a)$ is the Q-values of taking the action a in the state s. The α is the learning rate while the γ is the discount factor. The r is the actual reward received for the action a. The $max(Q(s',a'))$ is the highest expected reward for all possible actions in state s'.

Q-table is used to store Q-values. The Q-table contains a list of rewards for the optimal behavior in each state within a specific environment. It can aid in understanding the outcomes of various behaviors in different scenarios. Once an agent makes in deciding the environment, the corresponding Q-value in the Q-table will be updated. Through continuous iteration and receiving more feedback, the Q-table will become more accurate, allowing the agent to make better decisions toward achieving the optimal solution. The Q-table for local search selection during iterations is shown in Table 2, where both rows

and columns are set to local search operators. At the beginning, the Q-value of each local search in the Q-table is the same. With each iteration, the Q-value of each local search is updated. Compare the Q-values and choose the local search that receives the best feedback in the current state.

Table 2: Q-table

	Local search1	Local search2	Local search3	Local search4	Local search5	Local search6
S1	Q(1,1)	Q(1,2)	Q(1,3)	Q(1,4)	Q(1,5)	Q(1,6)
S2	Q(2,1)	Q(2,2)	Q(2,3)	Q(2,4)	Q(2,5)	Q(2,6)
S3	Q(3,1)	Q(3,2)	Q(3,3)	Q(3,4)	Q(3,5)	Q(3,6)
S4	Q(4,1)	Q(4,2)	Q(4,3)	Q(4,4)	Q(4,5)	Q(4,6)
S5	Q(5,1)	Q(5,2)	Q(5,3)	Q(5,4)	Q(5,5)	Q(5,6)
S6	Q(6,1)	Q(6,2)	Q(6,3)	Q(6,4)	Q(6,5)	Q(6,6)

4 Computational Results and Discussions

In this section, we compare the performance of Q-learning-assisted meta-heuristics and the classical meta-heuristics on a standard data set. To further verify the performance of the proposed algorithms, we also execute the Friedman test to compare the Q-learning-assisted meta-heuristics and one existing high-performance algorithm.

4.1 Experiment Setup

In this study, we take 60 instances with different scales, $n = \{40, 60, 80, 100\}$, $F = \{2, 3, 4, 5, 6\}$, and $s = \{2, 5, 8\}$ for our experiments [17]. Set the number of machines in each stage of each factory within 1 to 5. The running time of each algorithm is set according to the case scales with $n * F * s * t$ milliseconds, where t is a constant and is set to 20. Each instance is solved 10 times independently, and the average value is recorded. All algorithms are carried through the same experiment environment, i.e., a PC with an AMD Ryzen 5 processor (model 5600H) with Radeon Graphics. The CPU frequency is 3.30 GHz, and the memory size is 16 GB.

4.2 Results and Comparisons

To assess the performance of the proposed enhancement strategies, four basic meta-heuristics, four meta-heuristics with random local search operation selection, and four meta-heuristics with Q-learning-based local search operation selection are compared in respective groups. The results of four fundamental meta-heuristics, with random selection local search, and with Q-learning-based local search are shown in Tables 3–6. The experimental results consist of the average values in ten runs by four meta-heuristics and their variants across 60 examples. The best minimum values are highlighted in bold. From Tables 3–6, it can be seen that the results of the local search based on Q-learning have the most optimal solutions for each algorithm. It means that using Q-learning to enhance the meta-heuristics for solving DHFSP is effective.

