mathematics

MDPI

Article

Hybrid Multi-Objective Artificial Bee Colony for Flexible Assembly Job Shop with Learning Effect

Zhaosheng Du [1,2], Junqing Li [1,3,*] and Jiake Li [1]

[1] Department of Mathematics, Yunnan Normal University, Kunming 650500, China; 220170224@stu.lcu.edu.cn (Z.D.); lijiake_hainan@163.com (J.L.)
[2] School of Computer, Liaocheng University, Liaocheng 252000, China
[3] School of Information Engineering, HengXing University, Qingdao 266104, China
* Correspondence: lijunqing@lcu-cs.com

Abstract: The flexible job shop scheduling problem is a typical and complex combinatorial optimization problem. In recent years, the assembly problem in job shop scheduling problems has been widely studied. However, most of the studies ignore the learning effect of workers, which may lead to higher costs than necessary. This paper considers a flexible assembly job scheduling problem with learning effect (FAJSPLE) and proposes a hybrid multi-objective artificial bee colony (HMABC) algorithm to solve the problem. Firstly, a mixed integer linear programming model is developed where the maximum completion time (makespan), total energy consumption and total cost are optimized simultaneously. Secondly, a critical path-based mutation strategy was designed to dynamically adjust the level of workers according to the characteristics of the critical path. Finally, the local search capability is enhanced by combining the simulated annealing algorithm (SA), and four search operators with different neighborhood structures are designed. By comparative analysis on different scales instances, the proposed algorithm reduces 55.8 and 958.99 on average over the comparison algorithms for the GD and IGD metrics, respectively; for the C-metric, the proposed algorithm improves 0.036 on average over the comparison algorithms.

Keywords: artificial bee colony; flexible assembly job shop; learning effect; hybrid multi-objective algorithm; critical path

MSC: 90-08

check for updates

Academic Editor: Helena Sofia Rodrigues

Received: 29 November 2024
Revised: 19 December 2024
Accepted: 24 December 2024
Published: 31 January 2025

Citation: Du, Z.; Li, J.; Li, J. Hybrid Multi-Objective Artificial Bee Colony for Flexible Assembly Job Shop with Learning Effect. *Mathematics* **2025**, *13*, 472. https://doi.org/10.3390/math13030472

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The flexible job shop scheduling problem (FJSP) is a complex production scheduling problem (PSP) with a wide range of application scenarios and has been shown to be one of the most classical NP-hard problems [1]. FJSP is a multi-process, multi-machine PSP, which usually consists of a set of jobs and a set of machines. In this case, each job contains multiple operations that are processed on optional machines in a specific order of operations. Each machine can process multiple operations, but only one operation at a time, and only one machine can be selected at a time for each operation. In recent years, for FAJSPs, researchers have conducted a lot of research and proposed many effective algorithms to solve this problem. Methods used to solve FJSPs can be roughly categorized into exact and inexact methods. Examples of methods used for exact solutions are branch-and-bound [2], branch-and-cut [3], and mixed-integer linear programming models (MILPs) [4], which are more focused on the optimal solution and often take a lot of computational time, especially when

solving large-scale problems. Heuristics [5], meta-heuristics [6] and inexact algorithms such as hybrid algorithms [7] seem to be more competitive in solving large-scale problems.

However, with the changing production environment, it is often necessary to consider other processing stages during or after the production process. The flexible assembly job shop scheduling problem (FAJSP) is mainly applied to manufacturing systems that require assembly operations, which adds an assembly phase to FJSPs [8]. FAJSPs usually involve several processes and stations, each of which has different operational tasks and each of which may require different assembly resources such as tools, personnel and equipment. In FAJSPs, not only processing processes are involved but also multiple assembly processes, leading to more complexity in resource allocation. As the completion time of the production stage may lead to delays in the assembly stage, this may prolong the whole processing as well as cause resource wastage. Therefore, a reasonable scheduling strategy is needed to optimize the completion time.

The FAJSPLE added the learning effect to the FAJSP and considered the proficiency of the workers in operating the machine. Most studies have focused on only a single objective of the problem. In this paper, the objective is to minimize the completion time, total energy consumption and total cost of the FAJSPLE, and a HMABC algorithm is designed to find the Pareto frontier based on the characteristics of the problem. Firstly, an MILP model of the problem is developed to optimize the three objectives simultaneously. In addition, a critical path-based mutation strategy is designed for the characteristics of the problem, and an SA based neighborhood search algorithm is used to prevent from falling into local optima. The algorithm is designed to solve large-scale problem instances efficiently.

The rest of the paper is organized as follows. Section 2 reviews the related literature; Section 3 demonstrates the problem description and mathematical modeling; Section 4 details the HMABC algorithm proposed in this paper; Section 5 evaluates the effectiveness of the algorithm on instances of different scales; and Section 6 concludes the research results and prospective research topics.

2. Literature Review

The reference review in this section is divided into three parts as follows: related studies of assembly scheduling problems, flexible job shop scheduling problems with multiple constraints, and multi-objective optimization algorithms.

2.1. Assembly Scheduling Problems

In practical PSPs, products are often assembled from several subassemblies, and this type of problem is known as an assembly scheduling problem (ASP). The FJSP has been proven as NP-hard, and the two-stage AFJSP is added to the FJSP, so the two-stage AFJSP is also NP-hard. In the actual production scheduling, the product is assembled into multiple parts, and it is generally produced in the "part–component–product" process. Considering the production scheduling of assembly is generally called assembly scheduling, which can be roughly divided into two categories: assembly flow shop problems (AFSPs) and assembly job shop problems (AJSPs). For the former, the jobs are first processed in the flow shop and then assembled on a single assembly machine. Wu [9] and Kazemi et al. [10] considered processing and assembly from the perspective of the FSP. Mozdgir [11] and Navaei et al. [12] solved a two-stage hybrid AFSP where multiple assembly machines were considered; therefore, it can perform the concurrent assembly operation. Fattahi et al. [13] assumed the production phase was a hybrid flow shop and developed heuristic algorithms to optimize the makespan. Liao et al. [14] studied batch setup times in a two-stage AFSP and designed an efficient heuristic algorithm. Wu et al. [9] studied the position-based learning method in a two-stage AFSP. In addition, they also proposed three heuristics along

with different simulated annealing algorithms. Compared to the AFSP, the AJSP is more complex. Chan et al. [15] applied the lot streaming (LS) technique to the AJSP. Wong [16] solved the AJSP using LS by minimizing the makespan. The AJSP of the alternative processing machine is called the assembly flexible job shop scheduling (AFJSP). Zhang et al. [17] proposed a constraint programming model for the FAJSP in which partial sharing is allowed rather than production being associated with a specific product.

