# Energy-Efficient Iterative Greedy Algorithm for the Distributed Hybrid Flow Shop Scheduling With Blocking Constraints

Haoxiang Qin, Yuyan Han, *Member, IEEE*, Qingda Chen, *Member, IEEE*, Ling Wang, *Member, IEEE*, Yuting Wang, Junqing Li, *Member, IEEE*, and Yiping Liu, *Member, IEEE*



***Abstract*— With the global energy shortage, climate anomalies, environmental pollution becoming increasingly prominent, energy saving scheduling has attracted more and more concern than before. This paper studies the energy-efficient distributed hybrid flow-shop scheduling problem (DHFSP) with blocking constraints. Our aim is to find the job sequence with low energy consumption as much as possible in a limited time. In this paper, we formulate a mathematical model of the DHFSP with blocking constraints and propose an improved iterative greedy (IG) algorithm to optimize the energy consumption of job sequence. In the proposed algorithm, first, a problem-specific strategy is presented, namely, the global search strategy, which can assign appropriate jobs to the factory and minimize the energy consumption of each processing factory. Next, a new selection mechanism inspired by Q-learning is proposed to provide strategic guidance for factory scheduling. This selection mechanism provides historical experience for different factories. Finally, five types of local search strategies are designed for blocking constraints of machines and sequence to be scheduled. These proposed strategies can further improve the local search ability of the QIG algorithm and reduce the energy consumption caused by blocking. Simulation results and statistical analysis on 90 test problems show that the proposed algorithm is superior to several high-performance algorithms on convergence rate and quality of solution.**

Manuscript received 11 February 2023; revised 7 March 2023; accepted 20 March 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61973203, 61803192, 61966012, and 62106073, in part by the Guangyue Young Scholar Innovation Team of Liaocheng University under Grant LCUGYTD2022-03, in part by the Shandong Province Colleges and Universities through Youth Innovation Talent Introduction and Education Program, in part by the Natural Science Foundation of Hunan Province of China under Grant 2021JJ40116, and in part by the Natural Science Foundation of Shandong Province under Grants ZR2021QE195 and ZR2021QF036. *(Corresponding author: Yuyan Han.)*

Haoxiang Qin, Yuyan Han, and Yuting Wang are with the School of Computer Science, Liaocheng University, Liaocheng 252059, China (e-mail: 987352978@qq.com; hanyuyan@lcu-cs.com; wangyuting@lcu-cs.com).

Qingda Chen is with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: cqd0309@126.com).

Ling Wang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: wangling@tsinghua.edu.cn).

Junqing Li is with the School of Information and Engineering, Shandong Normal University, Jinan 250014, China, and also with the School of Computer Science, Liaocheng University, Liaocheng 252059, China (e-mail: lijunqing @lcu-cs.com).

Yiping Liu is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: yiping0liu@ gmail.com).

Digital Object Identifier 10.1109/TETCI.2023.3271331

***Index Terms*—Energy-efficient, distributed hybrid flow-shop scheduling, blocking constraints, iterative greedy algorithm, selection mechanism.**

## I. Introduction

With the development of social economy and science, the demand for energy has expanded rapidly. Coal, oil and other non-renewable fossil resources are becoming more and more important. Sustainable development and energy conservation have become a matter of importance for countries [1], [2]. Manufacturing is an energy-intensive sector that consumes nearly one-third of the world's energy and produces 36% of the world's carbon dioxide [3]. As a part of the manufacturing industry, intelligent optimization and scheduling play a very important role in the machining process for the improvement of resource utilization and energy consumption [4], [5]. Refer to [6], [7], [8], for reducing the production cost and energy consumption, many enterprises begin to use intelligent optimization algorithms to find better scheduling sequences.

In real-world, to improve the productivity of production process, balance the flexibility of each processing stage and reduce the impact of the bottleneck stage [9], enterprises start to set identical parallel machines in each stage to process jobs. The production line scheduling problem is noted as the hybrid flow shop scheduling problem (HFSP) [10]. However, in some cases, due to limits of storage space, product characteristics, or technology [11], there is usually no buffer between any adjacent parallel machines in actual process of scheduling. This problem is also named as the blocking HFSP (BHFSP). In addition, with the increasing of market competition, the centralized manufacturing approach has been hard to meet the current market demand flexibly [12], [13], [14], [15]. Therefore, some companies begin using a distributed production scheduling mode to share the production pressure of the factory. Due to the emergence of distributed production mode, many scholars started to conduct extensive and in-depth research on the distributed flow shop (multi-plant) scheduling problem (DPFSP) [16], [17]. In DPFSP, companies set up multiple factories to process the same batch of products in parallel, and it allows for more efficient resource allocation, decentralization of companies' production pressure as well as reduction of product production cycles and risks [18], [19], [20].

In this paper, the energy-efficient distributed hybrid flow-shop scheduling problem (DHFSP) with blocking constraints is studied. Due to the blockage of jobs, machines cannot perform normal machining operations, causing unnecessary energy consumption. This also directly leads to a decrease in processing efficiency and raises production costs for the company. Therefore, to reduce the occurrence of a job blocking, it is of great importance to design strategies that can seek a reasonable scheme and help factories find a near-optimal job sequence (i.e., the one with the lowest energy consumption) in a large and irregular job sorting.

However, the generation of blocking conditions usually changes uncertainly with the change of job sequencing in DHFSP, and these factors also lead to irregular changes in the optimal scheduling sequence. Thus, it is difficult to find a satisfactory solution in a short time using traditional mathematical methods. The intelligent optimization algorithm has been widely used by scholars to solve the flow shop scheduling problem for a long time and has achieved good results. This paper designs optimization methods for job sequencing under uncertainty for improving the productivity of enterprises and reducing energy consumption in sequence processing. Before designing corresponding strategies for DHFSP with blocking constraints, we first analyze existing challenges and difficulties in this problem, the details are as follows:

1) The quality of solution will change irregularly. Energy consumption of job sequence is affected by blocking constraints, resulting in irregular change and scope reduction of the sequence order in scheduling process. It is sometimes difficult to find a better feasible solution in a limited time.
2) Assigning jobs to factories while ensuring suitability and efficiency, simultaneously. In the allocation process, due to the inefficient strategies, it may result in huge energy consumption of factories and reduced enterprise productivity.
3) Each processing factory is usually enclosed and unrelated. In specific scheduling process, the processing environment of each factory is isolated from each other. Although the distributed scheduling mode shares part of the production pressure, such a closed processing environment is not conducive to factory processing to a certain extent.
4) Blocking conditions of factories are difficult to be improved. Blocking constraints limit the local search range of each factory, and there is a complex nonlinear relationship between these restrictions and job sequence. The scheduling sequence solution is easier to fall into local optimum if the job is blocked on the current machine.

To the best of our knowledge, there is little research on DHFSP with blocking constraints, but scholars have come to study the related scheduling problem. To determine the scheduling sequence and minimize the weighted completion time with a work shift, Nejati et al. [21] solve the HFSP with parallel machines. For minimizing the makespan and total flowtime, Marichelvam et al. [22] address the scheduling problem with parallel machines. Zhang et al. [23] utilized the shortest waiting time rule and a combined neighborhood search strategy to solve the HFSP with lot-streaming. To solve the blocking HFSP (BHFSP), Qin et al. [24] designed the local and global perturbation strategies based on the blocking constraints to optimize the energy consumption. Aqil et al. [25] investigated the BHFSP under the sequence-dependent setup time constraint to minimize the earliness and total tardiness with parallel machines. For solving the DPFSP, Bargaoui et al. [26] make the solution jump out of the local optimum and further improve the quality of the scheduling sequence. Wang et al. [27] analyzed the critical path of the job sequence and solve the distributed assembly permutation flow-shop scheduling problem (DAPFSP). Then, Wang et al. [13] designed a mixed-integer linear programming model of the DHFSP with heterogeneous factories and used the bi-population cooperative memetic algorithm to solve this problem. Shao et al. [12] developed the DNEH (Nawaz–Enscore–Ham) with the smallest-medium rule and the multi-neighborhood iterated greedy method to solve the DHFSP.

Although the above-mentioned researches have made significant contributions to the optimized scheduling problems, they still have the following limitations:

1) They did not make targeted strategy design for blocking conditions. When consider the blocking condition of jobs, energy consumption of the scheduling sequence will alter irregularly with the change of arrangement order. The existing Intelligent optimization scheduling algorithms are not suitable for solving the DHFSP with blocking constraints.
2) Too long allocation time for jobs will reduce the search performance of the algorithm and production efficiency. It is difficult to reduce the impact of blocking constraints quickly in such a large search region and find the best allocation scheme.
3) Neither of them considers breaking the closed processing state between different factories. The scheduling environment is isolated, resulting in production occlusion among different factories.
4) They do not take the blocking conditions of parallel machines of factories into account. When blocking constraints are considered, the local search scope of the scheduled sequence is narrowed, which makes its solution easily fall into the local optimal in the iterative process.

For solving these problems mentioned above, we reviewed and compared the performance of different intelligent optimization algorithms [9], [28], [29], [30], [31], hoping to find a suitable algorithm and making further improvements based on the characteristics of DHFSP with blocking constraints. Finally, we learned that Iterated Greedy (IG) algorithm [32] shows its superiority in many scheduling problems compared to other intelligent optimization algorithms. It has fewer parameters and a simple structure, which makes it easy to be realized. Thus, in this paper, we propose an improved IG algorithm to reduce the energy consumption of DHFSP with blocking constraints.

The main contributions of this paper are given as follows.

1) In this paper, to satisfy the manufacture demands, we first design the mathematical model of DHFSP with blocking constraints for minimizing the energy consumption and use the gurobi to verify its correctness.

2) To explore the promising solution more quickly, this paper proposes a global search strategy with the consideration of job sequences in different factories. The proposed strategy can improve the global search ability of the algorithm, and further reduce energy waste by adjusting the job arrangement order.
3) A new selection mechanism inspired by Q-learning is integrated into the IG algorithm to break the closed state between factories. This selection mechanism realizes experience sharing and interaction between different factories. Experiments show that it can slightly improve the performance of the proposed algorithm.
4) To further improve the local search ability of IG algorithm, we present five local search strategies for reducing the energy consumption. These strategies can perform a wide range of adjustments to blocking jobs and help find better solutions.

The remainder of this paper is organized as follows. Section II reviews some existing literature that solve the related problems. In section III, the mathematical model of DHFSP with blocking constraints is formulated. Section IV proposed the framework and details of the improved IG algorithm. In Section V, comparison results show the performance of the proposed algorithm. Section VI gives the concluding remarks and directions for future research.

## II. Literature Review

The DHFSP with blocking constraints, to the best of our knowledge, has not been previously studied in the existing researches. Therefore, we review the closely related contributions, e.g., HFSP, BHFSP, DPFSP, DHFSP.

In recent years, many types of intelligent optimization algorithms have been proposed to solve the HFSP and its extension problem, BHFSP.

For solving HFSP, A hybrid iterated local search algorithm is proposed to solve the HFSP for economic lot-sizing and sequence [33]. Kurdi [34] designed an AC system with a novel Non-Daemon Actions procedure for multiprocessor task scheduling in HFSP. Liu et al. [35] designed the hybrid algorithm to solve the specialized two-stage HFSP with parallel batching machines. Ztop et al. [36] suggested four variants of iterated greedy algorithms and a variable block insertion heuristic for the HFSP with total flowtime minimization. For minimizing the completion time, Yu et al. [9] proposed a genetic algorithm to solve the HFSP with machine eligibility and unrelated machines. The above algorithms effectively solve the problem of HFSP, they did not consider the blocking constraints into the HFSP. However, this condition is very common in the real world, such as concrete blocks [37] and metalwork [38]. Therefore, later, we look for some currently published literature on HFSP with blocking constraints (BHFSP).

The presence of the BHFSP in manufacturing industry system has been the subject of many researches. Luo et al. [38] presented a genetic algorithm (GA) algorithm to investigate a two-stage BHFSP in real-world metal-working company, the objective of makespan is optimized in this literature. Nakkaew et al. cite 2016 Acom presented a GA and a discrete artifical bee colony (DABC) algorithm to solve the BHFSP with sequence dependent setup times with the minimization of the overall production time. Missaoui et al. [39] proposed a meta heuristic centered on IG method to investigate the BHFSP with optimizing the sum of the tardiness and earliness. The above studies are all carried out on BHFSP, and they have solved this problem well. However, in these problems, distributed scheduling environment is not considered, let alone the distributed HFSP (DHFSP) with blocking constraints with energy consumption as the optimization goal.

In order to meet the needs of the modern market, many enterprises will establish multiple factories to improve the processing efficiency of products, which has become a new research hotspot: DPFSP. Many scholars have designed a serious of effective meta-heuristic algorithms to solve the DPFSP. Jian et al. [40] proposed a new tabu search algorithm to solve the DPFSP. Rifai et al. [41] proposed a multi-objective adaptive large neighborhood search algorithm to solve the distributed reentrant flow shop scheduling (DRPFS) problem with three objectives, the total cost, the maximum completion time, and the average delay. Pan et al. [20] proposed a series of algorithms based on construct heuristic and meta-heuristic frameworks to solve the DPFSP and the DAPFSP [42]. Ruiz et al. [43] used the IG algorithm to solve this problem and proved its effectiveness. Recently, Pan et al. [44] proposed an effective co-evolutionary algorithm to solve the distributed flow-shop group scheduling problems. Ochi et al. [45] designed the bounded search Iterated Greedy Algorithm BSIG to solve the DAPFSP problem. On the same problem, Huang et al. [46] proposed a group-think based IG algorithm (GIGA)) to optimize the total flow time. Recently, Shao et al. [47] proposed an efficient IG algorithm to solve DPFSP with blocking to minimize the maximum completion time. Similarly, Chen et al. [48] also used IG algorithm to solve DPFSP with blocking constraints. These algorithms effectively solve the multi-factory scheduling problem. However, the situation of the parallel machines is not included. Thus, next, we look for some literatures that integrate the distributed and parallel machine scheduling, and investigate whether the previously mentioned blocking constraints and energy consumption are considered as targets.

According to our survey, there are a few studies on DHFSP. For example, Ying et al. [49] proposed a self-tuning iterated Greedy (SIG) algorithm to optimize the maximum completion time of the job sequence. Lei et al. [50] proposed the Shuffled Frog-leaping algorithm with Memeplex grouping (MGSFLA) to solve the distributed two-stage hybrid flow shop scheduling problem with sequence-dependent setup times (DTHFSP). Wang et al. [13] proposed a bi-population cooperative memetic algorithm (BCMA) for solving the heterogeneous factories of the DHFSP. Shao et al. [12] proposed the DNEH with sharmedium rule and the multi-neighborhood iterative greedy algorithm to solve the DHFSP. Li et al. [51] proposed the hybrid discrete artificial bee colony algorithm to solve a parallel batching DPFSP with deteriorating jobs. Zheng et al. [52] proposed a cooperative coevolution algorithm to solve the multi-objective fuzzy DHFSP with fuzzy machining time and fuzzy delivery time. The above studies comprehensively consider the conditions of parallel machines and multiple factories. These papers are very new and efficient, but they do not consider the blocking condition of the jobs and energy consumption, simultaneously. As a result, we find that none of the published papers solve the DHFSP with blocking constraint with energy consumption as the optimization objective.