Table 3: The results of GA and its variants

Instance	GA	GA_LS	GA_QL	Instance	GA	GA_LS	GA_QL	Instance	GA	GA_LS	GA_QL
2-40-2	1075	1035.2	**1001.7**	3-80-8	1899.8	1871.8	**1844.5**	5-60-5	823.7	820.9	**777.5**
2-40-5	1276.2	1288.6	**1232.7**	3-100-2	1819.9	1786.4	**1755.3**	5-60-8	1086.3	1093.9	**1061.8**
2-40-8	1535.8	1541.8	**1522.7**	3-100-5	2117.6	2093.4	**2063.4**	5-80-2	857.2	851.2	**800.7**
2-60-2	1565.1	1589.1	**1522.4**	3-100-8	2329.3	2305.5	**2282.8**	5-80-5	1170.5	1168.8	**1146.7**
2-60-5	1819.6	1800.1	**1788.7**	4-40-2	526	502.7	**462.7**	5-80-8	1345	1342.8	**1306.2**
2-60-8	2171.3	2141.6	**2134.2**	4-40-5	795	812.2	**758.5**	5-100-2	1121.7	1121.1	**1053.6**
2-80-2	2017.2	1995.6	**1988.2**	4-40-8	945.9	961.6	**909.6**	5-100-5	1351.8	1368.3	**1330.6**
2-80-5	2356.2	2354.3	**2323.4**	4-60-2	805.9	783.2	**714.4**	5-100-8	1666.2	1648.3	**1646.1**
2-80-8	2696.8	2708.7	**2668.9**	4-60-5	1014.9	992.2	**974.2**	6-40-2	426.9	425.3	**335.1**
2-100-2	2697	2640.8	**2630.3**	4-60-8	1052.6	1045.2	**1021**	6-40-5	538.7	549.1	**502.6**
2-100-5	2951.3	2952.9	**2937.2**	4-80-2	1165.1	1194.2	**1109.1**	6-40-8	792.2	754.6	**743.7**
2-100-8	3373.4	3381.8	**3360.7**	4-80-5	1313.4	1311	**1279.1**	6-60-2	564.5	520.5	**507**
3-40-2	549.2	545.3	**519.9**	4-80-8	1492.7	1480.9	**1452.1**	6-60-5	854.4	835.8	**806**
3-40-5	918.9	904.4	**887**	4-100-2	1169.8	1164	**1145.1**	6-60-8	996.5	988.9	**939.3**
3-40-8	1173.2	1168.1	**1141.8**	4-100-5	1547.8	1531.5	**1497.6**	6-80-2	752.8	779.8	**726.9**
3-60-2	1003.9	1005.2	**921.2**	4-100-8	1880.6	1900.3	**1867.5**	6-80-5	**1056.5**	1080.9	1057.2
3-60-5	1322.7	1307.4	**1270.6**	5-40-2	492.2	482.2	**422.4**	6-80-8	1209.1	1203.5	**1193.9**
3-60-8	1570.6	1546.8	**1526**	5-40-5	701.6	**651.8**	663.3	6-100-2	921.8	890.3	**878.1**
3-80-2	1338.3	1307.4	**1231.2**	5-40-8	788.4	799.2	**781.8**	6-100-5	1180.5	1193.7	**1162.3**
3-80-5	1587.9	1585.5	**1552.6**	5-60-2	689.7	700.6	**635.1**	6-100-8	1418.3	1407.3	**1403.1**

Table 4: The results of PSO and its variants

Instance	PSO	PSO_LS	PSO_QL	Instance	PSO	PSO_LS	PSO_QL	Instance	PSO	PSO_LS	PSO_QL
2-40-2	1094.8	1039.1	**1002.4**	3-80-8	1869.4	1900.7	**1834.8**	5-60-5	838.2	834.9	**810.5**
2-40-5	1278.1	1271.7	**1251.4**	3-100-2	1831.1	1828.3	**1749.4**	5-60-8	1091.3	1096.6	**1062.1**
2-40-8	1532.3	1526.2	**1500.1**	3-100-5	2099.3	2100.7	**2070.5**	5-80-2	840.7	842.8	**801.4**
2-60-2	1567.2	1543.3	**1520.4**	3-100-8	2316.7	2301.1	**2282.1**	5-80-5	1159.6	1155.2	**1127.2**
2-60-5	1814.2	1806.7	**1758.8**	4-40-2	506	474.9	**451**	5-80-8	1340	1356.2	**1319.6**
2-60-8	2155.9	2146.6	**2118.1**	4-40-5	793.8	807.2	**776.5**	5-100-2	1122.9	1115.2	**1065.1**
2-80-2	2012.4	2011.3	**1953.2**	4-40-8	941	975.1	**908.3**	5-100-5	1357	1344.3	**1319**
2-80-5	2366.1	2362.8	**2320.5**	4-60-2	793	802.2	**726.7**	5-100-8	1672.2	1653.9	**1632.2**
2-80-8	2711.6	2700.7	**2672.1**	4-60-5	985.5	984.1	**966.6**	6-40-2	398.6	400	**343.9**
2-100-2	2658.4	2648.9	**2601.5**	4-60-8	1053.5	1033.3	**1025.9**	6-40-5	545.1	529.4	**512.9**
2-100-5	2948.2	2947.7	**2902.7**	4-80-2	1171.4	1147.2	**1105.7**	6-40-8	802.6	794.8	**751.2**
2-100-8	3362.7	3368.7	**3360.7**	4-80-5	1302	1318.5	**1269.8**	6-60-2	587.5	557.3	**495.9**
3-40-2	551.3	536.8	**478.3**	4-80-8	1484	1480	**1453.6**	6-60-5	828.9	833.7	**805.1**
3-40-5	905.2	903.3	**865.8**	4-100-2	1172.9	1215	**1140.4**	6-60-8	977.2	987.8	**951**
3-40-8	1196.9	1165.8	**1140.1**	4-100-5	1532.2	1521.2	**1485.1**	6-80-2	751.2	758.5	**711.1**
3-60-2	986	1005.2	**914.6**	4-100-8	1867.4	1897.9	**1860.2**	6-80-5	1061.5	1069	**1037.6**
3-60-5	1317.8	1322.7	**1253.9**	5-40-2	487.5	490.4	**436.6**	6-80-8	1223.7	1214.3	**1188.7**
3-60-8	**1535.5**	1537.3	1541.1	5-40-5	683.3	**613.4**	640.8	6-100-2	898.4	901	**863.9**