According to the above-described analysis, less work has been conducted on the two-stage assembly flexible job scheduling problem (AFJSP), which simultaneously optimizes completion time and energy consumption. Even few studies consider transportation and setup times in two-stage AFJSP. Accordingly, facing the cumbersome two-stage AFJSP, it is necessary to design an optimization algorithm that meets the problem characteristics to solve the above complex scheduling problem.

2.2. Flexible Job Shop Scheduling Problems with Multiple Constraints

For the transportation time, a disjunctive graph model constrained by the machine-dependent transportation time was introduced in the canonical JSP [18,19]. For the FJSP, the transportation time matrix for transporting jobs between two machines was used, where mathematical models or meta-heuristics were developed to find the optimal scheduling solution [20–22]. Karimi et al. [23] combined the transportation time between machines while using the imperialist competition algorithm (ICA) to solve the FJSP. The transportation times are considered critical issues for vehicle routing problems [24]. Zhang et al. [25] incorporated processing and transportation time into the FJSP. After presenting the issue, an improved genetic algorithm is proposed to minimize the makespan and total transportation time. Liu et al. [26] designed an integrated genetic algorithm and glowworm swarm optimization (GA-GSO) algorithm for the FJSP with crane transportation. Dai et al. [27] utilized an enhanced genetic algorithm to address a multi-objective FJSP with transportation constraints.

For the setup time, Azzouz et al. [28] proposed a hybrid algorithm based on a genetic algorithm and variable neighborhood search to study the FJSP with sequence-dependent setup times. Aiming at minimizing total tardiness, Moousakhani et al. [29] designed an efficient meta-heuristic algorithm based on an iterative local search. Li et al. [30] introduced an improved Jaya (IJaya) to solve the FJSP considering the setup and transportation times. Saidi-Mehrabad et al. [31] presented a hierarchical TS algorithm to solve the FJSP with setup times in an attempt to find the optimal sequences. Rossi [22] solved the FJSP with release dates, setup times, and transportation times by an ant colony algorithm. Du et al. [32,33] considered setup times in the FJSP. Li et al. [34] developed an elitist non-dominated sorting hybrid algorithm (ENSHA) to solve the multi-objective FJSSP with sequence-dependent setup times/costs. The objectives to be minimized were the makespan and total setup costs. Sun et al. [35] addressed a multi-objective FJSSP with setup times, established a network graph model, and developed a new neighborhood structure to optimize completion time, total workload, and penalties of earliness/tardiness.

For the learning effect, Tayebi et al. [28] integrated the worker learning effect into the FJSP to solve the sequence-related preparation time. The effectiveness of the hybrid algorithm combining the genetic algorithm and variable neighborhood search was analyzed. Gong [29] studied the multi-objective FJSP of machine and worker flexibility and built a nonlinear integer programming model. Liu et al. [30] established a scheduling model with dual-objective resource constraints. A hybrid genetic algorithm based on Pareto is proposed to rationally allocate machine resources and worker resources. Luo et al. [31] proposed a Multi-Objective Memetic Algorithm based on Learning and Decomposition (MOMA-LD). The learning-based adaptive local search is incorporated into the Multi-Objective

Evolutionary Algorithm based on Decomposition (MOEA/D). Zhu et al. [32] considered the impact of workers' learning ability on processing time and energy consumption, and they used meme algorithms to solve low-carbon FJSP with a learning effect and optimize the maximum completion time and total cost of workers. Luo et al. [33] studied the FJSP with worker cooperation flexibility and established a suitable mathematical model. A Pareto-based two-stage evolutionary algorithm is proposed, and a local search operator based on competitive objectives is designed to improve the local search capability.

For the FJSP optimization problem, at present, few studies in the literature comprehensively consider transportation time, setup time and learning effect constraints. However, there are even fewer researchers studying these three constraints in the FAJSP. Therefore, it is necessary to analyze the effects of the three factors in the FAJSP.

2.3. Multi-Objective Optimization Algorithm

Zou et al. [36] and Hedayatzadeh et al. [37] presented the multi-objective (MO) ABC algorithm. Kishor et al. [38] extended a non-dominated sorting-based multi-objective ABC to achieve two orthogonal goals of convergence and diversity, which is based on the single-objective (SO) ABC. Akay [39] developed three different MOABCs by changing the selection and fitness allocation strategies in the original ABC. Toktas et al. [40] designed a 3D objective space optimization strategy using an enhanced MOABC for the layered radar absorbing material. Li et al. [41] utilized the ABC to optimize the reverse logistics network system. Gao et al. [42] improved the canonical ABC via orthogonal learning and a modified search equation. By factoring into the model the selection of a hybrid energy storage system, Shan et al. [43] established an economically optimized MO dispatching model based on an improved ABC. For the optimal power flow problem of electric power systems, Chen et al. [44] extended the original ABC to the MO cooperative mode. Chen proposed a multi-hive multi-objective bee algorithm combining external archiving greedy learning and collaborative search strategies. Multi-objective algorithms were applied in scheduling, e.g., software requirement optimization, vehicle routing, and resource balancing, respectively [45–50].

Due to its outstanding performance, MOABC has been applied to a variety of complex scheduling problems. However, in FAJSPLE problems, most researchers tend to ignore the characteristics of workers' flexibility. Therefore, this paper improves the MOABC algorithm and designs an effective mutation strategy and neighborhood search algorithm for the characteristics of the FAJSPLE.

3. Problem Description and Mathematical Model

3.1. Problem Description

The FAJSPLE can be briefly stated as follows. It considers processing n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ on m processable machines $M = \{M_1, M_2, \ldots, M_m\}$ with l workers $W = \{W_1, W_2, \ldots, W_l\}$. Each job $J_j \in J$ has a sequence of operations $O_j = \left\{O_{j,1}, O_{J,2}, \ldots O_{j,l}\right\}$, $O_{j,l}$ is the lth operation of the J_j, $(j = 1, 2, \ldots, n)$. Each operation $O_{j,l}$ can be executed on any machine M_i among an optional set $M' \subseteq M$. Furthermore, workers have flexibility, i.e., each worker can manipulate several machines. The skill proficiency varies from worker to worker and directly influences the actual processing time of the operation. On assembly machine M_A, the n jobs are assembled into pr products $P = \left\{P_1, P_2, \ldots, P_{pr}\right\}$. Each product P_p with assembly time A_p consists of N_p jobs and satisfies $\sum_{p=1}^{pr} N_p = n$. The assembly operations of P_p would not start unless N_p jobs required for P_p are processed in the first stage. If two consecutive operations on the same machine are from two different jobs, the setup time is needed. In addition, the transportation of jobs

between machines also costs energy and time. The objectives are to optimize the C_{max} total cost and TEC.