The existing literature mentioned above only considers two or three of conditions (e.g., the parallel machines, blocking constraints, distributed environment, energy consumption). However, all kinds of these situations widely exist in the real-world, such as the production of glass and concrete. In this paper, we study the problem with these four conditions simultaneously. Then, we propose an improved IG algorithm to optimize energy consumption. Based on the advantages of IG algorithm, we design the global and local search strategies based on swap operators for blocking constraints. The proposed strategies can reduce the computational complexity of the algorithm and increase the quality of solutions.

## III. Problem Description

This section first proposes the DHFSP model with blocking constraints. Then it gives the mathematical definitions of the assumptions, parameters, optimization objective, and constraints of this problem in Section III-A. In Section III-B, it gives the Gantt charts with and without blocking constraints to illustrate the impact of restrictions on energy consumption.

### A. Problem Formulation

The DHFSP with blocking constraints is formulated as follows.It comprises $F$($f$=1,...,$F$) factories and each factory $f$ contains a set of parallel machines with $S$($s$=1,...,$S$) stages. Each stage has a different number of machines. A collection of $J$($j$=1,...,$J$) jobs are assigned to any one of these factories to process orderly. In all factories, there is no buffer between any two continuous stages. When jobs are finished but all machines in the next stage are in processing state, jobs will be blocked on the current production line until one of the downstream machines is available. Once the blocking condition occurs, it will affect the overall production efficiency of the sequence and increase the energy consumption. In this paper, the schedule problem contains two parts: allocating all jobs to one of the $F$ identical factories and determining the processing order with minimum energy consumption. According to the literature [10], [13], [53], we give the assumptions, parameters, decision variables, objective, and constraints of the DHFSP with blocking constraints.

The problem can be solved in three parts: job processing energy consumption, blocking energy consumption, and machine idle energy consumption. In the MILP model, the difference between the departure time and the completion time of each job on the same machine is the corresponding machine blocking time. If the departure time of the job is greater than its completion time, it means that the job is blocked on the current machine. In this paper, when establishing the mathematical model, the first step is to determine the factory allocation problem, and the second step is allocating machines according to the relationship between the completion time and the departure time of jobs, then process the job sequence and calculate the objective value.

**Assumptions:**

1) All jobs and machines are available at time zero.
2) The processing time of each job is predefined.
3) Each job must choose exactly one factory to process and once a job is assigned in one factory, it cannot be assigned to other factories.
4) The processing order is determined in the first stage, and jobs in this order are processed from the first stage to the last stage.
5) Each job must pass through all stages, and at any given time, a job can only be processed on exactly one machine and each machine can only process one job.
6) There is no buffer between any two continuous stages.
7) Both blocking and idle states of machines are considered.
8) Once a job is completed at the current machine, it must be transported to the next stage immediately.
9) No interruption and pre-emption are allowed.
10) Other time-consuming operations are included in processing time.

*1) Parameters:*

$J$: Number of jobs.
$F$: Number of factories.
$S$: Number of stages.
$\varGamma$: The set of jobs, $\varGamma \in \{1, 2, \ldots, N\}$.
$\varLambda$: The set of factories, $\varLambda \in \{1, 2 \ldots, F\}$.
$\varOmega$: The set of stages, $\varOmega \in \{1, 2 \ldots, S\}$.
$j, j_1, j_2$: Index of jobs, $j, j_1, j_2 \in \varGamma$.
$f$: Index of factories, $f \in \varLambda$.
$s$: Index of stages, $s \in \varOmega$.
$M_{f,s}$: Number of parallel machines at stage $s$ in factory $f$,.
$m$: Index of machines at stage $s$ in factory $f$, $m \in \{1, \ldots, M_{f,s}\}$.
$p_{j,s}$: Processing time of job $j$ at stage $s$.
$EC_s^{Process}$: The energy consumption per unit time of a job which is processed at stage $s$.
$EC_s^{Blocking}$: The energy consumption per unit time of a job which is blocked at stage $s$.
$EC_s^{Idle}$: The energy consumption per unit time of a machine which is in idle state at stage $s$.
$h$: Sufficiently large positive number.

*2) Decision variables:*

$C_{j,s}$: The completion time of job $j$ at stage $s$.
$D_{j,s}$: The departure time of job $j$ at stage $s$.
$E_{f,s,m}$: The shutdown time of machine $m$ at stage $s$ in factory $f$.
$x_{j,f}$: Binary decision variable, 1 if job $j$ is assigned in factory $f$, 0 otherwise.
$y_{j,f,s,m}$: Binary decision variable, 1 if the job $j$ is processed on machine $m$ at stage $s$ in factory $f$, 0 otherwise.
$z_{j_1,j_2,s}$: Binary decision variables, 1 if both the job $j_1$ is processed before the job $j_2$ at stage $s$, 0 otherwise.
$PEC$: The total energy consumption of machines when they stay at the processing state.

$BEC$: The total energy consumption of machines when they stay at the blocking state.

$IEC$: The total energy consumption of machines when they are at the idle state.

*3) Objective:*

$$Minimize\,(PEC + BEC + IEC)$$

$$\sum_{f=1}^{F} x_{j,f} = 1,\ \forall j \in \Gamma \tag{1}$$

$$\sum_{m=1}^{M_{f,s}} y_{j,f,s,m} = x_{j,f},\ \forall f \in \Lambda,\ \forall s \in \Omega,\ \forall j \in \Gamma \tag{2}$$

$$C_{j,s} - p_{j,s} \geq 0,\ \forall j \in \Gamma,\ \forall s \in \Omega \tag{3}$$

$$D_{j,s} \geq C_{j,s},\ \forall s \in \Omega,\ \forall j \in \Gamma \tag{4}$$

$$C_{j,s} = D_{j,s-1} + p_{j,s},\ \forall s \in \Omega,\ \forall j \in \Gamma \tag{5}$$

$$z_{j_1,j_2,s} + z_{j_2,j_1,s} = 1,\ \forall s \in \Omega,\ \forall j_1, j_2 \in \Gamma,\ j_1, \neq j_2 \tag{6}$$

$$\begin{aligned} C_{j_2,s} \geq D_{j_1,s} + p_{j_2,s} + (y_{j_1,f,s,m} + y_{j_2,f,s,m} + z_{j_1,j_2,s} - 3) \cdot h, \\ \forall f \in \Lambda, \forall s \in \Omega,\ \forall m \in \{1, 2, \ldots, M_{f,s}\}, \\ \forall j_1, j_2 \in \Gamma,\ j_1, \neq j_2 \end{aligned} \tag{7}$$

$$\begin{aligned} E_{f,s,m} \geq D_{j,s} + (y_{j,f,s,m} - 1) \cdot h,\ \forall f \in \Lambda, \\ \forall s \in \Omega,\ \forall m \in \{1, 2, \ldots, M_{f,s}\},\ \forall j \in \Gamma \end{aligned} \tag{8}$$

$$\begin{aligned} IEC = \sum_{s=1}^{S} EC_s^{Idle} \cdot \\ \left( \sum_{f=1}^{F} \sum_{m=1}^{M_{f,s}} E_{f,s,m} - \sum_{j=1}^{J} p_{j,s} - \sum_{j=1}^{J} (D_{j,s} - C_{j,s}) \right) \end{aligned} \tag{9}$$

$$PEC = \sum_{s=1}^{S} \sum_{j=1}^{J} \left( EC_s^{Process} \cdot p_{j,s} \right) \tag{10}$$

$$BEC = \sum_{s=1}^{S} \sum_{j=1}^{J} \left( EC_s^{Blocking} \cdot (D_{j,s} - C_{j,s}) \right) \cdot \tag{11}$$

$$x_{j,f} \in \{0, 1\},\ \forall f \in \Lambda,\ \forall j \in \Gamma \tag{12}$$

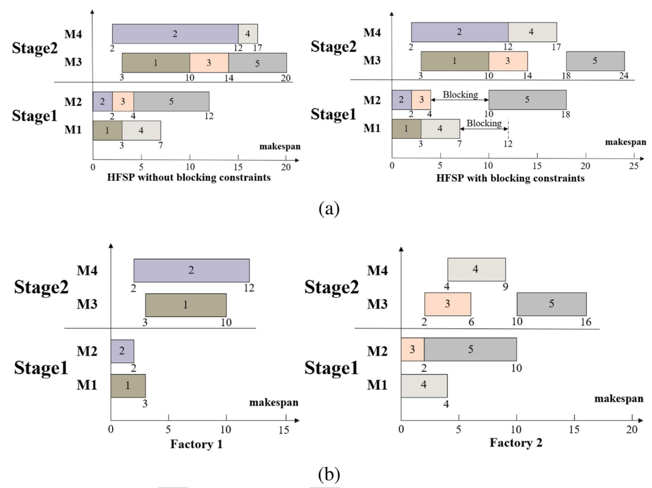



Fig. 1. (a) The Gantt diagram of HFSP with and without blocking constraints from the same factory. (b) The Gantt diagram of HFSP with blocking constraints in distributed environment.

$$\begin{aligned} y_{j,f,s,m} \in \{0, 1\},\ \forall f \in \Lambda,\ \forall s \in \Omega,\ \forall j \in \Gamma, \\ \forall m \in \{1, 2, \ldots, M_{f,s}\} \end{aligned} \tag{13}$$

$$z_{j_1,j_2,s} \in \{0, 1\},\ \forall s \in \Omega,\ \forall j_1, j_2 \in \Gamma \in \{0, 1\} \tag{14}$$

The objective is to minimize the total energy consumption of all machines. Eq. (1) ensures that each job can only be assigned to one factory for processing at most, and constraints (2) tell that each job can only be processed by one machine at each stage. Constraints (3) indicate that the completion time of each job should not be less than its processing time. Constraint set (4) ensures that due to the influence of blocking, the departure time of the job is greater than or equal to its completion time. Constraint set (5) defines that the completion time of a job in one stage is equal to the processing time in the same stage plus its departure time in the previous stage. Constraints (6) mean that there is only one sequence between jobs $j_1$ and $j_2$. Constraint set (7) represents that the completion time of one job is not less than its processing time in the same stage plus its departure time of its precursor. Constraint set (8) ensures that the shutdown time of a machine at one stage is not less than the departure time of the jobs on the same machine. Eq. (9) computes the energy consumption of the machines which are at idle states. Eq. (10) calculates the energy consumption of processing jobs. Eq. (11) computes the calculation of the blocking time of machines. Constraint set (12) ensures whether the job is assigned to the factory. Constraint set (13) indicates whether the job is assigned to a machine at a certain stage of a factory. Constraint set (14) makes sure whether there is a sequence of two jobs processed in the same stage.

## B. Example Instance

To further illustrate different conditions with and without blocking constraints in more detail, Fig. 1(a) shows the Gantt diagram of a simple example with five jobs and two stages in the same factory, each of which consists of two parallel machines in one stage. The horizontal axis is used to describe the completion time of jobs, and the vertical axis is used to indicate stages and machines. Relevant data are given as follows:

$$p_{j,s} = \begin{bmatrix} 3 & 7 \\ 2 & 10 \\ 2 & 4 \\ 4 & 5 \\ 8 & 6 \end{bmatrix} \quad EC_{f,s}^{Process} = [5 \quad 7]$$

$$EC_{f,s}^{Blocking} = [3 \quad 4] \;\; EC_{f,s}^{Idle} = [2 \quad 1]$$

To expound the energy consumption with and without blocking constraints, we take the case that is in Fig. 1(a) to elaborate processing status of the job sequence. Both the processing energy consumption of two cases are $19{\times}5{+}32{\times}7 = 319$. The idle energy consumption of the first case (without blocking constraints) is $5{\times}1{=}5$, where the idle and blocking energy consumption of the second case (with blocking constraints) is $9{\times}1{+}11{\times}3 = 42$. Therefore, the energy consumption of HFSP with and without blocking constraints are 361 and 324, respectively. It can be seen that due to the blocking constraints, these five jobs will waste 37 more energy consumption than the case without the same restrictions. To show that the distributed production environment can effectively reduce the energy waste, we allocate these five jobs to two factories for processing. As shown in Fig. 1(b), jobs 1 and 2 assigned to factory 1, and jobs 3, 4, 5 are assigned to factory 2. Blocking conditions are eliminated after the allocation of 5 jobs. Moreover, the energy consumption caused by idle states is $5{+}10 = 15$. Total energy consumption of scheduling in distributed environment is $319{+}15 = 334$, which consumes 27 less energy than HFSP with blocking constraints. The introduction of distributed scheduling mode can reduce the energy consumption caused by blocking constraints. Besides, with the increasing scale of jobs and stages, this kind of energy waste will also increase [24]. It shows the importance of a good scheduling strategy for the enterprise. Therefore, we can design highly efficient search strategies across and within factories to reduce blocking conditions.

## IV. Proposed Algorithm

This section describes the proposed QIG algorithm in detail and it is divided into three main parts, i.e., initialization strategy, global search strategy, and local search strategy. To make readers understand this algorithm more clearly, the framework of QIG is provided in Algorithm 1.

As shown in Algorithm 1, $NEH\_F(\pi)$ initialization strategy is used to allocate jobs to factories. The *GlobalSearchStrategy* operates the arrangement order of jobs in different factories. In the while loop, *SelectionMechanism* provides strategy selection guidance for all factories. In *SelectionMechanism*, each factory selects a certain local search strategy to execute, and the process is performed sequentially by factory number. When the operation of *SelectionMechanism* is finished, *GlobalSearchStrategy* for jobs, just as same as the strategy in initialization stage, will execute again. At the end of the while loop, the current best sequence is updated. If the termination condition is not reached, the procedure will continue to execute the while loop.

**Algorithm 1:** The Framework of The QIG Algorithm.

```
Require: π = {π_1, π_2, ..., π_J}, parameters used in this algorithm
Ensure: π^best and the corresponding energy consumption EC
Initialization:
π^temp = NEH_F(π)
GlobalSearchStrategy(π, π^temp)
Algorithm Body:
while the termination criterion is not satisfied do
   selectionMechanism(π^temp)
   GlobalSearchStrategy(π, π^temp)
   if f(π^temp) < f(π) then
      π = π^temp
      if f(π) < f(π^best) then
         π^best = π
      end if
   end if
end while
```

### A. Initialization Strategy

For the optimization of energy consumption in DHFSP with blocking constraints, the quality of the initial solution will have a direct impact on the later iteration. It can be seen from Algorithm 1 that QIG algorithm improves only one solution in the whole iterative process. Therefore, it is important to use a high-efficiency initialization strategy to sort the job sequence. Nawaz, Enscore, and Ham (NEH) [54] heuristic is an excellent heuristic algorithm that is often embedded into some metaheuristics. Huang and Pan et al. [55] used an algorithm, named *NEH_F*, which is based on NEH and characteristics of multiple factories to solve the allocation problem of jobs, and finally achieved good results. Thus, to get a better solution, this paper introduces the *NEH_F* method as the initialization strategy of QIG algorithm to allocate jobs to all factories. The processing steps of *NEH_F* are as follows. (1) At the beginning of the *NEH_F* strategy, a job sequence $= \pi = \{\pi_1, \pi_2, \ldots, \pi_J\}$ is arranged in descending order according to total processing time. (2) The first $F$ jobs are taken out from sequence and allocate them to $F$ factories one by one. (3) The remaining *n-F* jobs are extracted one by one and inserted into the best position of all factories. The details of *NEH_F* heuristic algorithm are presented in Algorithm 2.