(Continued)

Table 4 (continued)

Instance	PSO	PSO_LS	PSO_QL	Instance	PSO	PSO_LS	PSO_QL	Instance	PSO	PSO_LS	PSO_QL
3-80-2	1316	1289.3	**1263.6**	5-40-8	789.4	793.1	**770.7**	6-100-5	1183.3	1195	**1153.9**
3-80-5	1616.7	1592.9	**1551.3**	5-60-2	669.6	682.3	**637**	6-100-8	1413.3	1422.3	**1394.9**

Table 5: The results of DE and its variants

Instance	DE	DE_LS	DE_QL	Instance	DE	DE_LS	DE_QL	Instance	DE	DE_LS	DE_QL
2-40-2	1060.2	1044.2	**1004.7**	3-80-8	1887.8	1886.3	**1854.5**	5-60-5	842.7	834.4	**821**
2-40-5	1295.8	1269.3	**1264.8**	3-100-2	1829.5	1820.2	**1754.2**	5-60-8	**1084.9**	1099.5	1093.5
2-40-8	1529.2	1536.7	**1489.5**	3-100-5	2103.9	2083.2	**2066.1**	5-80-2	866.7	859.5	**818.8**
2-60-2	1551.6	1561.3	**1510.6**	3-100-8	2332.3	2324.6	**2247.5**	5-80-5	1149.7	1162.1	**1129.7**
2-60-5	1814.1	1794.9	**1775**	4-40-2	501.2	531.4	**460.9**	5-80-8	1363.8	**1340.9**	1350.7
2-60-8	2153	2154.8	**2125.6**	4-40-5	830.8	810.5	**759.8**	5-100-2	1117	1127.1	**1075.7**
2-80-2	1997.3	2015.7	**1956.3**	4-40-8	965.4	973.9	**941.9**	5-100-5	1363.2	1358.5	**1346.7**
2-80-5	2372.3	2347.6	**2313.4**	4-60-2	803.8	794.6	**707.5**	5-100-8	1648.3	1664.4	**1636.5**
2-80-8	2700	2703.2	**2664.8**	4-60-5	1002.8	**979.9**	996.4	6-40-2	398.5	411.6	**340.5**
2-100-2	2651.9	2668.1	**2602**	4-60-8	1054.1	1037.5	**1029.9**	6-40-5	538.1	524	**506**
2-100-5	2945.7	2952.1	**2911.9**	4-80-2	1174.1	1150.1	**1112.5**	6-40-8	795	792.1	**734.6**
2-100-8	3359.6	3370.4	**3312**	4-80-5	1316	1310.1	**1282.4**	6-60-2	548.5	572	**499.2**
3-40-2	562.9	564.2	**479.5**	4-80-8	1496	1494.5	**1474.4**	6-60-5	847.3	841.6	**814.3**
3-40-5	886.8	902.8	**875.1**	4-100-2	1180.4	1198.7	**1117.7**	6-60-8	980.4	997.7	**951.8**
3-40-8	1192.1	1180.2	**1140.2**	4-100-5	1520.2	1533.8	**1498.4**	6-80-2	764.3	763.5	**730**
3-60-2	980.2	978.3	**937.3**	4-100-8	1904.7	1884.9	**1852.4**	6-80-5	1082.6	1068.7	**1040.5**
3-60-5	1316.1	1303.9	**1278.8**	5-40-2	486.6	504.2	**443.2**	6-80-8	1222.7	1221.4	**1195.8**
3-60-8	1554.5	1552.4	**1532.6**	5-40-5	684.8	656.6	**643.6**	6-100-2	920	904.6	**882.1**
3-80-2	1307.2	1307.4	**1242.3**	5-40-8	797.1	790.4	**771.5**	6-100-5	1190.4	1185.9	**1156.4**
3-80-5	1595.7	1581.2	**1555.9**	5-60-2	677.5	688	**625.5**	6-100-8	1418.7	1420.9	**1379.7**