For ease of understanding, Figure 1 shows a simple example of the FAJSPLE with two products, four machines, and five workers along with the skill proficiency. Notably, proficiency less than, equal to, and greater than 1.0 indicate that the worker is skilled, average, and unskilled to manipulate the machine, respectively. The higher the level, the higher the skill level of the worker, and the shorter the processing time of the machine to operate, but the cost has increased.

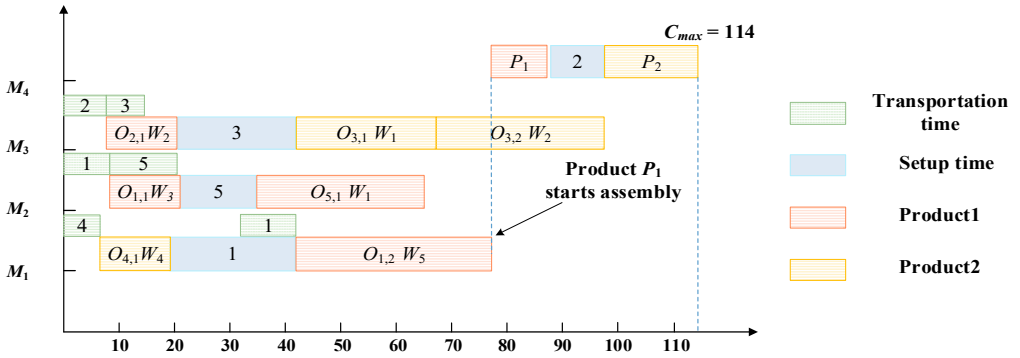


Figure 1. The Gantt chart of FAJSPLE example.

3.2. Assumptions and Notations

The studied problem is based on the following assumptions:

(1) Each machine can only process one operation at a time, and each sub-part can be processed only by one of its candidate machines.
(2) The processing time and assembly time are certain; specify the requirements for the final product in advance.
(3) The processing and assembly stages are independent of each other. Assembly operation is possible only after all the operations in the same group have been processed.
(4) At any time, each worker can operate only one machine chosen from the corresponding machine set.
(5) All machines remain continuously available. Interruptions are not considered for each operation.
(6) Different operations have different setup times, and if the processes of a job are continuously processed on the same machine, the setup time is 0.
(7) When the same job is carried out on different machines, the transportation time varies with the machine used. If two operations of a job are processed on the same machine, the transportation time is 0.

The notations used in the established mathematical model of the AFJSPLE are listed in Table 1.

Table 1. Notations and descriptions.

Indices:	**Descriptions:**
i	Machines index, $i = 1, 2, \ldots, m$.
s,p	Products index, $s, p = 1, 2, \ldots, pr$.
j, h	Jobs index, $j, h = 1, 2, \ldots, n$.
w	Workers index

Table 1. *Cont.*

Parameters:	
n	Total number of jobs.
m	Total number of machines.
pr	Total number of the products.
wn	Total number of the workers.
$Pe_{i,w}$	Proficiency of worker w on machine i.
n_j	Total number of the operations of job j.
$O_{j,l}$	lth operation of job j.
$P_{j,l,i}$	Standard processing time of $O_{j,l}$ on machine i.
$\widetilde{P}_{j,l,i}$	Actual processing time of $O_{j,l}$ on machine i.
$PC_{j,l,i,w}$	Cost of $O_{j,l}$ on machine i with worker w.
$S_{i,h,j}$	Setup time between job j and job h on machine i.
$SP_{p,s}$	Setup time between product p and product s.
$T_{j,k,i}$	Transportation time of job j from machine k to machine i.
Ap	Assembly time of product p.
$UEC_{j,l,i}$	Unit energy consumption of $O_{j,l}$ on machine i.
AEC_p	Assembly energy consumption of product p.
EC_j	Total energy consumption of job j.
$C_{j,l}$	Completion time of $O_{j,l}$ in the first stage.
ST_p	Start assembly time of product p.
CA_P	Completion time of product p.
TEC	Total energy consumption.
$C_{\max}$	Maximum assembly completion time.
TC	Total cost
M	A large positive number.
Decision variables:	
$X_{j,l,h,z}$	1 if $O_{j,l}$ is processed after $O_{h,z}$ and 0 otherwise. $j = \{1, 2, \ldots, n-1\}, h > j$.
$Y_{j,l,i,k}$	1 if $O_{j,l}$ is processed on machine i and $O_{j,l-1}$ on machine k.
$H_{j,l,i,w}$	1 if $O_{j,l}$ is processed on machine i with worker w.
$Y_{j,p}$	1 if job j is a sub-part of product p.
$Z_{s,p}$	1 if product s to be assembled just before each product p.

3.3. Multi-Objective Mathematical Model

An effective mathematical model is the key to solve complex scheduling problems. Ref. [8] summarized the definition of the FAJSP with related constraints and optimized the three objectives of the makespan, the total tardiness, and the total workload simultaneously. Ref. [51] considered the location-based learning effect on the basis of the FAJSP-SDST and optimized the makespan. In this paper, a mathematical model of the FAJSPLE is developed to simultaneously optimize the three objectives of the makespan, the total cost and the total energy consumption.

Considering the proficiency effect of each worker, the actual processing time for jobs processed by a worker on each machine is designed as

$$\widetilde{P}_{j,l,i} = P_{j,l,i,w} \times Pe_{i,w} \tag{1}$$

In this paper, the objective of the FAJSPLE is to minimize the makespan ($C_{\max}$), total cost (TC) and energy consumption (TEC) with the objective functions shown in Formulas (2)–(4).