### B. Global Search Strategy

Cross-factory operations using the insert strategy to allocate jobs usually spend a lot of time. In the process of scheduling for different factories, blocking constraints will directly influence the completion efficiency of processing task and energy consumption. Thus, to reduce the occurrence of blocking in

**Algorithm 2:** NEH_F Heuristic Algorithm.

**Require:** $\pi = \{\pi_1, \pi_2, \ldots, \pi_J\}$
**Ensure:** $\pi^{\text{temp}}$
Compute $\sum_{s=1}^{S} p_{j,s}, j = 1,2...,n$
$\pi^{\text{temp}} = Sort\_descend(\sum_{s=1}^{S} p_{j,s}), j = 1,2...,n$
**for** $j = 1 \rightarrow F$ **do**
  $f_j = \pi_j^{\text{temp}}$
**end for**
**for** $j = F+1 \rightarrow n$ **do**
  **for** $f = 1 \rightarrow F$ **do**
    Extract the job $\pi_j^{\text{temp}}$ from sequence $\pi^{\text{temp}}$ and insert into all the positions of factory $f$
    $Pos_j,\ EC_f$ % the best position and the minimum energy consumption
  **end for**
  $Pos_j^{best} = arg(min_{f=1}^{F} EC_f)$
  Insert $\pi_j^{\text{temp}}$ into the position $Pos_j^{best}$
**end for**

**Algorithm 3:** Global Search Strategy.

**Require:** $\pi$, $\pi^{\text{temp}}$, bool value $flag = false$
**Ensure:** $\pi^{\text{temp}}$
**Set parameter:** $f_{\max}$ %The factory that consumes the most energy
$EC_{old} = EC(\pi)$
Randomly select a factory $f_{random}$ that is different from the factory $f_{\max}$
**for** $j = 1 \rightarrow n$ **do**
  Randomly select a job from the factory $f_{random}$
  Randomly select a job from the factory $f_{\max}$
  Swap positions of the two jobs in factories $f_{\max}$ and $f_{random}$
  $EC_{new} = EC(\pi^{\text{temp}})$
  **if** $EC_{new} < EC_{old}$ **then**
    $flag = true$;
    $\pi = \pi^{\text{temp}}$
  **else**
    $\pi^{\text{temp}} = \pi$
  **end if**
**end for**

job sequence as much as possible and find the near-optimal solution more quickly and efficiently, we propose a global search strategy based on swap operator to sort jobs across factories. The proposed strategy can arrange a large number of jobs in a short time and can help the algorithm find a good solution. The steps for the global search strategy are as follows. (1) Based on the job sequence assigned by *NEH_F*, we first select the factory $f_{\max}$ with the highest energy consumption, and then randomly select another factory, denoted as $f_{random}$. (2) Randomly select one job from each of the two factories. (3) If the exchanged sequence is better than the original sequence, the original sequence is replaced. (4) After $n$ iterations, the procedure ends. The pseudocode of the global search strategy is shown in Algorithm 3.

### C. *The New Selection Mechanism*

In the iterative process, the processing environments of different factories are independent and closed to each other. A factory may reduce more energy consumption through strategies that are instructional in character. When the factory chooses an inappropriate strategy, the quality of the solution is difficult to improve and it is easy to fall into the local optimal state. Reinforcement learning (RL) is an important machine learning algorithm [56]. RL uses scalar reinforcement reward to interact with the complex environment [57], which maps the actions executed to the environment, and continuously learns new knowledge through the feedback to obtain the maximum cumulative return. As a free mode learning method, algorithms inspired by Q-learning idea [58] has been successfully applied to optimization problems in recent years [31], [59]. In related research, each factory completes the scheduling work independently. The processing experience is difficult to share with other factories, which leads to the isolation among factories. To help enterprise break this isolated phenomenon, this paper proposes a new selection mechanism inspired by Q-learning to select appropriate the local search strategy for each factory with blocking constraints. This selection mechanism can effectively solve the occlusion problem among factories and improve their production efficiency. The process of this selection mechanism is described as follows:

Refer to [59], we first set up a new type of 'Q-value table' to store the policy selection data of the factory. In the 'Q-value table', row represents different processing factories, column represents different local search strategies. The local search strategy of each column corresponding to all factories is the same. At the beginning of the iteration, strategies are randomly selected and all values in the 'Q-value table' are set as 1. As the iteration continues, the 'Q-value table' is gradually updated according to the sequence of row(factory) number, and if the Q-value in the corresponding column changes, it indicates that the strategy is implemented by the current factory. In the selection mechanism, an execution step is performed by selecting strategies according to the fitness value of each factory. In this paper, the fitness value is set as the reciprocal of energy consumption. Similar to the setting in reference [59], the Q-value update function is indicated as $Q(f_t, str_t) = (1-\alpha)Q(f_t, str_t) + \alpha(r_{t+1} + \gamma \underset{a}{max} Q(f_{t+1}, str_{t+1}))$. The $Q(f_t, str_t)$ value represents that factory $f_t$ carry out the strategy $str_t$, $\alpha$ indicates the learning rate, $\gamma$ indicates the discount rate, $r_{t+1}$ represents the reward value after carrying out the strategy, in the paper, it shows the difference between the new and old fitness values. $f_{t+1}$ and $str_{t+1}$ represent the next factory number and corresponding strategy. $Q(f_{t+1}, str_{t+1})$ represents that choose the maximum Q-value in factory $f_{t+1}$ using strategy $str_t$.

After the implementation of the corresponding strategy in all factories, fitness values $fitness_f$ of factory $f$ should be calculated according to the equation $fitness_f = 1/EC_f$. Then, factory numbers should be arranged in descending order according to their fitness values, so that the factory with high fitness value can be used as the guidance object for the previous factory to provide strategy selection. After the sorting of fitness

**Algorithm 4:** The New Selection Mechanism (Iteration == 1).

**Require:** $\pi^{\text{temp}}$, strategy $str \in \{0, 4\}$
$\pi^{\text{temp\_f}}$, % The sequence in factory $f$
$fitnessold_f, f = 1, 2, \ldots, F$, % The fitness value of each factory
**Ensure:** $\pi^{\text{temp}}$
**for** $f = 1 \rightarrow F$ **do**
  Randomly select the strategy *str* for the $\pi^{\text{temp\_f}}$
  $fitness_f$ () % Compute the fitness value of the factory $f$
**end for**
Arrange in descending order according to the fitness value of each factory, and new fitness is recorded as $fitnessnew_f$ **do**
Execute the equation $Q(f_t, str_t) = (1-\alpha)Q(f_t, str_t) + \alpha(r_{t+1} + \gamma \underset{a}{max} Q(f_{t+1}, str_{t+1}))$ to update the Q-value table
$fitnessold_f = fitnessnew_f$

**Algorithm 5:** The New Selection Mechanism (Iteration != 1).

**Require:** $\pi^{\text{temp}}$, strategy $str \in \{0, 4\}$
$\pi^{\text{temp\_f}}$, % The sequence in factory $f$
$fitnessold_f, f = 1, 2, \ldots, F$, % The fitness value of each factory
**Ensure:** $\pi^{\text{temp}}$
**for** $f = 1 \rightarrow F$
  $p = rand(), p \in \{0, 1\}$
  **if** $p < 0.5$ **then**
    $str = \underset{a}{max} Q(f_{t+1}, str_{t+1})$, $\pi^{\text{temp\_f}} = carryout(str)$
    $fiteness_f$ () % Compute the fitness value of the factory $f$
  **else**
    Randomly select a strategy for the $\pi^{\text{temp\_f}}$
    $fiteness_f$ () % Compute the fitness value of the factory $f$
  **end if**
**end for**
Arrange in descending order according to the fitness value of each factory, and new fitness is recorded as $fitnessnew_f$
Execute the equation $Q(f_t, str_t) = (1-\alpha)Q(f_t, str_t) + \alpha(r_{t+1} + \gamma \max_a Q(f_{t+1}, str_{t+1}))$ to update the Q-value table
$fitnessold_f = fitnessnew_f$

values, equation $Q(f_t, str_t) = (1-\alpha)Q(f_t, str_t) + \alpha(r_{t+1} + \gamma \underset{a}{max} Q(f_{t+1}, str_{t+1}))$ is used to calculate the Q-value of each factory, which $r_{t+1} = fitnessnew_f - fitnessold_f$, if the $r_{t+1}$ value is greater than 0, it means that using the strategy brings positive reward accumulation, otherwise, it has a negative accumulation. In addition, to prevent the accumulation of reward value for only one strategy, we set a larger probability parameter value, noted as p. From the second iteration, we first randomly generate a value of [0-1]. If the value is less than 0.5, the strategy with the largest reward value $\underset{a}{max} Q(f_{t+1}, str_{t+1})$ will be selected and executed. If not, a strategy will be randomly selected for execution. Through test results, we found that the method mentioned above can avoid the accumulation of reward value for only one strategy.

The pseudocode of the new selection mechanism is shown in Algorithms 4 and 5.

### D. Local Search Strategy

In the production scheduling workshop, the quality of the local solution also has a direct impact on the overall energy consumption. More efficient scheduling strategies can provide more possibilities for exploring broader search neighborhoods and finding better job sequences. To further improve the local search performance of the QIG algorithm, this paper presents five local search strategies to reduce the impact of blocking constraints on energy consumption. In the proposed local search strategies, there are four strategies based on swap operations, and two of these strategies are randomly are designed based on blocking constraints. The remaining two strategies are based on the randomness of the exchange of jobs. Finally, we introduce the destruction-reconstruction strategy of traditional IG algorithm as the fifth strategy. The proposed strategies can help the solution jump out of the local optimal and reduce the energy consumption of blocking by changing the order of the sequence. Moreover, the choice of strategy requires the use of the new selection mechanism proposed in the previous subsection C to improve the performance of these strategies as much as possible.

As can be seen from Algorithms 6, 7 and 8, strategies 1 and 2 (*str* 1 and 2 for short) are designed based on blocking jobs. In these two strategies, we first select two blocking jobs and exchange their positions, then we get sequence $\pi^{\text{temp\_f\_new}}$. If *str* 1 is executed and the new sequence $\pi^{\text{temp\_f\_new}}$ is better than the original sequence $\pi^{\text{temp\_f}}$, the original sequence $\pi^{\text{temp\_f}}$ is directly replaced by $\pi^{\text{temp\_f\_new}}$; If *str* 2 is executed and the new sequence $\pi^{\text{temp\_f\_new}}$ is better than the original sequence $\pi^{\text{temp\_f}}$, it continues to iterate on the basis of the original sequence $\pi^{\text{temp\_f}}$, at this time, $\pi^{\text{temp\_f}}$ is also replaced by $\pi^{\text{temp\_f\_new}}$. When implement *str* 3 and *str* 4, we first set the number of iterations as $|\pi^{\text{temp\_f}}|$. In these two strategies, first, two jobs with unequal positions are randomly selected for exchange. If execute the *str* 3 and a better sequence $\pi^{\text{temp\_f}'\text{\_interval}}$ is get, we update the new sequence by $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}}$, then continue to iterate until the end of the first level of the *for loop*. At the end of the loop, if $f(\pi^{\text{temp\_f}'\text{\_interval}}) < f(\pi^{\text{temp\_f}'})$, $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}'\text{\_interval}}$. If execute the *str* 4 and get a better sequence $\pi^{\text{temp\_f}'\text{\_new}}$, the $\pi^{\text{temp\_f}'}$ is directly replaced by $\pi^{\text{temp\_f}'\text{\_new}}$. *str* 5 uses destruction-construction strategy to change current solution. First, *d* random jobs are extracted from the original sequence $\pi^{\text{temp\_f}'}$ and inserted sequentially into all positions in the sequence $\pi^{\text{temp\_f}''}$ for testing, and finally,

**Algorithm 6:** The Local Search Strategy (str == 1 or str == 2).

**Require:** $\pi^{\text{temp\_f}}$, bool *flag* $=$ *true*, action *str*,
the number of the blocked jobs: $count_{block}$,
**Ensure:** $\pi^{\text{temp\_f}}$

**while** flag == false **do**
  **for** $count = 1 \rightarrow count_{block}$ **do**
    $\pi^{\text{temp\_f\_new}} = swap(\pi^{\text{temp\_f}})$ % Randomly swap two blocked jobs in the $\pi^{\text{temp\_f}}$
    **if** $f(\pi^{\text{temp\_f\_new}}) < f(\pi^{\text{temp\_f}})$ **then**
      **Case** *str* $==$ *1:* $\pi^{\text{temp\_f}} = \pi^{\text{temp\_f\_new}}$, *flag* $=$ *true*
      **Case** *str* $==$ *2:*
    $\pi^{\text{temp\_f\_interval}} = \pi^{\text{temp\_f\_new}}, \pi^{\text{temp\_f\_new}} = \pi^{\text{temp\_f}}, \pi^{\text{temp\_f}} = \pi^{\text{temp\_f\_interval}}, flag = true$
    **else**
      $\pi^{\text{temp\_f\_new}} = \pi^{\text{temp\_f}}$
    **end if**
  **end for**
**end while**

**Algorithm 7:** The Local Search Strategy (str == 3 or str == 4).

**Require:** $\pi^{\text{temp\_f}}$, bool *flag* $=$ *true*, action *str*,
the number of the blocked jobs: $count_{block}$,
**Ensure:** $\pi^{\text{temp\_f}}$

$\pi^{\text{temp\_f}'\text{\_interval}} = \pi^{\text{temp\_f}}$
**for** $j = 1 \rightarrow |\pi^{\text{temp\_f}}|$ **do**
  $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}}$
  **for** $i = 1 \rightarrow |\pi^{\text{temp\_f}}|$ **do**
    **if** $j \neq i$ **then**
      $\pi^{\text{temp\_f}'\text{\_new}} = swap(\pi_j^{\text{temp\_f}'}, \pi_i^{\text{temp\_f}'})$
    **end if**
    **if** $f(\pi^{\text{temp\_f}'\text{\_new}}) < f(\pi^{\text{temp\_f}'})$ **then**
      **Case** *str* $==$ *3:*
      **if** $f(\pi^{\text{temp\_f}'\text{\_new}}) < f(\pi^{\text{temp\_f}'\text{\_interval}})$ **then**
        $\pi^{\text{temp\_f}'\text{\_interval}} = \pi^{\text{temp\_f}'\text{\_new}}$, $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}}$
      **end if**
      **Case** *str* $==$ *4:* $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}'\text{\_new}}$
    **end if**
  **end for**
  **Case** *str* $==$ *3:*
  **if** $f(\pi^{\text{temp\_f}'\text{\_interval}}) < f(\pi^{\text{temp\_f}'})$ **then**
    $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}'\text{\_interval}}$
  **end if**
**end for**
**if** $f(\pi^{\text{temp\_f}'}) < f(\pi^{\text{temp\_f}})$ **then**
  $\pi^{\text{temp\_f}} = \pi^{\text{temp\_f}'}$
**end if**

**Algorithm 8:** The Local Search Strategy (str == 5).

**Require:** $\pi^{\text{temp\_f}}$, bool *flag* $=$ *true*, action *str*,
the number of the blocked jobs: $count_{block}$,
**Ensure** $\pi^{\text{temp\_f}}$

**Case** *str* $==$ *5:* $\pi^{\text{temp\_f}'} = \pi^{\text{temp\_f}}$
$U_{i=1}^{d} = extract(\pi^{\text{temp\_f}'})$
**for** $j = 1 \rightarrow d$ **do**
  $\pi^{\text{temp\_f}''} = \pi^{\text{temp\_f}'} \setminus U_j$
  $\pi^{\text{temp\_f}''} \xleftarrow[i=1\ to\ |\pi^{\text{temp\_f}'}|]{insert\ ith\ poisition} U_j$
  $\pi^{\text{temp\_f}'} = argmin_{i=1}^{|\pi^{\text{temp\_f}'}|} f(\pi^{\text{temp\_f}''})$
**end for**
**if** $f(\pi^{\text{temp\_f}'}) < f(\pi^{\text{temp\_f}})$ **then**
  $\pi^{\text{temp\_f}} = \pi^{\text{temp\_f}'}$
**end if**

we select the sequence with the smallest energy consumption value as the new sequence $\pi^{\text{temp\_f}'}$. If the energy consumption of $\pi^{\text{temp\_f}'}$ is better than $\pi^{\text{temp\_f}}$, $\pi^{\text{temp\_f}} = \pi^{\text{temp\_f}'}$.