Table 6: The results of ABC and its variants

Instance	ABC	ABC_LS	ABC_QL	Instance	ABC	ABC_LS	ABC_QL	Instance	ABC	ABC_LS	ABC_QL
2-40-2	1040.4	1050.8	**997.6**	3-80-8	1869.7	1895.9	**1858.9**	5-60-5	841.7	827.1	**811.7**
2-40-5	1274.9	1281.8	**1265.8**	3-100-2	1823.5	1816.2	**1743.4**	5-60-8	1075.9	1092.1	**1063.6**
2-40-8	1538.2	1532.9	**1478.7**	3-100-5	2153.6	2148.8	**2102.8**	5-80-2	855.1	825.9	**785.1**
2-60-2	1532.9	1546.1	**1508**	3-100-8	2330.6	2336.2	**2273.3**	5-80-5	1150.5	1144.3	**1129.2**
2-60-5	1820.3	1792.1	**1771.5**	4-40-2	499.4	506.2	**424.4**	5-80-8	1363.5	1348.3	**1324.3**
2-60-8	2157.1	2148.3	**2125.1**	4-40-5	832.9	793.9	**760.8**	5-100-2	1125.8	1125.7	**1071.9**
2-80-2	2016.8	2016	**1979.3**	4-40-8	956.6	971	**911.6**	5-100-5	1361	1354.6	**1324.8**
2-80-5	2367.8	2365.4	**2335.1**	4-60-2	810.6	782.8	**710.8**	5-100-8	1664.6	1675	**1622.5**
2-80-8	2707	2711.1	**2674.3**	4-60-5	992.6	983.2	**968.9**	6-40-2	399.8	392.7	**339.6**
2-100-2	2661.6	2679.3	**2615.8**	4-60-8	1051.1	1022.6	**1012.3**	6-40-5	532.5	528.8	**501.7**
2-100-5	2960.3	2948.5	**2915.3**	4-80-2	1161.2	1173.2	**1105.8**	6-40-8	781.4	769.7	**719.1**

(Continued)

Table 6 (continued)

Instance	ABC	ABC_LS	ABC_QL	Instance	ABC	ABC_LS	ABC_QL	Instance	ABC	ABC_LS	ABC_QL
2-100-8	3343.6	3362.7	**3339.5**	4-80-5	1293.5	1302.3	**1272.1**	6-60-2	550.6	542.7	**484.7**
3-40-2	570.7	559.2	**504.2**	4-80-8	1517.1	1476.3	**1455.1**	6-60-5	832.9	836.8	**792**
3-40-5	924.5	895.3	**854.3**	4-100-2	1152.3	1160.9	**1126.5**	6-60-8	**978.4**	981.8	1038.5
3-40-8	1184.9	1184.8	**1139.2**	4-100-5	1520	1515.2	**1476**	6-80-2	722.4	730	**712.5**
3-60-2	978.2	1004.1	**934.6**	4-100-8	1905.2	1899	**1854.1**	6-80-5	1068	**1047**	1050.9
3-60-5	1326.2	1306.7	**1287.4**	5-40-2	495.4	479.2	**446**	6-80-8	1220.3	1210.3	**1163.2**
3-60-8	1551.7	1562.8	**1533.7**	5-40-5	701.1	646.1	**636.2**	6-100-2	898.9	919.1	**855.7**
3-80-2	1326.2	1273.9	**1237.5**	5-40-8	808.8	798.8	**777.6**	6-100-5	1181.8	1182.3	**1159.2**
3-80-5	1632.6	1582.1	**1575.2**	5-60-2	691.7	659.8	**631.3**	6-100-8	1423.8	1423	**1369.7**

Coefficient of Variation (CV) is the coefficient of variation, which is used to measure the degree of variation in experimental results. It is particularly useful when comparing data sets with different units or scales, as it standardizes the standard deviation relative to the mean.

$$\mathrm{CV} = \frac{\mathrm{SD}}{\mathrm{Avg}} \tag{11}$$

where Avg is the average of the results in 10 runs for one instance and SD is the standard deviation. The CV values of four meta-heuristics and their variants are shown in Table 7. All of four meta-heuristics the best CV are GA_QL, PSO_QL, DE_QL and ABC_QL.