$$f_1 = \min C_{\max} = \min(\max CA_p) \tag{2}$$

$$f_2 = \min TC = \min(\sum_{j=1}^{n} \sum_{l=1}^{n_j} \sum_{i=1}^{m} \sum_{w=1}^{wn} PC_{j,l,i,w} H_{j,l,i,w}) \tag{3}$$

$$f_3 = \min TEC = \min(\sum_{j=1}^{n} EC_j + \sum_{p=1}^{pr} AEC_p \cdot A_p) \tag{4}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{m} Y_{j,l,i,k} = 1 \tag{5}$$

$$\sum_{w=1}^{wn} H_{j,l,i,w} = 1 \tag{6}$$

$$\sum_{k=1}^{m} Y_{j,l,i,k} \le e_{j,l,i} \tag{7}$$

$$C_{j,l} \ge C_{j,l-1} + \sum_{i=1}^{m}\sum_{k=1}^{m}\sum_{w=1}^{wm} Y_{j,l,i,k}(\widetilde{P}_{j,l,i,w} \cdot H_{j,l,i,w} + T_{j,k,i} + \sum_{h=1}^{n} X_{j,l,h,z} \cdot S_{i,h,j}) \tag{8}$$

$$ST_p \ge C_{j,n_j} - M(1 - Y_{j,p}) \tag{9}$$

$$CA_p \ge A_p + ST_p + SP_{s,p} + M(1 - Z_{s,p}) \tag{10}$$

$$CA_p \ge CA_s + A_p + M(1 - Z_{s,p}) \tag{11}$$

$$EC_j \ge \sum_{l=1}^{n_j}\sum_{i=1}^{m}\sum_{w=1}^{wn} UEC_{j,l} \cdot e_{j,l,i} \cdot \widetilde{P}_{j,l,i,w} \tag{12}$$

Constraint (5) ensures that each operation can only be processed by one machine. Constraint (6) ensures that each operation can only be processed by one worker operating the machine. Constraint (7) is defined to make sure the machine is selected from the eligible machines for $O_{j,l}$. Constraint (8) indicates that for job j, the next operation $O_{j,l+1}$ can be performed only after its current operation $O_{j,l}$ is completed. Constraint (9) represents the earliest start time for assembly. Constraint (10) indicates the completion time of the product. Constraint (11) ensures assembly constraints between products. Constraint (12) gives the total energy consumption of job j.

3.4. Properties of FAJSPLE

According to the definition of the critical path for a JSP problem, i.e., the longest path from the first step of the first job to the last step of the last job [52], the FSJSPLE also has a critical path. Jobs on the critical path determine the completion time of the whole process if by increasing the level of workers on the critical path, the processing time of the tasks can be reduced and the makespan of the whole problem can be shortened.

Property 1. *For the FAJSPLE, improve worker levels for operators along critical paths, reducing completion time and energy consumption.*

Proof of Property 1. Assume that job j is on the critical path with a start time S_j and a learning effect coefficient α. If the level of workers is increased, S_j decreases to $S_j^{'}$, the length of the entire critical path decreases, and then the makespan also decreases. In addition, since the TEC depends on the processing time, the total energy will also decrease. Therefore, the makespan and TEC can be reduced by increasing the level of workers on the critical path. □

Property 2. *Lower labor levels for operators on non-critical paths, thereby reducing costs.*

Proof of Property 2. Since jobs on non-critical paths have no effect on the makespan, increasing the level of its workers will not affect the critical path. If the level of its workers is reduced, it will only increase the start time of the jobs on the non-critical path without

affecting the makespan. In addition, reducing the level of workers on the non-critical path reduces the labor cost, which optimizes the overall cost. □

Specific examples of this process are demonstrated in Section 4.4.1.

4. Algorithm Descriptions

4.1. Framework of Proposed HMABC

The ABC algorithm is an optimization algorithm based on honeybee swarm behavior. It performs well in solving multi-objective job shop scheduling problems [46]. It usually consists of the steps of initialization, employed bees to update the neighborhood of the current solution, onlooker bees to select high-quality solutions and optimize them further and scout bees to replace solutions that have not been improved for a long time. Since the algorithm has good adaptability and can dynamically adjust the search strategy to be more adapted to multi-objectives, we improved the algorithm to solve the FAJSPLE.

Figure 2 shows the HMABC algorithm framework. In order to make the Pareto set explored have good convergence and diversity, and it is reasonable for different solutions to search in different directions, the HMABC first standardizes individuals in the initial population before executing the employed bee stage. Then, in the employed bee stage, POX crossover is performed on the standardized individual OS vector, random mutation operation is performed on the MS vector and knowledge-based critical path mutation operation is performed on the WS vector. In the onlooker bee stage, the SA is used for the local search. Each time the algorithm moves, a new solution will be generated around the current solution, which can improve the local optimization ability. rank(x) represents the Pareto level of solution x, $m = PS/5$.

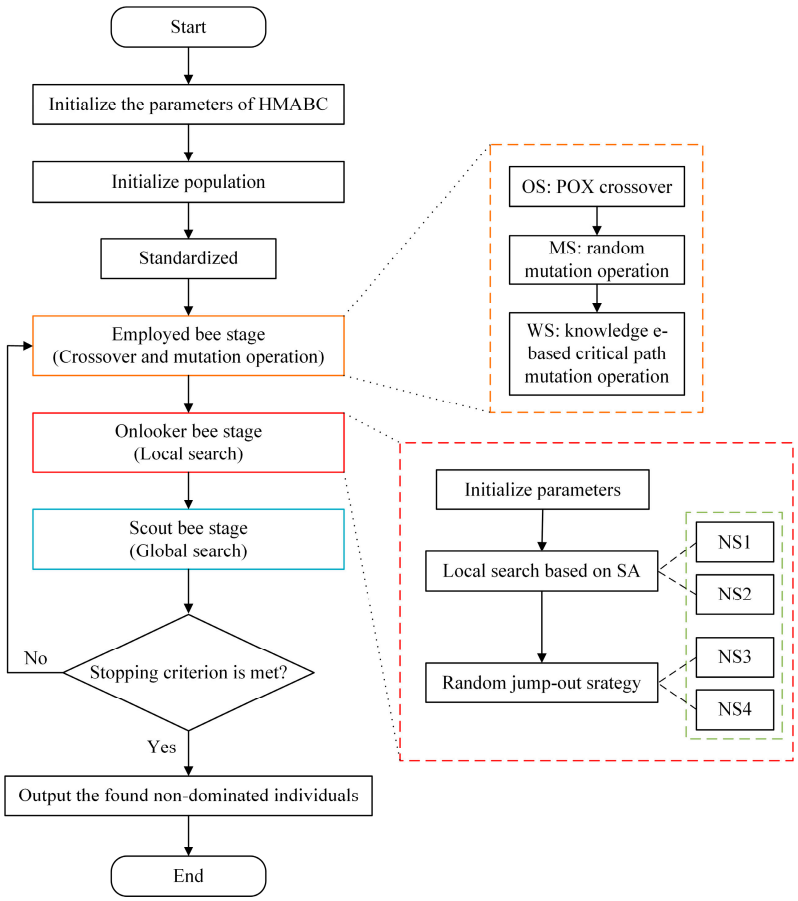


Figure 2. The framework of HMABC.