## V. Experimental Results and Comparisons

### A. Experiment Settings

In this section, we evaluate the QIG algorithm for solving DHFSP with blocking constraints. To evaluate the performance of the new selection mechanism in this paper, we firstly compare the energy consumption with and without the selection mechanism. Then, we compare four existing classical intelligent optimization algorithms for solving related problems. Under the condition of the same running time, if the QIG can achieve the best results in most test cases, it can be proved that the proposed algorithm is effective to solve the DHFSP with blocking constraints.

Refer to reference [55], we set the test set in 90 different scale instances, and make $f \in \{2, 3, 4, 5, 6, 7\}$, $n \in \{50, 100, 150, 200, 300\}$ and $s \in \{5, 8, 10\}$. For each $f \times n \times s$ combination, 30 replicas are generated and tested. The processing time $p_{j,s}$ is uniformly distributed in the range of [1, 30]. The number of parallel machines at all stages in each factory equals two. The energy consumption per unit time for idle, blocking, and processing are from uniform distribution ranges [1, 2], [3, 4], and [5, 7], respectively.

In the experiments, all the algorithms are coded in C++ in Visual studio 2019 and all the instances are run on a Pentium processor with 2.60 GHZ, Intel Core i7, and a 16 GB RAM under the Windows 10 operating system. For the sake of the comparison in reference [55], the stopping condition for all algorithms are set to the identical execution time, i.e., $t = \omega \cdot n \cdot s$ milliseconds, as the termination condition. In the condition, $\omega$ is a predefined parameter value, it is a parameter that controls the length of running time. In this paper, two values are set for parameter $\omega$: 5, 10. $n$ is the number of the job, $s$ is the number of the stage. The performance measure is calculated by relative percentage increase (RPI) and the formula is shown in (15). The RPI is used to estimate the difference between the current value obtained and the best value.

$$RPI(i) = (c_i - c_{best})/c_{best} \times 100 \tag{15}$$

where $c_i$ is the average value of energy consumption obtained by the algorithm i, $c_{best}$ is the best energy consumption value

TABLE I
RESULTS FOR THE MILP MODEL

| | MILP | | | | | QIG | |
|---|---|---|---|---|---|---|---|
| **F_J_S** | **Time/s** | **Constraints** | **Gap** | **Lower bound** | **Energy Consumption** | **Time/s** | **Energy Consumption** |
| 2_8_2 | 3600 | 723 | 2.06% | 1381 | **1410** | 0.32 | 1442 |
| 2_12_2 | 3600 | 1563 | 2.79% | 2023 | **2081** | 0.48 | 2109 |
| 3_16_2 | 3600 | 3779 | 3.20% | 2665 | **2753** | 0.96 | 2783 |
| 3_20_2 | 3600 | 5843 | 5.31% | 3386 | 3576 | 1.2 | **3524** |
| 4_24_3 | 3600 | 18363 | 9.60% | 5907 | 6534 | 2.88 | **6514** |
| 4_28_3 | 3600 | 24895 | 12.64% | 6897 | 7872 | 3.36 | **7600** |
| 5_32_3 | 3600 | 39683 | 12.92% | 7918 | 9093 | 4.8 | **8722** |
| 5_36_3 | 3600 | 50115 | 15.96% | 8860 | 10542 | 5.4 | **9755** |
| 6_40_4 | 3600 | 103843 | / | 11801 | / | 9.6 | **15130** |
| 6_44_4 | 3600 | 125491 | / | 13006 | / | 10.56 | **16574** |
| 7_48_4 | 3600 | 172419 | / | 14219 | / | 13.44 | **18216** |
| 7_52_4 | 3600 | 202179 | / | 15464 | / | 14.56 | **19659** |

The bold entities represent the best values of all comparison algorithms.

that has been found in all of these compared algorithms. We first calculate RPI of each instance, and then compute the average values of RPI for all the instances. It is notable that the range of RPI value obtained by the different scale, respectively, has a little difference according to the simulation experimental results. Thus, in the following tables, "mean" value that is the average values of RPI of all the instances, and can be calculated to test the overall performance of algorithms. In addition, we give the best energy consumption of each algorithm in Tables II–V, where the best results of the algorithms are marked in bold.

## B. Verification of the MILP Model

In this section, we run 12 instances to verify the MILP model and the performance of QIG algorithm. The MILP model of the DHFSP with blocking constraints is solved by Gurobi in PyCharm software, using the python as the programming language to find a feasible solution. The running time is set as 3600 s. Table I summarizes the simulation results of 12 instances. In Table I, time represents the computation time cost of the instance. For each instance, the number of constraints is reported that indicates the complexity of the problem. The Energy consumption express the optimal value of the MILP and QIG algorithm. Symbol '/' means that the solution cannot be found within 3600 s, and black bold font indicates the best results. Gap=0 indicates that the optimal solution is found. The smaller the Gap value, the better solution is. However, for a minimization model, Gap is computed as (ObjVal-lower bound)/ObjVal, where ObjVal is the objective value for the current solution, and the lower bound is a bound of the best possible objective obtained by using branch-and-bound method of Gurobi. Thus, if the gap is not equal to 0, it does not mean that no optimal solution is found.

As can be seen from Table I, due to the complexity of DHFSP with blocking constraints and time limit, both MILP and QIG algorithm can not find the same value as lower bound values. Except for these 2_8_2, 2_12_2, and 3_16_2 instances, the MILP model obtains better solutions than QIG algorithm, in the following 9 instances of different scales, the QIG algorithm achieves better objective values in much less time. With the number of jobs, factories and stages increasing, the number of constraints in MILP model is also increasing dramatically. Similarly, the Gap value is also increasing. Through the experiment, we found that in the last four large scale examples, the MILP model could not find a feasible solution in 3600 s, while QIG could still give a feasible solution in a relatively short time, which means that when the problem size increases to a certain extent, the MILP model is not suitable for solving, which is also a major reason why we propose a meta-heuristic algorithm, i.e., the QIG algorithm, to solve the DHFSP with blocking constraints.

After experimental analysis, the reason why the QIG algorithm proposed in this paper can solve this problem may be: the idle and blocking state of machines become more, and the optimal solution is getting harder and harder to find. Moreover, all machines start at 0 time, when a job is assigned to a new machine, it is necessary to calculate the idle energy consumption of the machine from time 0 to process the first job. Therefore, the system will balance whether to vacate the machine to reduce the idle time of the machine. It is also the reason that greatly increase the computational complexity of the algorithm. From Table I, it is clear that the MILP model is able to find an optimal solution for small instances. However, when the scale becomes large, the model can not find the optimal solution in less time, or even can not find the feasible solution. Thus, we believe that the proposed QIG algorithm is more suitable than MILP for solving large-scale and complicated instances.

## C. Performance Analysis of the QIG Variants

To investigate the effectiveness of the global search strategy and proposed selection mechanism, we compare the situations that do not include these strategies. In the variant without the selection mechanism, all local search strategies are selected by random seeds. Similarly, the variant without the global search strategy indicates that remove the strategy both in initialization stage and iterative while loop. Among these algorithms, None-Selection mechanism (NO_S) represents the QIG algorithm without the new selection mechanism. None-Global search strategy (NO_G) represents the QIG algorithm without the global search strategy. In Tables II and III, the number of factories ($f$), jobs ($n$) and stages ($s$) are different, wherein the best values are marked in bold, respectively. For each instance, the termination criterion parameter $\omega$ is set to 10. All algorithms are repeatedly executed 30 times, and the best value is selected for comparison.

As can be seen from results in Table II, compared to the NO_S algorithm, QIG obtains 57 best results, NO_S obtains 33 best results. The number of best values obtained by QIG is nearly double than that of the NO_S algorithm. In the six mean sets, QIG gets 5 best results and NO_S gets only 1 best result. The reason may be that the new selection mechanism effectively solves the problem of experience occlusion between factories and help them choose the appropriate strategy to improve their production efficiency. Under the mechanism of sharing experience in different factories, the energy consumption value of the factory can be reduced. By observing results shown in Table III, QIG shows outstanding performance and outperforms the NO_G in all tests. The reason may be that the global search strategy can effectively explore the wide irregular range and promising neighborhood, and improve the search ability of the QIG algorithm. Energy consumption caused by blocking constraints is reduced.

According to above results and analysis, the main advantages of the proposed strategies are as follows:

1) The scheduling order of jobs in each factory is different, which will cause varying degrees of energy consumption. The proposed local search strategies can

TABLE II
RESULTS FOR THE ENERGY CONSUMPTION WITH AND WITHOUT SELECTION MECHANISM

| **Instance** | **n×s** | **QIG** | **NO_S** | | | **QIG** | **NO_S** |
|---|---|---|---|---|---|---|---|
| | 50×5 | **25087** | 25097 | | 50×5 | **24698** | 24748 |
| | 50×8 | **43888** | 44033 | | 50×8 | 47806 | **47792** |
| | 50×10 | **54536** | 54730 | | 50×10 | **59790** | 59905 |
| | 100×5 | 52840 | **52781** | | 100×5 | 52394 | **52058** |
| | 100×8 | 80546 | **80511** | | 100×8 | **91511** | 91676 |
| | 100×10 | **109810** | 109881 | | 100×10 | **108461** | 108739 |
| | 150×5 | **87076** | 87190 | | 150×5 | 74849 | **74726** |
| | 150×8 | **130880** | 131141 | | 150×8 | 123303 | **123259** |
| *f*=2 | 150×10 | **164648** | 164760 | *f*=5 | 150×10 | **159820** | 159914 |
| | 200×5 | **98267** | 98431 | | 200×5 | 104022 | **103846** |
| | 200×8 | 176215 | **175901** | | 200×8 | **170633** | 170916 |
| | 200×10 | **219773** | 220486 | | 200×10 | 210710 | **210534** |
| | 300×5 | **161974** | 162360 | | 300×5 | **154908** | 154952 |
| | 300×8 | 234025 | **233663** | | 300×8 | **259988** | 260637 |
| | 300×10 | 340497 | **339994** | | 300×10 | 296874 | **296083** |
| | **mean** | **132004.13** | 132063.93 | | **mean** | **129317.8** | 129319 |
| | 50×5 | 22594 | **22583** | | 50×5 | **29692** | 29756 |
| | 50×8 | **39356** | 39486 | | 50×8 | 44283 | **44257** |
| | 50×10 | **57007** | 57138 | | 50×10 | **59289** | 59593 |
| | 100×5 | 52711 | **52584** | | 100×5 | 54364 | **54331** |
| | 100×8 | **78140** | 78380 | | 100×8 | 87144 | **86994** |
| | 100×10 | **114560** | 114869 | | 100×10 | **114950** | 115141 |
| | 150×5 | 72109 | **71933** | | 150×5 | 88734 | **88661** |
| | 150×8 | 127465 | **127091** | | 150×8 | **123762** | 123990 |
| *f*=3 | 150×10 | **147251** | 147324 | *f*=6 | 150×10 | 156256 | **155896** |
| | 200×5 | **103406** | 103496 | | 200×5 | 95597 | **95515** |
| | 200×8 | 165556 | **165135** | | 200×8 | **165556** | 165646 |
| | 200×10 | **216027** | 216223 | | 200×10 | **213770** | 213915 |
| | 300×5 | **160533** | 160744 | | 300×5 | 146358 | **146265** |
| | 300×8 | **242968** | 243481 | | 300×8 | 260697 | **260350** |
| | 300×10 | **333630** | 333724 | | 300×10 | 319391 | **318973** |
| | **mean** | **128887.53** | 128946.07 | | **mean** | 130656.2 | **130618.87** |
| | 50×5 | **26652** | 26690 | | 50×5 | **27346** | 27348 |
| | 50×8 | **39411** | 39524 | | 50×8 | **50142** | 50185 |
| | 50×10 | **58341** | 58431 | | 50×10 | **60114** | 60472 |
| | 100×5 | **54282** | 54392 | | 100×5 | **51455** | 51694 |
| | 100×8 | 80954 | **80935** | | 100×8 | **89578** | 89640 |
| | 100×10 | **110059** | 110469 | | 100×10 | 108960 | **108958** |
| | 150×5 | **81159** | 81241 | | 150×5 | **71869** | 71876 |
| | 150×8 | **119406** | 119509 | | 150×8 | 136269 | **136082** |
| *f*=4 | 150×10 | **168550** | 168757 | *f*=7 | 150×10 | **170983** | 171203 |
| | 200×5 | **99103** | 99201 | | 200×5 | **111459** | 111551 |
| | 200×8 | 177409 | **177211** | | 200×8 | **173060** | 173536 |
| | 200×10 | 224228 | **224067** | | 200×10 | **214261** | 214550 |
| | 300×5 | **157040** | 157111 | | 300×5 | 133149 | **133101** |
| | 300×8 | **251453** | 251736 | | 300×8 | **254803** | 254835 |
| | 300×10 | 319878 | **319148** | | 300×10 | **335510** | 335616 |
| | **mean** | **131195** | 131228.13 | | **mean** | **132597.2** | 132709.8 |

The bold entities represent the best values of all comparison algorithms.