Table 7: CV values by four algorithms and their variants

Instance	CV		CV		CV		CV
GA	0.032	PSO	0.035	DE	0.031	ABC	0.029
GA_LS	0.030	PSO_LS	0.031	DE_LS	0.030	ABC_LS	0.029
GA_QL	0.017	PSO_QL	0.016	DE_QL	0.013	ABC_QL	0.012

4.3 Statistical Test

In this section, we conduct the Friedman test on four meta-heuristics combined with Q-learning and improved iterated greedy algorithm (IIG) [33] to compare their performance. The reason for choosing IIG is that the comparison of the IIG algorithm with IG_VND [34], GA, IG [35], and iterated local search (ILS) [36] shows that the IIG algorithm has the best performance. The results of the Friedman test are reported in Table 8. The asymptotic significance (Asymp. Sig.) is 0.000, which is less than the specified significance level of 0.05. It means that there are significant differences among the five algorithms. The ranks of the five algorithms are shown in Fig. 11. The algorithm with a smaller rank value has better performance. From Fig. 11, we can see that PSO_QL has the smallest rank value (2.175), indicating the most competitiveness of the PSO_QL. The distribution by ranks of five algorithms is depicted in Fig. 12. Through the rank value and rank distribution of the Friedman test, it can be obtained that the PSO_QL is the most competitive one among the five algorithms.