4.2. Chromosome Encoding and Decoding

A three-layer chromosome encoding method, comprising the machine selection (MS), worker assignment (WA) and operation sequence (OS), is designed to show a scheduling solution. In this case, MS reports the assigned machine for each sub-part. WA represents the sequence of workers whose operations are processed on the machine. The number in each part reveals the worker–number selected for the corresponding operation. The OS consists of two vectors. (1) The first records the processing sequence of all parts. The indicator of each job is duplicated n_j times. (2) The second vector notes the product to which the part belongs, corresponding to the first vector. Both vectors have a length equal to $\sum_{j=1}^{n} = 1\ n_j$.

To clarity this representation, Figure 3 gives an example with four machines, three parts and three workers.

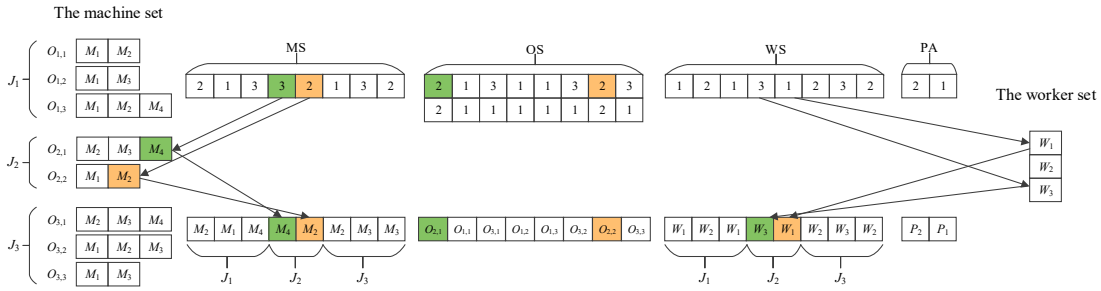


Figure 3. The representation of the encoding for FAJSPLE.

The decoding process takes the knowledge mentioned in Section 3.4 into account. When a solution is decoded, the OS is converted into a sequence of operations. Each operation is assigned to a selected machine from the MS and corresponded worker. Finally, each optimal objective is calculated according to its rules.

4.3. Initial Population

To initialize a group of solutions, a simple and effective method is utilized in this study. The detailed steps are as follows.

Step 1. In order to obtain a high-quality initial population, the initial MS is generated by means of Global Selection (GS), Local Selection (LS) and Random Selection (RS), respectively. GS selects machines for each operation of each workpiece based on the machining cost; i.e., it selects the machine with the lowest machining cost. LS selects the optimal machine among the neighboring machines for each operation. RS assigns the machines randomly for each operation of each workpiece.

Step 2. The sequences of OS and WA are randomly determined.

Step 3. The product assembly (PA) sequence depends on the OS of the processing stage, so the PA sequence is determined based on the OS.

The main advantage of this encoding scheme is that unfeasible solutions are not produced.

4.4. Offspring Reproduction

4.4.1. Crossover and Mutation

The crossover and mutation operators are crucial operations in the employed bee phase. To balance various optimization objectives, the individuals in the initial population were standardized before executing the employed bee phase. In the FAJSPLE, considering the production energy consumption, the maximum completion time and total energy consumption are equally important. Therefore, only the completion time and cost are standardized in this context.

The process of standardization is as follows.

Step 1: For the solution x, Formulas (13) and (14) are executed, where $C_{\max}^{\min}$, $C_{\max}^{\max}$, $TC^{\min}$, and $TC^{\max}$ refer to the minimum value of the $C_{\max}$, maximum value of the $C_{\max}$, minimum value of the cost and maximum value of the current population, respectively.

$$\overline{C}_{\max}(x) = (C_{\max}(x) - C_{\max}^{\min})/(C_{\max}^{\max} - C_{\max}^{\min}) \tag{13}$$

$$\overline{TC}(x) = (TC(x) - TC^{\min})/(TC^{\max} - TC^{\min}) \tag{14}$$

Step 2: The γ value of the solution x is calculated by the formula (15).

$$\gamma(x) = \overline{TC}(x)/\overline{C}_{\max}(x) \tag{15}$$

Step 3: The solutions in the population were sorted in ascending order according to the γ value, and the population was divided into two equal sub-populations P_a and P_b. P_a had a small γ value—that is, the completion time was relatively large; and P_b had a large γ value—that is, the cost was the dominant factor.

During the employed bee phase, the POX (Precedence Operation Crossover) operation is applied to the OS vectors. The execution process is as follows, and Figure 4 illustrates an example of performing the POX.

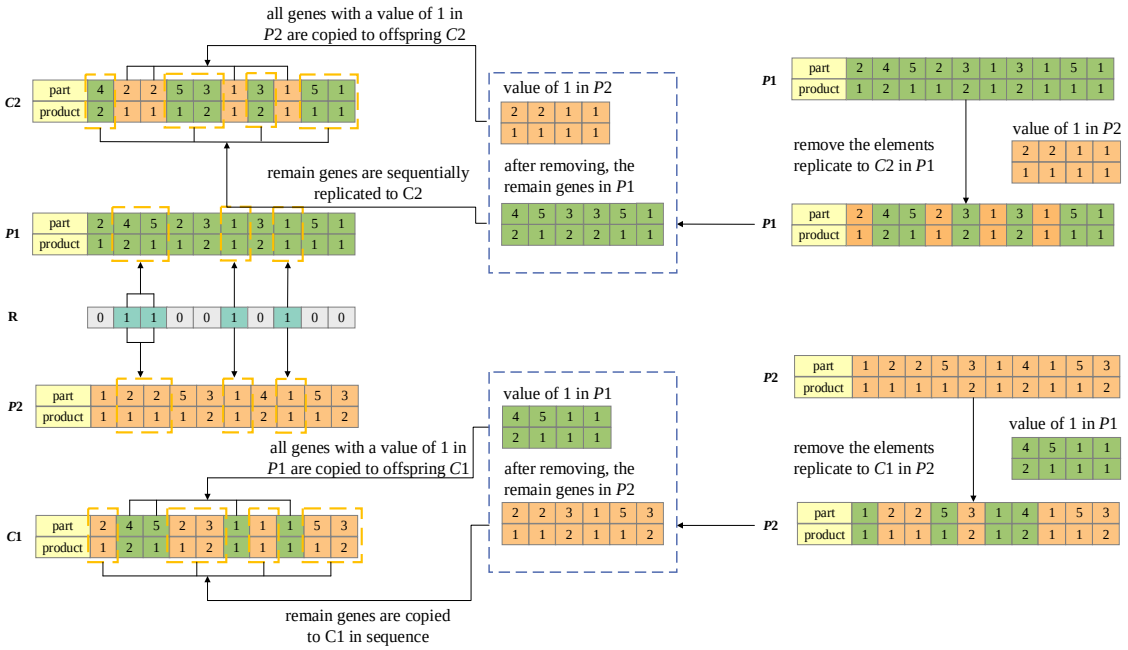


Figure 4. An example of the POX.