TABLE III
RESULTS FOR THE ENERGY CONSUMPTION WITH AND WITHOUT GLOBAL SEARCH STRATEGY

| **Instance** | **n×s** | **QIG** | **NO_G** | | | **QIG** | **NO_G** |
|---|---|---|---|---|---|---|---|
| | 50×5 | **25087** | 25545 | | 50×5 | **24726** | 25076 |
| | 50×8 | **38008** | 38215 | | 50×8 | **44756** | 45898 |
| | 50×10 | **53121** | 53430 | | 50×10 | **58743** | 59814 |
| | 100×5 | **46811** | 47093 | | 100×5 | **50579** | 51642 |
| | 100×8 | **83773** | 84282 | | 100×8 | **87821** | 89141 |
| | 100×10 | **107767** | 108959 | | 100×10 | **110134** | 112164 |
| | 150×5 | **81444** | 82021 | | 150×5 | **77872** | 78475 |
| | 150×8 | **128755** | 129369 | | 150×8 | **124359** | 126075 |
| *f*=2 | 150×10 | **166609** | 168012 | *f*=5 | 150×10 | **164659** | 166921 |
| | 200×5 | **104791** | 105449 | | 200×5 | **100308** | 101630 |
| | 200×8 | **173262** | 173628 | | 200×8 | **172713** | 174378 |
| | 200×10 | **206127** | 206561 | | 200×10 | **209126** | 211294 |
| | 300×5 | **166511** | 167120 | | 300×5 | **150081** | 151278 |
| | 300×8 | **249191** | 249280 | | 300×8 | **245829** | 247496 |
| | 300×10 | **335624** | 335753 | | 300×10 | **295431** | 297100 |
| | mean | **131125.4** | 131647.8 | | mean | **127809.13** | 129225.47 |
| | 50×5 | **28074** | 28418 | | 50×5 | **29670** | 30246 |
| | 50×8 | **41762** | 42669 | | 50×8 | **50181** | 51336 |
| | 50×10 | **55113** | 55927 | | 50×10 | **59866** | 61528 |
| | 100×5 | **49015** | 49689 | | 100×5 | **55779** | 56942 |
| | 100×8 | **92964** | 94139 | | 100×8 | **85674** | 87462 |
| | 100×10 | **119284** | 120888 | | 100×10 | **112597** | 114587 |
| | 150×5 | **72165** | 72743 | | 150×5 | **69064** | 70483 |
| | 150×8 | **129206** | 130520 | | 150×8 | **134731** | 136781 |
| *f*=3 | 150×10 | **163899** | 165996 | *f*=6 | 150×10 | **172427** | 174716 |
| | 200×5 | **101573** | 102327 | | 200×5 | **107406** | 108683 |
| | 200×8 | **152865** | 154273 | | 200×8 | **166054** | 167636 |
| | 200×10 | **195881** | 197075 | | 200×10 | **210306** | 213000 |
| | 300×5 | **163181** | 164271 | | 300×5 | **152446** | 153993 |
| | 300×8 | **261920** | 263272 | | 300×8 | **236520** | 237917 |
| | 300×10 | **324815** | 326073 | | 300×10 | **328695** | 330366 |
| | mean | **130114.47** | 131218.67 | | mean | **131427.73** | 133045.07 |
| | 50×5 | **28449** | 28941 | | 50×5 | **23935** | 24515 |
| | 50×8 | **43826** | 44758 | | 50×8 | **49590** | 50747 |
| | 50×10 | **61446** | 62441 | | 50×10 | **70725** | 72941 |
| | 100×5 | **51594** | 52141 | | 100×5 | **60171** | 61063 |
| | 100×8 | **89724** | 90785 | | 100×8 | **82583** | 84414 |
| | 100×10 | **110097** | 111980 | | 100×10 | **119147** | 121910 |
| | 150×5 | **77385** | 77874 | | 150×5 | **81839** | 83471 |
| | 150×8 | **122008** | 123449 | | 150×8 | **125359** | 127631 |
| *f*=4 | 150×10 | **172106** | 174746 | *f*=7 | 150×10 | **170619** | 173360 |
| | 200×5 | **91792** | 92754 | | 200×5 | **98028** | 99645 |
| | 200×8 | **174718** | 176771 | | 200×8 | **176540** | 178754 |
| | 200×10 | **218970** | 221119 | | 200×10 | **216192** | 218422 |
| | 300×5 | **145869** | 146637 | | 300×5 | **145602** | 146962 |
| | 300×8 | **236842** | 238357 | | 300×8 | **247366** | 249632 |
| | 300×10 | **314139** | 315962 | | 300×10 | **334557** | 337737 |
| | mean | **129264.33** | 130581 | | mean | **133483.53** | 135413.6 |

The bold entities represent the best values of all comparison algorithms.

reduce the blocking conditions of jobs by adjusting the sequence order appropriately with the selection mechanism. It can improve the performance of QIG algorithm in big search neighborhoods, and increase the quality of the solutions.

2) In different periods of the iteration, using a reasonable local search strategy will produce a good result. The new selection mechanism chooses the appropriate strategy at the right time with the probability selection, which can effectively improve the production efficiency of each factory, reducing the energy consumption caused by the blocking constraints.
3) Jobs in different factories will have a direct impact on the energy consumption of the enterprise. Consider using a strategy based on cross-factory swap operation of jobs, which facilitates faster access to promising solutions in a shorter period of time, greatly improving the efficiency and performance of algorithm execution.

### *D. Comparisons With the Presented Efficient Algorithms*

In this section, we compare energy consumption and RPI values of different algorithms. The parameter $\omega$ is set to 5 and 10, respectively. All algorithms run in the same termination condition. The specific settings have been illustrated in subsection A.

To verify the performance of the proposed algorithm, we compare the QIG algorithm to four different optimization algorithms, i.e., the CRO [26], the IG [43] for DPFSP, the DPSO [22] for HFSP, and the modeling and multi-neighborhood IG (MN-IG) [12] for DHFSP. These comparison algorithms have shown great performance in solving related problems. To show the performance of these comparison algorithms, experiment parameters are set according to the original literature. CRO generates many solutions during initialization. In each solution, the factory is assigned to the job. Among them, the initialization of one solution uses *NEH_F* heuristic, while the arrangement order of other solutions is generated randomly, but the position of job insertion is still based on the minimum energy consumption value in all positions. Other operations are carried out according to the original literature. IG and MN-IG algorithm also utilize the *NEH_F* heuristic to get an initial solution, and other operations are as same as the original paper. Due to the DPSO algorithm is used for solving the HFSP, there is no factory assigned strategy in the original literature. Thus, in this paper, we utilize some methods for assigning the jobs to factories in these initial solutions. For these solutions, one solution is generated by using the *NEH_F* heuristic. The other one is to first assign a number of jobs equal to the number of factories, then extract one remaining job at a time and place it at the end of the sequence in all factories to test which factory has the smallest energy consumption. Finally, we choose the location with the smallest objective value to

TABLE IV
ENERGY CONSUMPTION AND RPI VALUES FOR $T = 5$ *NS* MILLISECONDS

| | | CRO[26] | | IG[44] | | DPSO[22] | | MN_IG[12] | | QIG | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Instance** | **n×s** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** |
| | 50×5 | 27100† | 10.69 | 27533† | 9.79 | 26956† | 8.89 | 25549† | 1.84 | **25087** | **0.93** |
| | 50×8 | 42723† | 9.81 | 43982† | 9.77 | 42745† | 7.94 | 41261† | 2.97 | **40069** | **0.86** |
| | 50×10 | 55716† | 9.3 | 57190† | 10.28 | 55019† | 7.63 | 53279† | 2.74 | **51857** | **0.74** |
| | 100×5 | 51641† | 10.22 | 51876† | 8.54 | 52357† | 10.33 | 48997† | 2.52 | **47794** | **0.66** |
| | 100×8 | 83112† | 11.74 | 82462† | 9.24 | 83859† | 11.34 | 78883† | 3.84 | **75969** | **0.97** |
| | 100×10 | 110356† | 8.46 | 112158† | 8.5 | 112121† | 9.24 | 105348† | 1.68 | **103603** | **0.44** |
| | 150×5 | 72921† | 9.6 | 73951† | 9.31 | 74129† | 10.28 | 68443† | 1.17 | **67652** | **0.55** |
| | 150×8 | 137129† | 8.05 | 137864† | 7.72 | 138499† | 8.27 | 130100† | 1.32 | **128400** | **0.33** |
| $f$=2 | 150×10 | 172058† | 9.55 | 173599† | 9.21 | 173352† | 9.53 | 162788† | 2.3 | **159123** | **0.42** |
| | 200×5 | 110005† | 10.21 | 111236† | 9.07 | 114538† | 12.53 | 103707† | 1.44 | **102235** | **0.71** |
| | 200×8 | 179421† | 10.86 | 179960† | 9.12 | 184449† | 12.37 | 167646† | 1.56 | **165075** | **0.58** |
| | 200×10 | 228826† | 8.78 | 229563† | 8.29 | 230746† | 9.11 | 216563† | 1.75 | **212841** | **0.49** |
| | 300×5 | 149997† | 5.97 | 150811† | 6.53 | 154544† | 8.68 | 138828† | 1.96 | **137786** | **0.41** |
| | 300×8 | 274850† | 10.55 | 275281† | 9.45 | 281371† | 12.44 | 255763† | 0.76 | **253083** | **0.53** |
| | 300×10 | 349534† | 9.93 | 349628† | 9.29 | 356081† | 11.79 | 328185† | 1.06 | **324066** | **0.58** |
| | **mean** | 136359.27 | 9.58 | 137139.6 | 8.94 | 138717.73 | 10.02 | 128356 | 1.93 | **126309.33** | **0.61** |
| | 50×5 | 27928† | 11.51 | 29054† | 12.44 | 28194† | 10.39 | 27237† | 5.41 | **25839** | **0.89** |
| | 50×8 | 48324† | 10.15 | 49215† | 9.78 | 47607† | 8.02 | 46593† | 3.93 | **44830** | **0.92** |
| | 50×10 | 60528† | 7.04 | 60997† | 7.52 | 59520† | 6.49 | 58491† | 3.1 | **56733** | **0.53** |
| | 100×5 | 54413† | 9.21 | 55485† | 10.21 | 55566† | 10.72 | 52618† | 3.77 | **50705** | **0.78** |
| | 100×8 | 86648† | 11.01 | 88212† | 10.22 | 87226† | 10.19 | 82437† | 2.79 | **80196** | **0.83** |
| | 100×10 | 117431† | 10.27 | 118608† | 9.36 | 118904† | 10.28 | 111760† | 2.88 | **108627** | **0.72** |
| | 150×5 | 77246† | 9.59 | 77918† | 9.01 | 78882† | 11.24 | 73075† | 1.86 | **71738** | **0.63** |
| | 150×8 | 140602† | 9.36 | 143271† | 9.57 | 144698† | 11.25 | 135259† | 3.26 | **130983** | **0.52** |
| $f$=3 | 150×10 | 155841† | 9.5 | 157533† | 8.65 | 158501† | 10.23 | 148501† | 2.25 | **145227** | **0.61** |
| | 200×5 | 107876† | 12.11 | 108069† | 9.73 | 110870† | 12.36 | 101370† | 2.19 | **99202** | **0.65** |
| | 200×8 | 180457† | 8.87 | 181642† | 8.81 | 183945† | 10.67 | 170692† | 2.2 | **167011** | **0.51** |
| | 200×10 | 221613† | 8.63 | 221344† | 7.61 | 224939† | 9.73 | 209364† | 1.74 | **205790** | **0.69** |
| | 300×5 | 167474† | 8.89 | 168875† | 7.89 | 173542† | 10.35 | 155744† | 1.27 | **154313** | **0.63** |
| | 300×8 | 282564† | 10.59 | 283616† | 9.74 | 288299† | 12.92 | 266757† | 0.93 | **263788** | **0.59** |
| | 300×10 | 346640† | 8.58 | 350271† | 7.72 | 355977† | 9.74 | 328397† | 1.13 | **323833** | **0.5** |
| | **mean** | 138372.33 | 9.69 | 139607.33 | 9.22 | 141111.33 | 10.31 | 131219.67 | 2.58 | **128587.67** | **0.67** |
| | 50×5 | 31980† | 7.37 | 32661† | 7.99 | 31979† | 7.2 | 31661† | 4.69 | **30244** | **0.47** |
| | 50×8 | 44513† | 8.35 | 45561† | 8.91 | 44576† | 8.06 | 44316† | 5.49 | **42011** | **0.71** |
| | 50×10 | 67311† | 6.91 | 67595† | 6.74 | 66785† | 6.32 | 68408† | 7.33 | **63736** | **0.59** |
| | 100×5 | 60275† | 8.4 | 60432† | 7.44 | 61055† | 8.71 | 58136† | 2.75 | **56581** | **0.76** |
| | 100×8 | 95791† | 6.72 | 97485† | 7.83 | 97604† | 8.41 | 92597† | 1.97 | **90808** | **0.78** |
| | 100×10 | 115036† | 8.76 | 117663† | 9.37 | 117131† | 9.67 | 112716† | 4.29 | **108075** | **0.61** |
| | 150×5 | 83870† | 8.59 | 86268† | 8.89 | 87014† | 10.46 | 80727† | 1.9 | **79223** | **0.38** |
| | 150×8 | 135913† | 8.31 | 137944† | 9.4 | 139093† | 10.56 | 131160† | 3.43 | **126805** | **0.59** |
| $f$=4 | 150×10 | 166950† | 8.59 | 170011† | 9.35 | 171632† | 10.58 | 160798† | 2.97 | **156160** | **0.68** |
| | 200×5 | 99608† | 11.02 | 101804† | 10.86 | 104894† | 14.24 | 95149† | 3 | **92382** | **0.62** |
| | 200×8 | 189050† | 9.53 | 190811† | 9.02 | 193890† | 11.19 | 180788† | 3.02 | **175480** | **0.44** |
| | 200×10 | 214854† | 8.31 | 217370† | 8.38 | 219644† | 9.92 | 204588† | 1.78 | **201015** | **0.47** |
| | 300×5 | 152143† | 8.69 | 153029† | 8.36 | 156234† | 10.33 | 143484† | 1.41 | **141877** | **0.32** |
| | 300×8 | 271859† | 8.69 | 277508† | 8.23 | 282167† | 11.1 | 259766† | 1.13 | **254713** | **0.46** |
| | 300×10 | 354152† | 8.1 | 356392† | 9.28 | 360792† | 11.26 | 338278† | 1.98 | **331848** | **0.44** |
| | **mean** | 138887 | 8.42 | 140835.6 | 8.67 | 142299.33 | 9.87 | 133504.8 | 3.14 | **130063.87** | **0.55** |
| | 50×5 | 27561† | 8.49 | 28336† | 7.48 | 28161† | 8.14 | 27399† | 3.92 | **26365** | **0.39** |
| | 50×8 | 48226† | 8.7 | 50015† | 9.59 | 48568† | 8.21 | 47304† | 3.55 | **45684** | **0.85** |
| | 50×10 | 70542† | 8.26 | 71968† | 9.28 | 69879† | 7.72 | 71019† | 7.61 | **65996** | **0.8** |
| | 100×5 | 56528† | 7.86 | 57550† | 7.95 | 57819† | 9.36 | 55484† | 4.02 | **53339** | **0.47** |
| | 100×8 | 97270† | 9.18 | 98464† | 7.79 | 99106† | 9.3 | 94051† | 2.91 | **91392** | **0.52** |
| | 100×10 | 117345† | 8.25 | 119838† | 8.21 | 119202† | 8.47 | 112996† | 1.94 | **110846** | **0.73** |
| | 150×5 | 78236† | 9.23 | 79322† | 8.23 | 80939† | 10.67 | 76025† | 3.27 | **73616** | **0.7** |
| | 150×8 | 131441† | 7.42 | 132989† | 8.21 | 135066† | 10.53 | 127630† | 3.48 | **123338** | **0.49** |
| $f$=5 | 150×10 | 172767† | 6.97 | 175889† | 7.91 | 177075† | 9.19 | 167306† | 2.32 | **163517** | **0.51** |
| | 200×5 | 118354† | 10.09 | 119856† | 9.49 | 122029† | 11.76 | 113070† | 2.95 | **109825** | **0.56** |
| | 200×8 | 179340† | 6.93 | 182860† | 8.59 | 184775† | 10.01 | 172705† | 2.19 | **169003** | **0.37** |
| | 200×10 | 244889† | 6.94 | 247832† | 7.13 | 250673† | 8.77 | 236556† | 2.05 | **231808** | **0.31** |
| | 300×5 | 176939† | 7.73 | 178041† | 7.66 | 181252† | 9.54 | 167837† | 1.94 | **165786** | **0.46** |
| | 300×8 | 283581† | 7.98 | 286155† | 7.55 | 290460† | 9.65 | 274575† | 1.24 | **269038** | **0.47** |
| | 300×10 | 352199† | 6.98 | 355406† | 6.68 | 362079† | 8.68 | 337519† | 2.06 | **332180** | **0.47** |
| | **mean** | 143681.2 | 8.07 | 145634.73 | 8.12 | 147138.87 | 9.33 | 138765.07 | 3.03 | **135448.87** | **0.54** |
| | 50×5 | 30419† | 9.01 | 30320† | 7.41 | 30338† | 7.73 | 30618† | 7.57 | **28464** | **0.7** |
| | 50×8 | 47374† | 9.32 | 48594† | 9.44 | 47790† | 9.08 | 48300† | 8.42 | **44551** | **1.05** |
| | 50×10 | 70857† | 8.82 | 71728† | 8.16 | 70126† | 6.9 | 71048† | 7.14 | **66316** | **0.84** |
| | 100×5 | 52532† | 7.36 | 53752† | 7.44 | 54234† | 9.46 | 51852† | 3.26 | **50217** | **0.46** |
| | 100×8 | 97422† | 8.75 | 99972† | 8.63 | 99667† | 9.08 | 95632† | 3.58 | **92324** | **0.54** |
| | 100×10 | 118759† | 8.71 | 121851† | 9.61 | 120360† | 9.4 | 114349† | 2.86 | **111170** | **0.76** |
| | 150×5 | 83903† | 8.05 | 85534† | 8.13 | 86813† | 9.73 | 81306† | 2.09 | **79645** | **0.41** |
| | 150×8 | 141265† | 9.07 | 141972† | 8.56 | 144181† | 10.65 | 135756† | 3.25 | **131477** | **0.61** |
| $f$=6 | 150×10 | 171428† | 6.42 | 175129† | 7.54 | 175282† | 8.48 | 167433† | 2.77 | **162927** | **0.56** |
| | 200×5 | 103354† | 9.31 | 104970† | 8.95 | 106890† | 11.28 | 99110† | 2.59 | **96605** | **0.61** |
| | 200×8 | 185843† | 7.39 | 188038† | 6.59 | 191515† | 9.15 | 180848† | 2.51 | **176416** | **0.48** |
| | 200×10 | 225792† | 7.13 | 229274† | 7.59 | 230852† | 8.97 | 218728† | 2.43 | **213548** | **0.36** |
| | 300×5 | 164613† | 6.76 | 166665† | 7.14 | 170519† | 9.32 | 155668† | 1.61 | **153079** | **0.49** |
| | 300×8 | 251377† | 9.49 | 255909† | 9.26 | 260830† | 11.81 | 241602† | 1.69 | **235988** | **0.64** |
| | 300×10 | 354410† | 8.14 | 359972† | 8.56 | 366597† | 11.29 | 343445† | 2.38 | **335195** | **0.48** |
| | **mean** | 139956.53 | 8.25 | 142245.33 | 8.2 | 143732.93 | 9.49 | 135713 | 3.61 | **131861.47** | **0.6** |
| | 50×5 | 28313† | 8.65 | 29272† | 9.12 | 28446† | 8.03 | 29544† | 10.14 | **26825** | **0.68** |
| | 50×8 | 50216† | 7.27 | 51208† | 6.93 | 50499† | 6.98 | 51906† | 8.32 | **47918** | **0.67** |
| | 50×10 | 76431† | 7.35 | 76611† | 6.39 | 76000† | 6.71 | 76082† | 5.52 | **72103** | **1.01** |
| | 100×5 | 56199† | 6.38 | 57431† | 6.67 | 58618† | 9.38 | 56440† | 4.56 | **53978** | **0.55** |
| | 100×8 | 97381† | 8.57 | 97525† | 7.22 | 98688† | 9.01 | 94356† | 3.23 | **91402** | **0.89** |
| | 100×10 | 115538† | 6.14 | 117565† | 7.51 | 117186† | 8.28 | 115207† | 5.13 | **109590** | **0.57** |
| | 150×5 | 80945† | 8.43 | 82350† | 8.62 | 83789† | 10.85 | 78914† | 3.45 | **76281** | **0.46** |
| | 150×8 | 144936† | 7.69 | 147332† | 7.63 | 149417† | 9.87 | 140604† | 2.53 | **137133** | **0.62** |
| $f$=7 | 150×10 | 175689† | 7.7 | 177693† | 7.9 | 179346† | 9.35 | 168981† | 2.21 | **165326** | **0.58** |
| | 200×5 | 117711† | 8.64 | 120124† | 8.38 | 122109† | 10.45 | 113768† | 2.44 | **111061** | **0.41** |
| | 200×8 | 188753† | 6.28 | 191523† | 7.21 | 193504† | 8.93 | 182689† | 2.03 | **179047** | **0.51** |
| | 200×10 | 237210† | 7.22 | 239587† | 7.72 | 244778† | 10.47 | 228489† | 2.59 | **222728** | **0.66** |
| | 300×5 | 157016† | 6.8 | 159523† | 7.65 | 163563† | 9.96 | 149541† | 2.46 | **146285** | **0.4** |
| | 300×8 | 267384† | 8.05 | 273415† | 9.48 | 276953† | 12.46 | 256290† | 2.23 | **252433** | **0.51** |
| | 300×10 | 342313† | 7.45 | 349851† | 8.6 | 355649† | 10.28 | 331318† | 1.53 | **325427** | **0.39** |
| | **mean** | 142402.33 | 7.51 | 144734 | 7.8 | 146569.67 | 9.4 | 138275.27 | 3.89 | **134502.47** | **0.59** |