Table 8: The statistical results of the Friedman test

Test statistics	
N	60
Chi-square	103.376
df	4
Asymp. Sig.	0.000

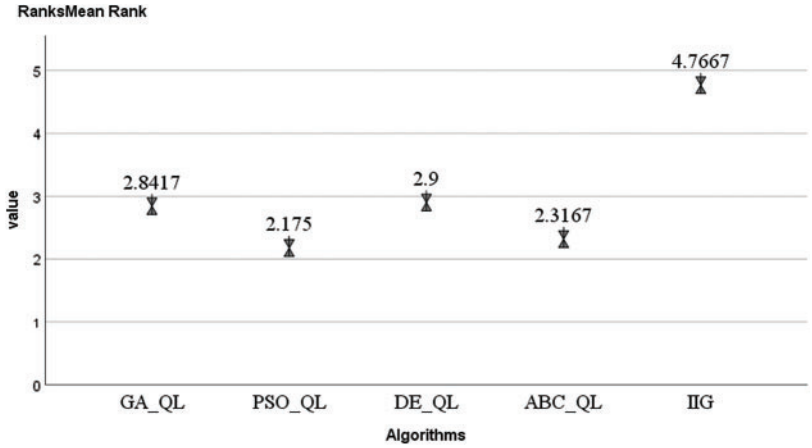

Figure 11: The rank value of algorithms

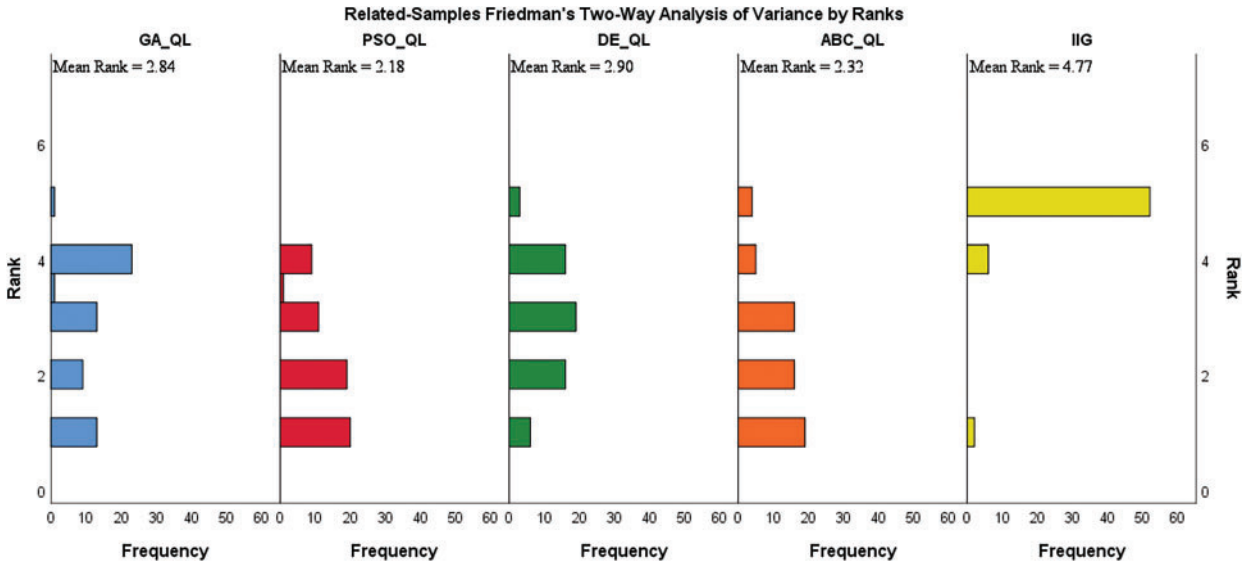

Figure 12: The distribution by ranks of algorithms

5 Conclusions and Future Directions

This paper introduces the integration of Q-learning and meta-heuristics to address DHFSP. Four meta-heuristics (GA, PSO, DE, ABC) are utilized, and six local search strategies are developed to avoid

algorithms falling into local optimality based on the feature of DHFSP. The meta-heuristics combine Q-learning to select six types of local searches to choose the appropriate local search strategy more efficiently. In the experiments, 60 different examples are used to analyze algorithms' performances. The experimental results show that the PSO_QL exhibits the highest competitiveness among all the compared algorithms.

Based on this work, the following issues are planned to be addressed in the future. (1) In DHFSP, multiple optimization goals are considered, including energy efficiency and cost. (2) Consider comparing Q-learning with other reinforcement learning methods, such as Sarsa. (3) There are many complex situations in the actual shop, such as job setup time. We will consider adding more constraints to the concerned DHFSP problems.

Acknowledgement: None.

Funding Statement: This study is partially supported by the Guangdong Basic and Applied Basic Research Foundation (2023A1515011531), the National Natural Science Foundation of China under Grant 62173356, the Science and Technology Development Fund (FDCT), Macau SAR, under Grant 0019/2021/A, Zhuhai Industry-University-Research Project with Hongkong and Macao under Grant ZH22017002210014PWC, the Key Technologies for Scheduling and Optimization of Complex Distributed Manufacturing Systems (22JR10KA007).

Author Contributions: Study conception and design: Qianyao Zhu and Kaizhou Gao; data collection: Qianyao Zhu; analysis and interpretation of results: Qianyao Zhu; draft manuscript preparation: Qianyao Zhu; review and editing: Kaizhou Gao and Adam Slowik. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. T. Wang, Q. -K. Pan, L. Gao, Z. -H. Miao, and H. -Y. Sang, "Modeling and scheduling a constrained flowshop in distributed manufacturing environments," *J. Manuf. Syst.*, vol. 72, pp. 519–535, 2024.

[2] S. Wang, X. Li, L. Gao, and J. Li, "A multi-disjunctive-graph model-based memetic algorithm for the distributed job shop scheduling problem," *Adv. Eng. Inform.*, vol. 60, 2024, Art. no. 102401. doi: 10.1016/j.aei.2024.102401.

[3] A. M. Fathollahi-Fard, L. Woodward, and O. Akhrif, "A distributed permutation flow-shop considering sustainability criteria and real-time scheduling," *J. Ind. Inf. Integr.*, vol. 39, 2024, Art. no. 100598. doi: 10.1016/j.jii.2024.100598.

[4] D. Lei, J. Zhang, and H. Liu, "An adaptive two-class teaching-learning-based optimization for energy-efficient hybrid flow shop scheduling problems with additional resources," *Symmetry*, vol. 16, no. 2, 2024, Art. no. 203. doi: 10.3390/sym16020203.

[5] M. Geetha, R. Chandra Guru Sekar, M. K. Marichelvam, and Ö. Tosun, "A sequential hybrid optimization algorithm (SHOA) to solve the hybrid flow shop scheduling problems to minimize carbon footprint," *Processes*, vol. 12, no. 1, 2024, Art. no. 143. doi: 10.3390/pr12010143.