Step 1: For the parental food source $P1$, another parental food source $P2$ is randomly chosen for crossover.

Step 2: Define two parts sets $J1$ and $J2$, dividing even part numbers to $J1$ and odd part numbers to $J2$.

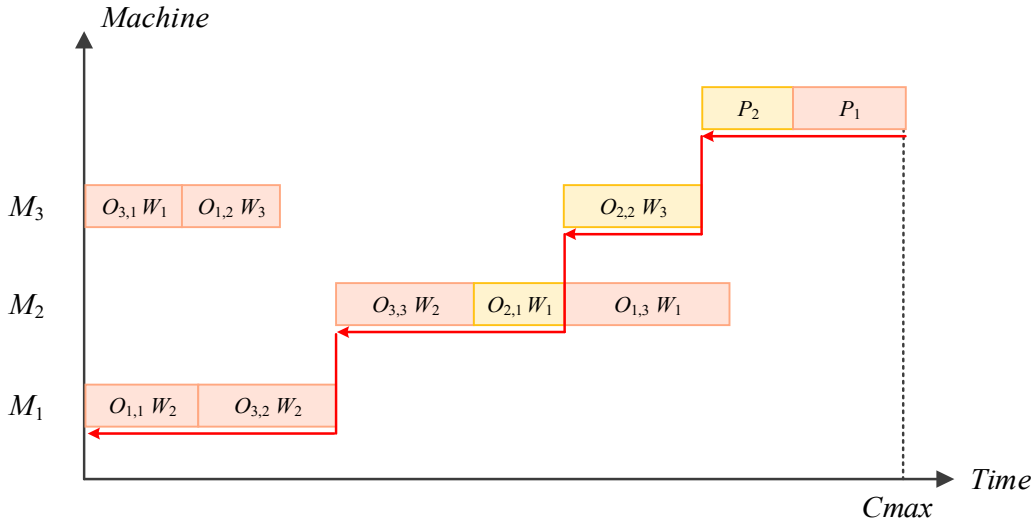
Step 3: The offspring $C1$ inherits the parts belonging to $J1$ in $P1$, and the offspring $C2$ inherits the parts belonging to $J2$ in $P2$ and retains their positions.

Step 4: The parts including $J2$ are removed in $P1$, and the remaining parts are assigned to $C2$ in the original order. Similarly, the parts including $J1$ are removed in $P2$, and the remaining parts are copied to $C1$ in the original order.

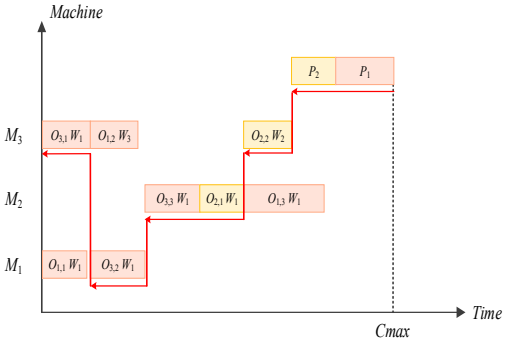
Step 5: Change the corresponding product arrangement.

The WS vector performs a mutation operation based on the critical path with the process as follows.

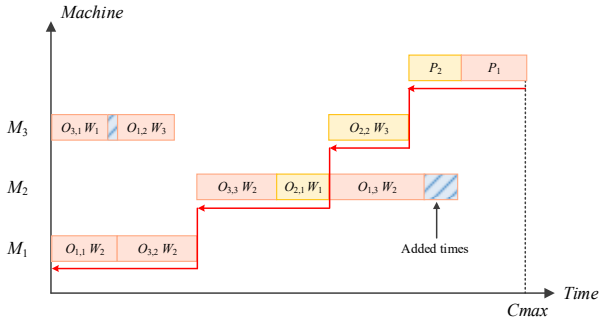
Step 1: Determine the critical path of the current individual x. The critical path is a continuous sequence of operations from completing the last operation back to the beginning of the entire process, where there is no idle time between any two operations. Operations within the critical path are referred to as critical operations. Figure 5a provides an example of a critical path in the FAJSPLE, where W_1 corresponds to skill level 1, W_2 corresponds to skill level 2, and W_3 corresponds to skill level 3.



(**a**) An example of the critical path



(**b**) Improving the level of the workers



(**c**) Lowering the level of the workers

Figure 5. An example of the mutation.

Step 2: If $x \in P_a$, then proceed to step 3, else if $x \in P_b$, then proceed to step 4.

Step 3: Improve the level of workers on the critical path and reduce the completion time and energy consumption, as shown in Figure 5b.