The bold entities represent the best values of all comparison algorithms.

TABLE V
ENERGY CONSUMPTION AND RPI VALUES FOR $T = 10$ *NS* MILLISECONDS

| | | CRO[26] | | IG[44] | | DPSO[22] | | MN_IG[12] | | QIG | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Instance** | **n×s** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** | **best** | **RPI** |
| | 50×5 | 27100† | 11.1 | 27533† | 9.79 | 26934† | 8.44 | 25549† | 1.84 | **25087** | **0.93** |
| | 50×8 | 47904† | 10.14 | 47895† | 7.6 | 47327† | 7.45 | 45596† | 2.43 | **44513** | **0.81** |
| | 50×10 | 55842† | 8.27 | 57766† | 10.06 | 55384† | 7.15 | 53628† | 2.18 | **52485** | **1.02** |
| | 100×5 | 56967† | 9.87 | 57452† | 8.23 | 57818† | 9.57 | 54579† | 2.68 | **53155** | **0.54** |
| | 100×8 | 96309† | 9.24 | 96362† | 7.82 | 96805† | 8.77 | 91086† | 1.66 | **89602** | **0.58** |
| | 100×10 | 115305† | 9.18 | 117472† | 8.88 | 116131† | 8.2 | 110338† | 2.21 | **107954** | **0.5** |
| | 150×5 | 82273† | 9.76 | 82817† | 8.53 | 84231† | 10.44 | 78362† | 2.19 | **76686** | **0.56** |
| | 150×8 | 143264† | 8.46 | 144900† | 8.69 | 144271† | 8.8 | 134908† | 1.19 | **133319** | **0.4** |
| $f$=2 | 150×10 | 172720† | 8.56 | 175027† | 8.7 | 176196† | 10.18 | 166392† | 3.34 | **161021** | **0.69** |
| | 200×5 | 103474† | 10.8 | 105276† | 10.73 | 105105† | 11.11 | 96604† | 1.61 | **95075** | **0.69** |
| | 200×8 | 190084† | 9.23 | 191690† | 8.49 | 194018† | 10.47 | 179387† | 1.52 | **176708** | **0.52** |
| | 200×10 | 227452† | 8.39 | 228427† | 7.68 | 229469† | 8.88 | 215652† | 1.65 | **212142** | **0.37** |
| | 300×5 | 168692† | 9 | 170234† | 8.24 | 175189† | 10.68 | 157573† | 0.96 | **155856** | **0.43** |
| | 300×8 | 275105† | 10.63 | 276250† | 9.25 | 281493† | 12.02 | 257231† | 1.43 | **253598** | **0.5** |
| | 300×10 | 346114† | 7.88 | 348896† | 7 | 354094† | 8.59 | 326359† | 1.57 | **321641** | **0.47** |
| | **mean** | 140573.67 | 9.51 | 141866.47 | 9.38 | 142964.33 | 12.86 | 132882.93 | 1.1 | **130589.47** | **0.44** |
| | 50×5 | 30315† | 10.27 | 30932† | 8.99 | 30244† | 11.62 | 29053† | 1.43 | **28452** | **0.55** |
| | 50×8 | 47355† | 8.81 | 48044† | 8.47 | 47548† | 10.49 | 46746† | 1.47 | **44949** | **0.37** |
| | 50×10 | 59866† | 9.39 | 60767† | 8.7 | 59111† | 9.76 | 57725† | 1.8 | **55828** | **0.57** |
| | 100×5 | 61952† | 8.61 | 62761† | 9.56 | 63069† | 7.25 | 60072† | 2.11 | **58636** | **0.88** |
| | 100×8 | 90861† | 8.23 | 92453† | 7.02 | 92332† | 7.47 | 87296† | 4 | **84888** | **0.61** |
| | 100×10 | 114935† | 8.43 | 116814† | 8.91 | 115954† | 7.52 | 109464† | 3.4 | **106968** | **0.74** |
| | 150×5 | 83802† | 9.15 | 84621† | 7.38 | 85893† | 8.74 | 80048† | 2.45 | **77949** | **0.57** |
| | 150×8 | 139326† | 9.07 | 140704† | 9.43 | 141858† | 9.53 | 132546† | 2.84 | **129419** | **0.76** |
| $f$=3 | 150×10 | 182628† | 9.61 | 184792† | 9.47 | 184852† | 9.64 | 174641† | 2.33 | **170914** | **0.63** |
| | 200×5 | 108381† | 9.64 | 109123† | 9.26 | 110628† | 10.92 | 101452† | 2.69 | **99958** | **0.57** |
| | 200×8 | 175798† | 10.02 | 178202† | 9.11 | 180046† | 10.25 | 166083† | 2.42 | **162226** | **0.47** |
| | 200×10 | 240530† | 8.57 | 243285† | 8.24 | 244822† | 8.86 | 229325† | 2.18 | **224099** | **0.56** |
| | 300×5 | 159096† | 10.5 | 159844† | 9.69 | 164568† | 11.32 | 151071† | 1.49 | **148832** | **0.52** |
| | 300×8 | 273324† | 10.49 | 277199† | 10.08 | 279993† | 12.08 | 258825† | 2.38 | **255444** | **0.59** |
| | 300×10 | 346952† | 8.83 | 352836† | 8.65 | 355125† | 9.92 | 328093† | 2.33 | **324485** | **0.44** |
| | **mean** | 141008.07 | 9.34 | 142825.13 | 8.03 | 143736.2 | 10.52 | 134162.67 | 1.78 | **131536.47** | **0.51** |
| | 50×5 | 27716† | 10.5 | 28115† | 8.9 | 27715† | 11.29 | 28213† | 1.79 | **26451** | **0.61** |
| | 50×8 | 52526† | 8.77 | 52565† | 7.67 | 51761† | 9.78 | 51160† | 1.77 | **48757** | **0.41** |
| | 50×10 | 67034† | 9.15 | 67178† | 7.86 | 66274† | 10.99 | 65118† | 1.5 | **62573** | **0.4** |
| | 100×5 | 51386† | 6.55 | 53028† | 7.65 | 52844† | 9.49 | 49895† | 2.28 | **48560** | **0.48** |
| | 100×8 | 89637† | 8.16 | 93585† | 8.7 | 92864† | 10.34 | 88348† | 1.32 | **85595** | **0.57** |
| | 100×10 | 112875† | 9.09 | 114990† | 8.64 | 114712† | 9.77 | 110617† | 2.28 | **107152** | **0.57** |
| | 150×5 | 82603† | 8.37 | 83163† | 7.27 | 84616† | 7.05 | 79825† | 6.66 | **77502** | **0.85** |
| | 150×8 | 141051† | 9.01 | 143029† | 8.34 | 143992† | 7.41 | 135155† | 4.93 | **130882** | **0.88** |
| $f$=4 | 150×10 | 190064† | 8.66 | 192768† | 7.6 | 192463† | 7.06 | 181841† | 4.07 | **177795** | **0.68** |
| | 200×5 | 108022† | 7.76 | 109908† | 9.35 | 112003† | 9.85 | 103746† | 2.75 | **102018** | **0.65** |
| | 200×8 | 187361† | 7.16 | 188692† | 9.54 | 192623† | 9.27 | 179221† | 3.22 | **175771** | **0.5** |
| | 200×10 | 229538† | 6.65 | 233131† | 7.42 | 236079† | 7.73 | 220806† | 3.23 | **216126** | **0.73** |
| | 300×5 | 182590† | 8.23 | 185142† | 7.51 | 189360† | 9.97 | 173490† | 3 | **170665** | **0.56** |
| | 300×8 | 287682† | 8.72 | 290146† | 9.76 | 296315† | 10.91 | 275278† | 3.26 | **270237** | **0.67** |
| | 300×10 | 334820† | 8.11 | 340687† | 8.42 | 346698† | 9.06 | 318926† | 2.28 | **312606** | **0.42** |
| | **mean** | 142993.67 | 7.89 | 145075.13 | 8.12 | 146687.93 | 10.09 | 137442.6 | 1.69 | **134179.33** | **0.33** |
| | 50×5 | 25563† | 8.69 | 25942† | 7.79 | 25218† | 10.37 | 24914† | 1.96 | **23584** | **0.5** |
| | 50×8 | 47575† | 7.34 | 48926† | 8.35 | 47337† | 9.9 | 46668† | 2.17 | **45012** | **0.47** |
| | 50×10 | 67183† | 9.43 | 67809† | 7.91 | 66431† | 10.61 | 68393† | 1.39 | **62855** | **0.56** |
| | 100×5 | 53312† | 8.07 | 53707† | 9.02 | 53862† | 10.45 | 51140† | 1.97 | **49819** | **0.5** |
| | 100×8 | 92039† | 6.88 | 94437† | 7.84 | 92898† | 10.66 | 88567† | 2.37 | **86266** | **0.44** |
| | 100×10 | 114817† | 8.28 | 116081† | 8.95 | 116278† | 9.95 | 111094† | 1.11 | **108079** | **0.43** |
| | 150×5 | 85417† | 8.67 | 86291† | 8.67 | 87504† | 11.42 | 82295† | 1.66 | **79466** | **0.51** |
| | 150×8 | 146370† | 7.23 | 147997† | 7.59 | 149581† | 9.99 | 141357† | 1.87 | **138078** | **0.29** |
| $f$=5 | 150×10 | 176220† | 8.06 | 178573† | 8.3 | 179977† | 9.54 | 170678† | 2.75 | **165762** | **0.55** |
| | 200×5 | 118627† | 10.11 | 120235† | 10 | 121812† | 8.94 | 113847† | 5.64 | **111597** | **0.89** |
| | 200×8 | 185895† | 7.49 | 188538† | 8.72 | 192359† | 6.65 | 179535† | 3.68 | **176734** | **0.6** |
| | 200×10 | 231677† | 8.44 | 234019† | 7.96 | 237160† | 7.06 | 222228† | 8.81 | **217456** | **0.49** |
| | 300×5 | 152631† | 9.3 | 154398† | 8.35 | 158264† | 9.23 | 146857† | 2.65 | **144635** | **0.59** |
| | 300×8 | 276720† | 9.09 | 279348† | 9.47 | 283625† | 8.58 | 265110† | 2.67 | **261212** | **0.78** |
| | 300×10 | 357725† | 8.22 | 361275† | 7.76 | 367829† | 8.7 | 341701† | 2.79 | **335676** | **0.46** |
| | **mean** | 142118.07 | 10.75 | 143838.4 | 8.72 | 145342.33 | 11.66 | 136958.93 | 3.56 | **133748.73** | **0.61** |
| | 50×5 | 26728† | 7.8 | 27810† | 7.28 | 27270† | 9.06 | 26641† | 2.37 | **25586** | **0.36** |
| | 50×8 | 55130† | 7.52 | 55343† | 8.18 | 54869† | 9.45 | 56574† | 2.97 | **51858** | **0.39** |
| | 50×10 | 66038† | 8.61 | 67007† | 7.97 | 65791† | 9.64 | 66637† | 2.02 | **62391** | **0.47** |
| | 100×5 | 50572† | 6.86 | 51192† | 6.84 | 51214† | 9.28 | 48489† | 1.58 | **46864** | **0.29** |
| | 100×8 | 95755† | 7.97 | 97302† | 7.92 | 97405† | 9.59 | 92262† | 2.19 | **89565** | **0.74** |
| | 100×10 | 113625† | 9.47 | 115588† | 10.65 | 115406† | 13.55 | 109479† | 2.41 | **107144** | **0.53** |
| | 150×5 | 77278† | 8.9 | 78953† | 9.88 | 79377† | 11.13 | 74962† | 2.38 | **72700** | **0.59** |
| | 150×8 | 136829† | 7.55 | 140219† | 8.39 | 141766† | 10.45 | 134096† | 3.33 | **129847** | **0.6** |
| $f$=6 | 150×10 | 170048† | 8.32 | 172634† | 9.65 | 172818† | 11.34 | 162151† | 2.02 | **158121** | **0.51** |
| | 200×5 | 100418† | 7.23 | 103199† | 6.9 | 105863† | 9.79 | 97338† | 1.54 | **94444** | **0.24** |
| | 200×8 | 184442† | 6.86 | 187637† | 7.16 | 189422† | 9.19 | 179076† | 1.49 | **173638** | **0.3** |
| | 200×10 | 241047† | 8.36 | 243807† | 8.43 | 247277† | 9.63 | 233372† | 3.01 | **226709** | **0.52** |
| | 300×5 | 157202† | 7.24 | 158925† | 8.69 | 164678† | 8.25 | 151368† | 4.12 | **148006** | **0.65** |
| | 300×8 | 283146† | 7.85 | 286314† | 6.84 | 291269† | 7.04 | 271646† | 9.09 | **266053** | **0.71** |
| | 300×10 | 321769† | 7.62 | 326988† | 7.6 | 332790† | 6.34 | 309580† | 6.81 | **302694** | **0.7** |
| | **mean** | 138668.47 | 11.1 | 140861.2 | 9.6 | 142481 | 11.06 | 134244.73 | 3.47 | **130374.67** | **0.53** |
| | 50×5 | 28313† | 9 | 29272† | 8.65 | 28446† | 9.91 | 29544† | 3.01 | **26797** | **0.69** |
| | 50×8 | 56109† | 8.66 | 56581† | 8.08 | 56231† | 8.97 | 57645† | 2.18 | **53217** | **0.37** |
| | 50×10 | 67547† | 7.89 | 68361† | 8.77 | 67297† | 10.16 | 67714† | 3.11 | **64234** | **0.7** |
| | 100×5 | 55346† | 7.45 | 56325† | 8.35 | 56175† | 9.96 | 54619† | 3.27 | **51977** | **0.39** |
| | 100×8 | 93213† | 8.78 | 95370† | 9.21 | 95154† | 10.16 | 95699† | 2.55 | **87966** | **0.41** |
| | 100×10 | 113906† | 9.17 | 114304† | 9.58 | 115688† | 12.81 | 115660† | 3.06 | **107426** | **0.67** |
| | 150×5 | 77580† | 7.44 | 79021† | 8.06 | 79940† | 10.27 | 75378† | 3.13 | **73079** | **0.44** |
| | 150×8 | 133152† | 7.54 | 135253† | 7.72 | 137414† | 9.6 | 133090† | 2.94 | **125631** | **0.45** |
| $f$=7 | 150×10 | 179962† | 7.66 | 182288† | 8.66 | 183899† | 11.14 | 174818† | 2.35 | **170238** | **0.37** |
| | 200×5 | 112389† | 6.25 | 114897† | 7.59 | 116594† | 8.91 | 108688† | 1.86 | **104857** | **0.32** |
| | 200×8 | 179944† | 7.43 | 181926† | 8.37 | 186803† | 10.28 | 175384† | 2.84 | **169640** | **0.29** |
| | 200×10 | 231688† | 6.83 | 234810† | 7.86 | 237572† | 10.22 | 223440† | 1.79 | **218539** | **0.46** |
| | 300×5 | 152366† | 6.91 | 155242† | 7.73 | 158560† | 11.75 | 146445† | 2.27 | **143159** | **0.38** |
| | 300×8 | 261959† | 7.11 | 265861† | 8.01 | 272575† | 9.87 | 252167† | 2.1 | **247557** | **0.48** |
| | 300×10 | 358833† | 7.89 | 364805† | 8.3 | 369988† | 9.82 | 347179† | 3.33 | **339454** | **0.5** |
| | **mean** | 140153.8 | 8.95 | 142287.73 | 9.24 | 144155.73 | 7.87 | 137164.67 | 10.25 | **132251.4** | **0.5** |