[6] C. -C. Lin, Y. -C. Peng, Y. -S. Chang, and C. -H. Chang, "Reentrant hybrid flow shop scheduling with stockers in automated material handling systems using deep reinforcement learning," *Comput. Indus. Eng.*, vol. 189, no. 3, 2024, Art. no. 109995. doi: 10.1016/j.cie.2024.109995.

[7] O. Moursli and Y. Pochet, "A branch-and-bound algorithm for the hybrid flowshop," *Int. J. Prod. Econ.*, vol. 64, pp. 113–125, 2000. doi: 10.1016/S0925-5273(99)00051-1.

[8] L. Hidri and A. Gharbi, "New efficient lower bound for the hybrid flow shop scheduling problem with multiprocessor tasks," *IEEE Access*, vol. 5, pp. 6121–6133, 2017. doi: 10.1109/ACCESS.2017.2696118.

[9] B. Khurshid, S. Maqsood, Y. Khurshid, K. Naeem, and Q. S. Khalid, "A hybridization of evolution strategies with iterated greedy algorithm for no-wait flow shop scheduling problems," *Sci. Rep.*, vol. 14, no. 1, 2024, Art. no. 2376. doi: 10.1038/s41598-023-47729-x.

[10] M. Y. Wang, S. P. Sethi, and S. L. van de Velde, "Minimizing makespan in a class of reentrant shops," *Oper. Res.*, vol. 45, no. 5, pp. 702–712, 1997. doi: 10.1287/opre.45.5.702.

[11] Y. Zhu, Q. Tang, L. Cheng, L. Zhao, G. Jiang and Y. Lu, "Solving multi-objective hybrid flowshop lot-streaming scheduling with consistent and limited sub-lots via a knowledge-based memetic algorithm," *J. Manuf. Syst.*, vol. 73, no. 5, pp. 106–125, 2024. doi: 10.1016/j.jmsy.2024.01.006.

[12] Z. Shao, W. Shao, J. Chen, and D. Pi, "A feedback learning-based selection hyper-heuristic for distributed heterogeneous hybrid blocking flow-shop scheduling problem with flexible assembly and setup time," *Eng. Appl. Artif. Intell.*, vol. 131, no. 4, 2024, Art. no. 107818. doi: 10.1016/j.engappai.2023.107818.

[13] G. Ziadlou, S. Emami, and E. Asadi-Gangraj, "Network configuration distributed production scheduling problem: A constraint programming approach," *Comput. Indus. Eng.*, vol. 188, no. 10, 2024, Art. no. 109916. doi: 10.1016/j.cie.2024.109916.

[14] B. Khurshid and S. Maqsood, "A hybrid evolution strategies-simulated annealing algorithm for job shop scheduling problems," *Eng. Appl. Artif. Intell.*, vol. 133, 2024, Art. no. 108016. doi: 10.1016/j.engappai.2024.108016.

[15] J. Wang, H. Tang, and D. Lei, "A feedback-based artificial bee colony algorithm for energy-efficient flexible flow shop scheduling problem with batch processing machines," *Appl. Soft Comput.*, vol. 153, no. 1, 2024, Art. no. 111254. doi: 10.1016/j.asoc.2024.111254.

[16] Y. Li, X. Li, L. Gao, and L. Meng, "An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times," *Comput. Indus. Eng.*, vol. 147, no. 2, 2020, Art. no. 106638. doi: 10.1016/j.cie.2020.106638.

[17] Y. Li *et al.*, "A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 59, no. 13, pp. 3880–3899, 2020. doi: 10.1080/00207543.2020.1753897.

[18] X. -R. Tao, Q. -K. Pan, and L. Gao, "An efficient self-adaptive artificial bee colony algorithm for the distributed resource-constrained hybrid flowshop problem," *Comput. Indus. Eng.*, vol. 169, no. 18, 2022, Art. no. 108200. doi: 10.1016/j.cie.2022.108200.

[19] J. -H. Hao, J. -Q. Li, Y. Du, M. -X. Song, P. Duan and Y. -Y. Zhang, "Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm," *IEEE Access*, vol. 7, pp. 66879–66894, 2019. doi: 10.1109/ACCESS.2019.2917273.

[20] C. Lu, J. Zhou, L. Gao, X. Li, and J. Wang, "Modeling and multi-objective optimization for energy-aware scheduling of distributed hybrid flow-shop," *Appl. Soft Comput.*, vol. 156, no. 1, 2024, Art. no. 111508. doi: 10.1016/j.asoc.2024.111508.