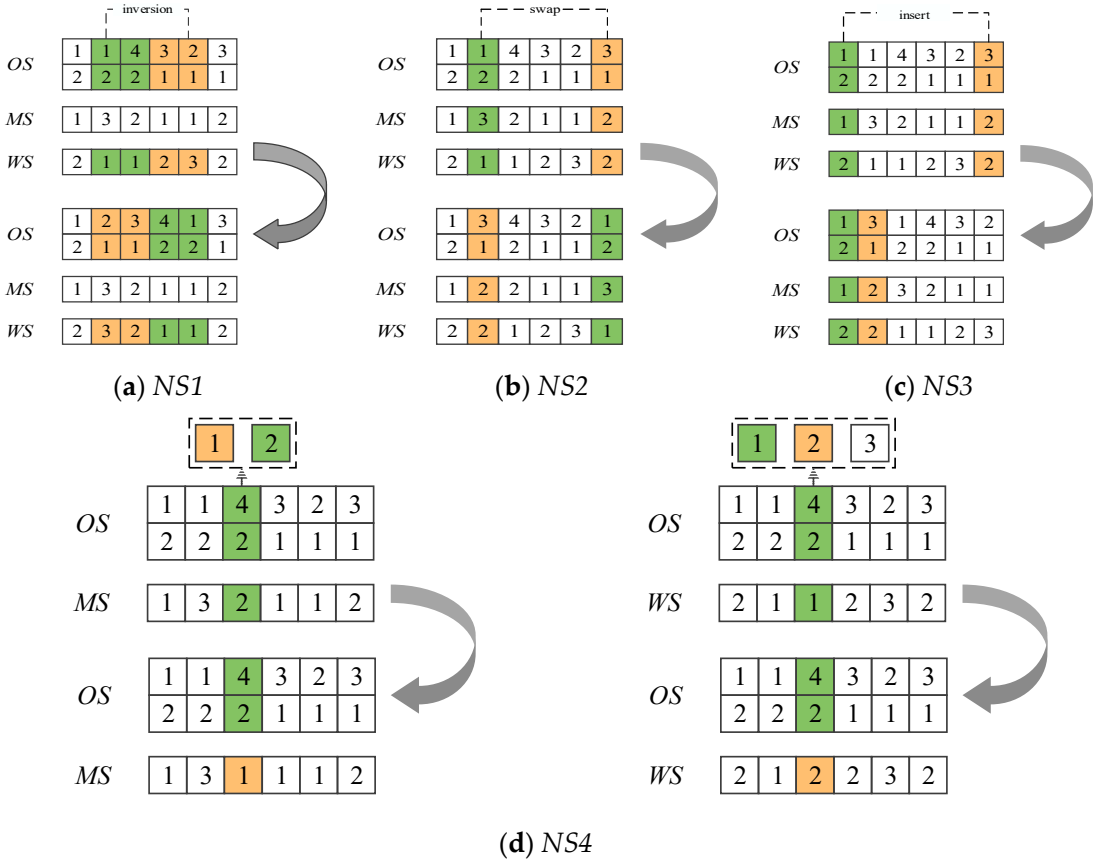
Step 4: Reduce the level of workers on non-critical paths to reduce costs, as shown in Figure 5c.

4.4.2. Local Search

In the HMABC algorithm, the SA is embedded to enhance the local search capability of the HMABC. In this section, two local search strategies are designed, and each move of the SA is equivalent to generating a neighborhood solution around the current solution.

(1) Generate solutions better than the current search strategy. This strategy has a higher probability of finding better solutions and includes two neighborhood structures, *NS1* and *NS2*, but it may also lead the algorithm to fall into local optimal.

NS1: In the OS vector, select two different positions and reverse the genes between the two positions one by one, as illustrated in Figure 6a.



(**a**) *NS1* (**b**) *NS2* (**c**) *NS3*

(**d**) *NS4*

Figure 6. Four neighborhood methods.

NS2: Randomly select two different positions in the OS, exchange the genes at the two positions, and exchange the corresponding genes in the WS at the corresponding positions, as illustrated in Figure 6b.

(2) Random jump-out strategy. This strategy is similar to mutation operation, and executing it with a small probability can help the algorithm escape from local optima. It includes two neighborhood structures, *NS3* and *NS4*.

*NS3***:** For food source *P*, an element is randomly selected and inserted into another random position in OS. At the same time, determine whether the two selected operations belong to the same group. If that is the case, the product vector will not be changed; otherwise, change, as illustrated in Figure 6c.

*NS4***:** This neighborhood structure involves the reallocation of machines and reallocation of workers. The reallocation of machines is performed to OPR, while the reallocation of workers aims to reselect from the optional worker set, as illustrated in Figure 6d.

The flowchart of the local search is shown in Figure 7.

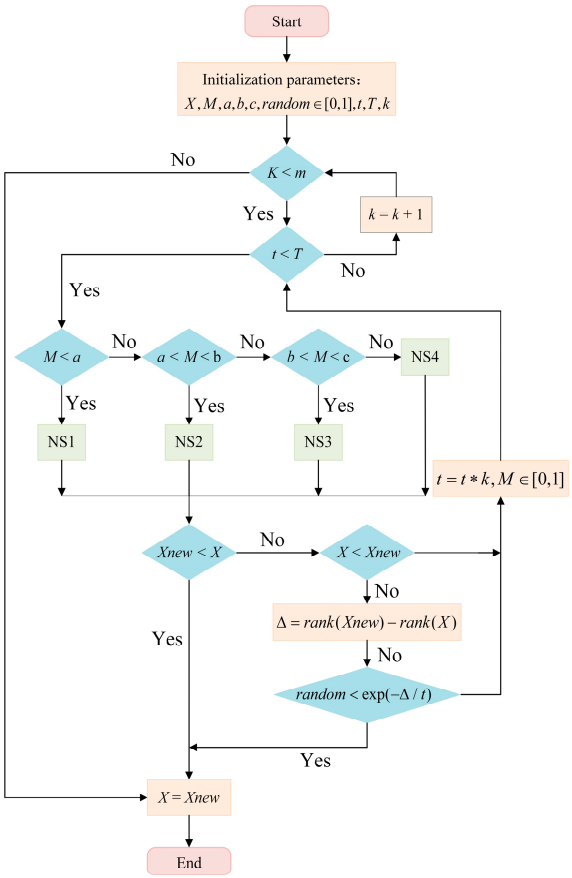


Figure 7. The flowchart of local search.

5. Numerical Experiments and Results

Test examples include three dimensions: the number of jobs, number of machine tools and number of workers, where the number of jobs $n \in \{10, 20, 30, 50, 100\}$, number of machine tools $m \in \{5, 7\}$, and number of workers $wn \in \{3, 5, 7\}$. If a calculation example contains 10 jobs, 5 machine tools, and 3 workers, the calculation example is abbreviated as "10_5_3". One of each dimension is selected for full arrangement, and a total of $5 \times 2 \times 3 = 30$ examples are obtained. In the first stage, processing time $P_{j,l,i,w} \in [10, 30]$, transportation time $T_{j,k,i} \in [5, 15]$, preparation time $S_{i,h,j} \in [15, 20]$, assembly stage, assembly time $A_p \in [0, 20)$, preparation time $SP_{p,s} \in [5, 25)$. Each example divides workers into three levels. The higher the level, the higher the proficiency level Pei and w, and the higher the corresponding worker cost $PC_{j,l,i}$ and w. It is worth noting that the proficiency level less than, equal to and greater than 1.0 indicates that the worker can operate the machine skillfully, generally and unskillfully, respectively. Level 1 worker's proficiency $Pe_{i,w} \in [0.5, 1]$, worker's cost $PC_{j,l,i,w} \in [30, 45]$, level 2 $Pe_{i,w} = 1$, worker's cost $PC_{j,l,i,w} \in [16, 30]$, level 3 $Pe_{i,w} \in (1, 1.5)$, worker cost $PC_{j,l,i,w} \in [10, 15]$. In order to ensure fairness, each calculation example is run independently 10 times, and the stopping standard of CPU = 30 s is set uniformly. The parameters of the algorithm are set as shown in Table 2.