The bold entities represent the best values of all comparison algorithms.

insert. This step stops until all jobs are assigned to factories. The rest solutions are produced by replacing the descending order in *NEH_F* with ascending order. and the other steps are the same as the original *NEH_F*. After the initialization is completed, the remaining operations are performed according to the original literature. For the proposed QIG algorithm in this paper, through simulation tests, we find that the setting value of parameters in these sub strategies has little effect on the final results, and there is no significant difference between solutions. Finally, after the consideration based on results, we set the parameter learning rate $\alpha = 0.5$, the discount rate $\gamma = 0.8$, and the number of destruction jobs $d = 3$ [32].

In addition, to evaluate different degrees of results between two algorithms in statistics, we perform Wilcoxon rank-sum tests with the significance level of 0.05 to examine whether there is a significant difference between the comparison algorithm and the QIG. In Tables IV and V, the symbol '†' represents that best result whether is significantly different from the QIG algorithm. If there is no symbol identification between these two algorithms, the difference is not significant.

It can be seen from formula (12), $c_i$ is the value for the current algorithm, $c_{best}$ is obtained from the best result of all algorithms. According to Tables IV, and V, QIG algorithm achieves the best value in each scale, and its RPI values are all minimum in all repeated experiments. In addition, for results of Wilcoxon rank-sum tests with the 0.05 significance level, the significance level of any algorithm compared to the QIG is far less than 0.05, indicating that comparison algorithms are significantly different from the QIG algorithm. According to

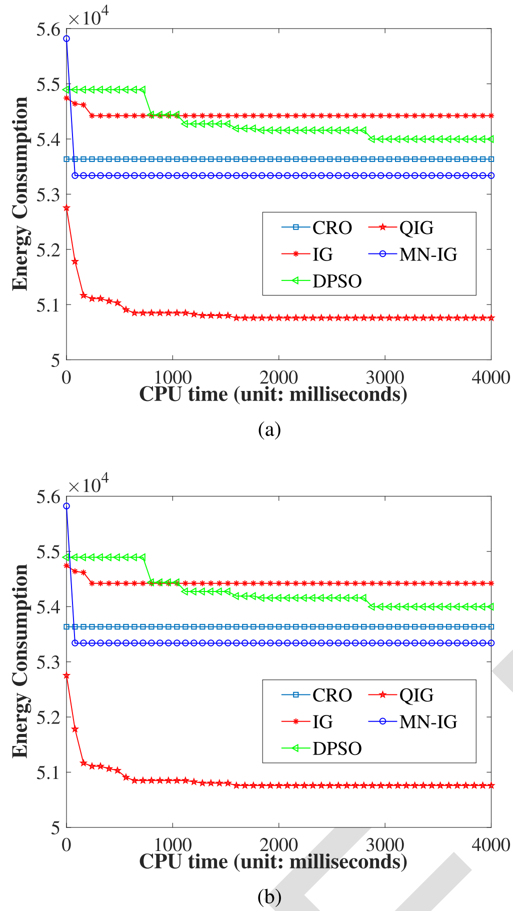



Fig. 2. The convergence curves of compared algorithms.

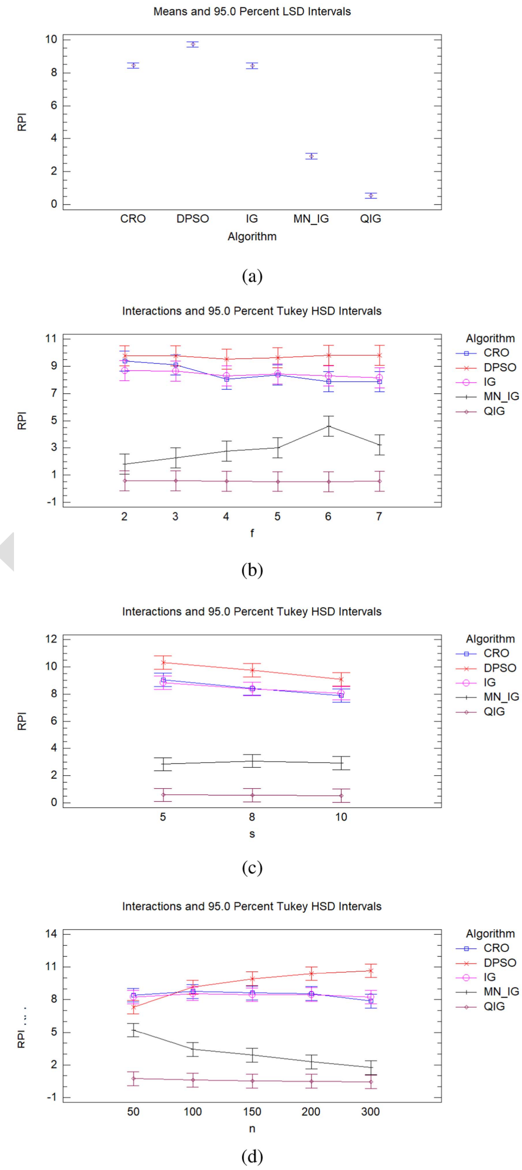



Fig. 3. Interactions for CRO, DPSO, IG, MIN_IG, and QIG. (a) Interactions of all the compared algorithms. (b) Interactions of algorithms and factory. (c) Interactions of algorithms and stage. (d) Interactions of algorithms and the number of jobs.

simulation results, the QIG algorithm substantially outperforms comparison algorithms.

These results mentioned above show advantages of the QIG algorithm, the reasons are as follows:

1) The proposed strategy improves the diversity of the algorithm while maintaining the local search ability of the original IG algorithm, and the reordering of jobs can greatly reduce blocking conditions.

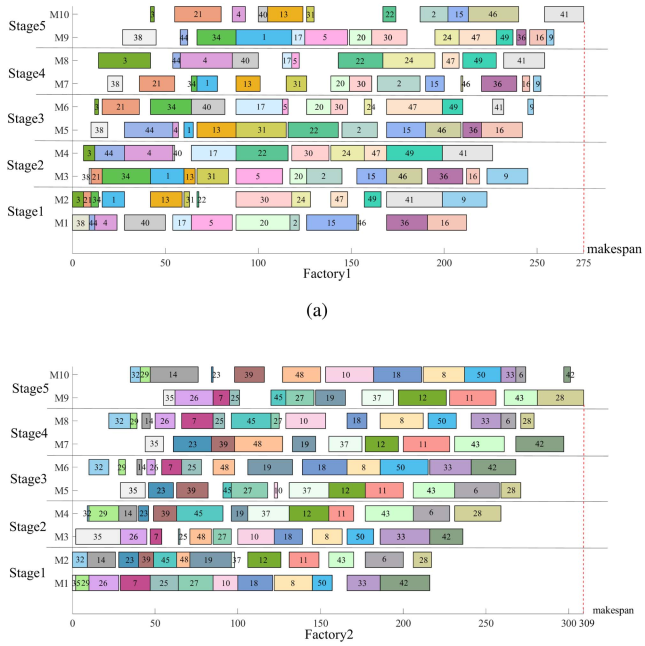
2) The proposed global search strategy greatly improves the quality of the solution by exchanging jobs across factories quickly, and helps find a better solution in the big search neighborhood, so as to reduce the energy consumption.

3) The new selection mechanism embedded in the IG algorithm helps the factory choose the appropriate strategy at a reasonable time. It enables experience sharing and interaction among factories and reduces inappropriate policy choices.

### *E. Convergence Curves and Confidence Intervals*

In this section, we further evaluate the performance of algorithms. To give the convergence graphic display of all comparison algorithms, we select two representative examples, where scales $(f \times n \times s)$ are $5\times50\times8$, $2\times100\times10$, respectively. As shown in Figs. 2(a) and (b), convergence curves of these algorithms use different colors and symbols, ordinate represents the energy consumption value of the job sequence, and abscissa represents the execution time of the algorithm (unit: milliseconds). These two examples represent changes in the convergence performance of algorithms respectively when the problem scale is continuously expanded.

To have a clear identification of results, we give the ANOVA analysis of all algorithms. As shown in Fig. 3(a)–(d), mean plots and interactions plots with 95% HSD intervals represent the average level and overall performance of algorithms. HSD is a method that can compare the average values of each pair. LSD uses t-test to perform all pairwise comparisons between group means. It verifies that there is a significant difference between two values. Subfigures 3(a), 3(b), 3(c), and 3(d) represent types of RPI, factory, stage and job number, respectively. RPI means the gap between different algorithms and the best value at each scale. All algorithms are executed when $\omega = 10$.



(a)

(b)

Fig. 4. Gantt charts of the QIG algorithm.