[21] X. Li, Q. Zhao, H. Tang, S. Yang, D. Lei and X. Wang, "Joint scheduling optimisation method for the machining and heat-treatment of hydraulic cylinders based on improved multi-objective migrating birds optimisation," *Manuf. Syst.*, vol. 73, pp. 170–191, 2024.

[22] J. Cai, D. Lei, and M. Li, "A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop," *Int. J. Prod. Res.*, vol. 59, no. 18, pp. 5404–5421, 2021. doi: 10.1080/00207543.2020.1780333.

[23] M. Wu, "Application of particle swarm optimisation algorithm incorporating frog-leaping algorithm in optimal scheduling for production management in manufacturing plant," *Int. J. Interact. Des. Manuf.*, vol. 286, no. 1, 2024, Art. no. 32. doi: 10.1007/s12008-024-01767-5.

[24] J. -J. Wang and L. Wang, "A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 26, no. 3, pp. 461–475, Jun. 2022. doi: 10.1109/TEVC.2021.3106168.

[25] Z. Zhang, Z. Shao, W. Shao, J. Chen, and D. Pi, "MRLM: A meta reinforcement learning-based metaheuristic for hybrid flow-shop scheduling problem with learning and forgetting effects," *Swarm Evol. Comput.*, vol. 85, no. 1, 2024, Art. no. 101479. doi: 10.1016/j.swevo.2024.101479.

[26] H. Yu, K. Gao, N. Wu, M. Zhou, P. N. Suganthan and S. Wang, "Scheduling multiobjective dynamic surgery problems via Q-learning-based meta-heuristics," *IEEE Trans. Syst. Man, Cyber.: Syst.*, vol. 54, no. 6, pp. 3321–3333, Jun. 2024.

[27] F. Q. Wang, Y. P. Fu, K. Z. Gao, Y. X. Wu, and S. Gao, "A Q-learning-based hybrid meta-heuristic for integrated scheduling of disassembly and reprocessing processes considering product structures and stochasticity," *Complex Syst. Model. Simul.*, vol. 4, no. 4, pp. 184–209, 2024. doi: 10.23919/CSMS.2024.0007.

[28] C. Luo, W. Gong, F. Ming, and C. Lu, "A Q-learning memetic algorithm for energy-efficient heterogeneous distributed assembly permutation flowshop scheduling considering priorities," *Swarm Evol. Comput.*, vol. 85, 2024, Art. no. 101497. doi: 10.1016/j.swevo.2024.101497.

[29] Y. Liu, F. Zhang, Y. Sun, and M. Zhang, "Evolutionary trainer-based deep Q-network for dynamic flexible job shop scheduling," *IEEE T. Evolut. Comput.*, doi: 10.1109/TEVC.2024.3367181.

[30] F. Zhao, G. Zhou, T. Xu, N. Zhu, and Jonrinaldi, "A knowledge-driven cooperative scatter search algorithm with reinforcement learning for the distributed blocking flow shop scheduling problem," *Expert Syst. App.*, vol. 230, 2023, Art. no. 120571. doi: 10.1016/j.eswa.2023.120571.

[31] Q. -K. Pan, L. Gao, X. -Y. Li, and K. -Z. Gao, "Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times," *Appl. Math. Comput.*, vol. 303, pp. 89–112, 2017. doi: 10.1016/j.amc.2017.01.004.

[32] K. -C. Ying and S. -W. Lin, "Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks," *Expert Syst. Appl.*, vol. 92, no. 2, pp. 132–141, 2018. doi: 10.1016/j.eswa.2017.09.032.

[33] C. Lu, J. Zheng, L. Yin, and R. Wang, "An improved iterated greedy algorithm for the distributed hybrid flowshop scheduling problem," *Eng. Optim.*, vol. 56, no. 5, pp. 792–810, 2023. doi: 10.1080/0305215X.2023.2198768.

[34] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, pp. 754–768, 2010. doi: 10.1016/j.cor.2009.06.019.

[35] H. Oztop, M. F. Tasgetiren, D. T. Eliiyi, and Q. K. Pan, "Metaheuristic algorithms for the hybrid flowshop scheduling problem," *Comput. Oper. Res.*, vol. 111, no. 1, pp. 177–196, 2019. doi: 10.1016/j.cor.2019.06.009.

[36] W. S. Shao, D. C. Pi, and Z. S. Shao, "Local search methods for a distributed assembly no-idle flow shop scheduling problem," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1945–1956, 2019. doi: 10.1109/JSYST.2018.2825337.