As can be seen from Fig. 2(a) and (b), the initial solution of QIG algorithm is better than other algorithms, because the global search strategy greatly improves the quality of the initial solution by swapping jobs across factories. In the whole iteration process, some algorithms, such as CRO, IG and MN-IG, which are easy to fall into local optimal, while the local search strategy selected by proposed selection mechanism improves the performance of the QIG algorithm. The QIG algorithm combines with the global search strategy, to a certain extent, it prevents the solution from falling into local optimum. In Fig 3(a), the difference of RPI value obtained by QIG algorithm between the best value and the worst value is very small, and it shows that the QIG is the best one among all algorithms. From Fig. 3(b)–(d), we can further see that the stability and performance of the proposed algorithm is superior than other comparison algorithms, then MN-IG, IG, CRO, DPSO follows with the good performance. Obviously, IG series of algorithms are better for solving the DHFSP with blocking constraints, and QIG algorithm achieve best results among them. All in all, through four test instance subgraphs, we can intuitively see that the proposed QIG gets the best energy consumption value of job sequence.

### *F. Gantt Charts of the QIG Algorithm*

To intuitively observe the processing sequence and blocking status of jobs in different factories, we provide the Gantt charts of the 2×50×5 $(f \times n \times s)$ instance. The advantage of drawing Gantt chart is that it can provide the optimal scheduling scheme for the factory managers and help them make the right decisions. Fig. 4(a), and (b) show Gantt charts of two identical factories when $\omega = 10$, respectively. In these Gantt charts, the abscissa represents the completion time of jobs, and the ordinate represents different machine numbers at different stages. Each job has unique color and number. Through the experimental test, the minimum energy consumption in this example is 25087. The order of the processing jobs in factory 1 is 38-3-21-44-34-4-1-40-13-17-31-5-22-20-30-2-24-15-47-46-49-36-41-16-9. The order of the processing jobs in factory 2 is 35-32-29-14-26-23-7-39-25-45-48-27-19-10-37-18-12-8-11-50-43-33-6-42-28. The completion times of the two factories are 275 and 309 respectively.

## VI. Conclusion

In this paper, we design an effective QIG algorithm to solve the DHFSP with blocking constraints with minimizing the energy consumption. This work contributes to the scheduling and allocation of the distributed hybrid flow shop. To solve the DHFSP with blocking constraints, we proposed an improved QIG algorithm. From extensive simulation tests, it can be seen that the QIG algorithm is superior to other compared algorithms in solution quality and search ability. The outperformance of the QIG algorithm is mainly attributed to the following aspects: 1) The proposed global search strategy helps the algorithm to generate a good initial solution, so that it has a greater probability to find the near-optimal solution than other algorithms in the iterative process. Moreover, the strategy improves the global search ability of the algorithm, and prevents the solution from falling into the local optimum: 2) A new selection mechanism inspired by Q-learning is embedded into IG algorithm to help factories make a reasonable strategy choice at the certain moment. It helps the enterprise break the closed states of each factory: 3) All five local search strategies are designed for blocking constraints in a single factory, and the energy consumption is reduced by continuously rearranging jobs. It demonstrates the effectiveness of the proposed strategies to solve the DHFSP with blocking constraints.

In future research, the proposed QIG algorithm can be further explored to solve other types of the DHFSP with various constraints, such as lot-streaming, setup time, assembly, and some uncertain scheduling problems. Besides, for the DHFSP, we will expand the optimization goal from single to multiple, such as the maximum completion time. However, the implementation of this algorithm is somewhat complicated, such as the combination of selection mechanism and local search strategy, which will be further optimized later. Next, we will redesign the appropriate strategies to solve the problem according to characteristics of the problem. It is also meaningful to integrate the intelligent method into the strategy to realize a self-learning mode.

## References

[1] J. Q. Li, H. Y. Sang, Y. Y. Han, C. G. Wang, and K. Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Cleaner Prod.*, vol. 181, pp. 584–598, 2018.

[2] J. Q. Li, Y. Q. Han, P. Y. Duan, and Y. Y. e. Han, "Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems," *J. Cleaner Prod.*, vol. 250, 2020, Art. no. 119464.

[3] L. Meng, C. Zhang, X. Shao, and Y. Ren, "MILP models for energy-aware flexible job shop scheduling problem," *J. Cleaner Prod.*, vol. 210, pp. 710–723, 2019.

[4] X. Gong et al., "Integrating labor awareness to energy-efficient production scheduling under real-time electricity pricing: An empirical study," *J. Cleaner Prod.*, vol. 168, pp. 239–253, 2017.

[5] X. Li and M. Li, "Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem," *IEEE Trans. Eng. Manage.*, vol. 62, no. 4, pp. 544–557, Nov. 2015.

[6] C. Yu, P. Andreotti, and Q. Semeraro, "Multi-objective scheduling in hybrid flow shop: Evolutionary algorithms using multi-decoding framework," *Comput. Ind. Eng.*, vol. 147, 2020, Art. no. 106570.

[7] F. Shrouf, J. Ordieres-Mere, A. Garcia-Sanchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *J. Cleaner Prod.*, vol. 67, no. 6, pp. 197–207, 2014.

[8] D. Giglio, M. Paolucci, and A. Roshani, "Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems," *J. Cleaner Prod.*, vol. 148, pp. 624–641, 2017.

[9] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," *Comput. Operations Res.*, vol. 100, no. 9, pp. 211–229, 2018.

[10] B. Zhang, Q. -K. Pan, L. Gao, L. -L. Meng, X. -Y. Li, and K.-K. Peng, "A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 12, pp. 4984–4999, Dec. 2020.

[11] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking," *Omega*, vol. 39, no. 3, pp. 293–301, 2011.

[12] W. Shao, Z. Shao, and D. Pi, "Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem," *Knowl.-Based Syst.*, vol. 194, pp. 1–17, 2020.

[13] J. -J. Wang and L. Wang, "A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 6, pp. 947–961, Dec. 2021.

[14] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Comput. Ind. Eng.*, vol. 142, 2020, Art. no. 106347.

[15] L. Meng, K. Gao, Y. Ren, B. Zhang, H. Sang, and Z. Chaoyong, "Novel milp and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times," *Swarm Evol. Comput.*, vol. 71, 2022, Art. no. 101058.

[16] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Operations Res.*, vol. 37, no. 4, pp. 754–768, 2010.

[17] S. Hatami, R. Ruiz, and C. Andrés-Romano, "The distributed assembly permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5292–5308, 2013.

[18] S. W. Lin and K. C. Ying, "Minimizing makespan for solving the distributed no-wait flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 99, no. 9, pp. 202–209, 2016.

[19] W. Shao, D. Pi, and Z. Shao, "A pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 3, pp. 1344–1360, Jul. 2019.

[20] Q. K. Pan, L. Gao, L. Wang, J. Liang, and X. Y. Li, "Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem," *Expert Syst. Appl.*, vol. 124, no. 6, pp. 309–324, 2019.

[21] M. Nejati, I. Mahdavi, R. Hassanzadeh, N. Mahdavi-Amiri, and M. S. Mojarad, "Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint," *Int. J. Adv. Manuf. Technol.*, vol. 70, no. 1–4, pp. 501–514, 2014.

[22] M. K. Marichelvam, M. Geetha, and M. Tosun, "An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors–a case study," *Comput. Operations Res.*, vol. 114, pp. 1–9, 2019.

[23] Q. K. Zhang, B. Pan, L. Gao, X. L. Zhang, H. Y. Sang, and J. Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," *Appl. Soft Comput.*, vol. 52, pp. 14–27, 2017.

[24] H. X. Qinet al.,, "An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem," *Swarm Evol. Computation*, vol. 69, 2022, Art. no. 100992.

[25] S. Aqil and K. Allali, "Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem," *Eng. Appl. Artif. Intell.*, vol. 100, 2021, Art. no. 104196.

[26] H. Bargaoui, O. B. Driss, and K. Ghédira, "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion," *Comput. Ind. Eng.*, vol. 111, no. 9, pp. 239–250, 2017.

[27] S. Y. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 46, no. 1, pp. 139–149, Jan. 2016.

[28] Q. Chen, J. Ding, T. Chai, and Q. Pan, "Evolutionary optimization under uncertainty: The strategies to handle varied constraints for fluid catalytic cracking operation," *IEEE Trans. Cybern.*, vol. 52, no. 4, pp. 2249–2262, Apr. 2020.

[29] G. Zhang, K. Xing, and F. Cao, "Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan," *Int. J. Prod. Res.*, vol. 56, no. 9/10, pp. 3226–3244, 2018.

[30] G. Zhang and K. Xing, "Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion," *Eng. Appl. Artif. Intell.*, vol. 76, pp. 96–107, 2018.

[31] Y. Z. Hsieh and M. C. Su, "A Q-learning-based swarm optimization algorithm for economic dispatch problem," *Neural Comput. Appl.*, vol. 27, pp. 2333–2350, 2016.

[32] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, 2007.

[33] H. Zohali, B. Naderi, M. Mohammadi, and V. Roshanaei, "Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems," *Comput. Operations Res.*, vol. 104, pp. 127–138, 2019.

[34] M. Kurdi, "Ant colony system with a novel Non-DaemonActions procedure for multiprocessor task scheduling in multistage hybrid flow shop," *Swarm Evol. Comput.*, vol. 44, pp. 987–1002, 2019.

[35] A. Sl, C. Jpa, C. A. Hao, B. Xla, and C. Pmp, "Two-stage hybrid flow shop scheduling on parallel batching machines considering a job-dependent deteriorating effect and non-identical job sizes," *Appl. Soft Comput.*, vol. 84, pp. 1–15, 2019.

[36] H. Ztop, M. F. Tasgetiren, D. T. Eliiyi, and Q. K. Pan, "Metaheuristic algorithms for the hybrid flowshop scheduling problem," *Comput. Operations Res.*, vol. 111, pp. 177–196, 2019.

[37] V. Riahi, M. Newton, K. Su, and A. Sattar, "Constraint guided accelerated search for mixed blocking permutation flowshop scheduling," *Comput. Operations Res.*, vol. 102, pp. 102–120, 2019.

[38] H. Luo, G. Q. Huang, Y. Zhang, Q. Dai, and X. Chen, "Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 25, no. 6, pp. 962–971, 2009.

[39] A. Missaoui and Y. Boujelbene, "An effective iterated greedy algorithm for blocking hybrid flow shop problem with due date window," *RAIRO -Operations Res.*, vol. 55, no. 3, pp. 1603–1616, 2021.

[40] J. Gao, R. Chen, and W. Deng, "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 3/4, pp. 641–651, 2013.

[41] A. P. Rifai, H. T. Nguyen, and S. Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Appl. Soft Comput.*, vol. 40, pp. 42–57, 2016.

[42] Q. K. Pan, L. Gao, L. X. Yu, and F. M. Jose, "Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem," *Appl. Soft Comput.*, vol. 81, 2019, Art. no. 105492.

[43] R. Ruiz, Q. K. Pan, and B. Naderi, "Iterated greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, no. 3, pp. 213–222, 2019.

[44] Q. K. Pan, L. Gao, and L. Wang, "An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 5999–6012, Jul. 2022.

[45] H. Ochi and O. B. Driss, "Scheduling the distributed assembly flowshop problem to minimize the makespan," *Procedia Comput. Sci.*, vol. 164, pp. 471–477, 2019.

[46] Y. Y. Huang, Q. K. Pan, J. P. Huang, P. N. Suganthan, and L. Gao, "An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 152, no. 3, pp. 1–11, 2021.

[47] Z. Shao, W. Shao, and D. Pi, "Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem," *Knowl.-Based Syst.*, vol. 221, no. 5, pp. 1–19, 2021.

[48] S. Chen, Q. K. Pan, and L. Gao, "Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm," *Robot. Comput.- Integr. Manuf.*, vol. 71, no. 3, pp. 1–16, 2021.

[49] 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.

[50] D. M. Lei and T. Wang, "Solving distributed two-stage hybrid flowshop scheduling using a shuffled frog-leaping algorithm with memeplex grouping," *Eng. Optim.*, no. 12, pp. 1–14, 2019.

[51] J. -Q. Li et al., "Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2425–2439, Jun. 2020.

[52] J. Zheng, L. Wang, and J. J. Wang, "A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop," *Knowl.-Based Syst.*, vol. 194, pp. 1–11, 2020.

[53] J. -J. Wang and L. Wang, "A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 5, pp. 1805–1819, May 2020.

[54] M. Nawaz, E. Jr, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[55] J. P. Huang, Q. K. Pan, and L. Gao, "An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times," *Swarm Evol. Comput.*, vol. 59, 2020, Art. no. 100742.

[56] R. Sutton and A. Barto, *Reinforcement Learning:An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[57] Y. H. Wang, T. H. S. Li, and C. J. Lin, "Backward Q-learning: The combination of sarsa algorithm and Q-learning," *Eng. Appl. Artif. Intell.*, vol. 26, no. 9, pp. 2184–2193, 2013.

[58] J. C. H. Watkins, Christopher, and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, 1992.

[59] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 149, no. 1993, pp. 1–12, 2020.


**Haoxiang Qin** received the B.Sc. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, where he is currently working toward the M.S. degree. His research interests include intelligent optimization and scheduling.

**Yuyan Han** (Member, IEEE) received the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2012, and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, Xuzhou, China, in 2016. Since 2016, she has been an Associate Professor with the School of Computer Science, Liaocheng University. She has authored more than 30 refereed papers. Her research interests include evolutionary computation, multiobjective optimization, and flow-shop scheduling.

**Qingda Chen** (Member, IEEE) received the Ph.D. degree in control theory and engineering from Northeastern University, Shenyang, China, in 2020. Since 2021, he has been working with the State Key Laboratory of Synthetical Automation for Process Industries of Northeastern University. He has authored or coauthored more than ten papers. His research interests include modeling, plant-wide control and optimization for the complex industrial systems, stochastic distribution control, and multiobjective evolutionary algorithms and its applications.

**Ling Wang** (Member, IEEE) received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively. Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a Full Professor in 2008. He has authored five academic books and more than 260 refereed papers. His research interests include intelligent optimization and production scheduling. Professor Wang was the recipient of the National Natural Science Fund for Distinguished Young Scholars of China, National Natural Science Award (Second Place) in 2014, Science and Technology Award of Beijing City in 2008, Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. Professor Wang is currently the Editor-In-Chief of the *International Journal of Automation and Control*, and an Associate Editor for IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

**Yuting Wang** received the master's degree in computer software and theory from the China University of Petroleum Beijing, China, in 2005. Since 2013, he has been an Associate Professor with the School of Computer Science, Liaocheng University, Liaocheng, China. He has authored or coauthored more than 20 papers. His research interests include mathematical modeling, intelligent optimization algorithms, and software development technology.

**Junqing Li** (Member, IEEE) received the master's degree in computer science and technology from Shandong Economic University, Shandong, China, in 2004 and the Ph.D. degree in system engineering from Northeastern University, Shenyang, China, in 2016. Since 2004, he has been with the School of Computer, Liaocheng University, Liaocheng, China. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, Jinan, China, where he became a Professor in 2017. He has authored more than 70 refereed papers. His research interests include intelligent optimization and scheduling.

**Yiping Liu** (Member, IEEE) received the B.Eng. degree in electrical engineering and automation and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, Xuzhou, China in 2012 and 2017, respectively. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. During 2018–2020, he was a Research Assistant Professor with the Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Sakai, Japan. During 2016–2017, he was a Visiting Scholar with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. His research interests include evolutionary computation, multi-objective optimization, and machine learning.