Table 2. Parameters and values.

Parameters	Values
Population size (PS)	200
Limited iteration number (Ln)	5
Probability of mutation operator (PMO)	0.5
LimitForScout	3

To prove the performance of the HMABC in solving the FAJSPLE, the iterative Generational Distance (GD), Inverted Generational Distance (IGD) and C-Metric were used.

GD measures the distance between the approximate solution set PF_{known} generated by the algorithm and the true Pareto optimal solution set PF_{true}. The smaller GD is, the closer the solution set generated by the proof algorithm is to the real Pareto optimal solution set, and the better the algorithm performance. GD calculation Formula (16) is as follows:

$$GD(P, P^*) = \frac{1}{|P|} \sum_{x \in p} \min\{dis(x, y) | y \in P^*\}^{1/m} \tag{16}$$

IGD calculates the average distance between the individuals in solution sets P^* and P; dis(x, P) is the minimum Euclidean distance from the individual to population P.

$$IGD(P^*, P) = \frac{\sum_{x \in P^*} \min dis(x, P)}{|P^*|} \tag{17}$$

The C-Metric measures the PF_{known} domination ratio obtained by the two algorithms, and $C(P, P^*)$ calculates the ratio of P domination to P^*. $C(P, P^*) = 0$, indicating that the solution in P cannot dominate any solution in P^*, $C(P, P^*) = 1$, and at least one solution in A dominates all solutions in P^*, so the larger $C(P, P^*)$, the better the solution in P is proved. The C-Metric Formula (18) is as follows:

$$C(P, P^*) = \frac{|y \in P^* | \exists x \in P : x \prec y|}{|P^*|} \tag{18}$$

5.1. Experimental Parameters

The key parameters of the HMABC algorithm include the population size (PS) and limited iteration number (Ln), in which PS has five levels of {50, 100, 150, 200, 300}, Ln has four levels of {5, 10, 15, 20}, and all parameter combinations are $5 \times 4 = 20$. Furthermore, according to the parameter settings of [46], the probability of the mutation operator is set to 0.5 and the LimitForScout parameter of the scout bee is set to 3. Table 3 lists the parameter combinations and average IGD values (Avg_IGD), Table 4 shows the average IGD of each parameter, and Figure 7 shows the parameter correction results. Figure 8 shows that PS = 200 and Ln = 5.

Table 3. Orthogonal table.

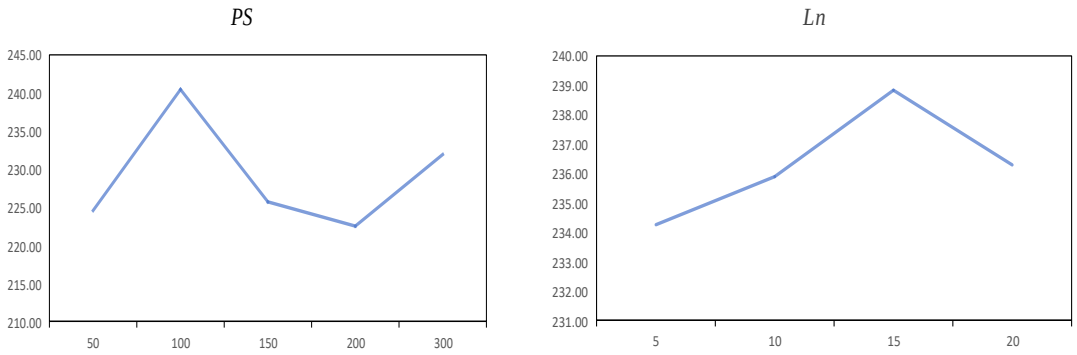
Experiment Number	Parameters		Avg_IGD
	PS	***Ln***	
1	50	5	257.893
2	50	10	272.127
3	50	15	256.877
4	50	20	201.632
6	100	10	221.75
7	100	15	219.505
8	100	20	246.454
9	150	5	210.531
10	150	10	252.647
11	150	15	240.202
12	150	20	255.994
13	200	5	212.966
14	200	10	197.816

Table 3. *Cont.*

Experiment Number	Parameters		Avg_IGD
	PS	***Ln***	
15	200	15	244.403
16	200	20	234.862
17	300	5	216.865
18	300	10	235.11
19	300	15	233.175
20	300	20	242.503

Table 4. The average IGD values of each parameter.

Parameters	Values	Avg_IGD
PS	50	224.5600
	100	240.4523
	150	225.6937
	200	222.5118
	300	231.9133
Ln	5	234.2530
	10	235.8900
	15	238.8324
	20	236.2890



(**a**) *PS* trend (**b**) *Ln* trend

Figure 8. Factor level trend of parameters in HMABC.

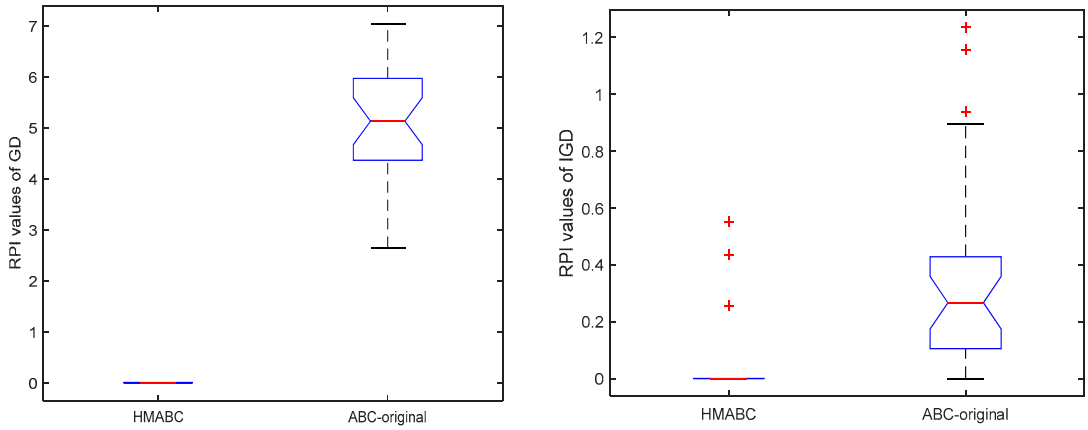
5.2. Effectiveness of Each Improvement of the HMABC

In order to evaluate the performance of the HMABC algorithm, a corresponding comparison algorithm was designed to verify the effectiveness of initialization, mutation and local search. The same test examples were used for each algorithm, and the experimental results were collected for comparison and analysis. In order to study the performance of the HMABC initialization strategy, this section designs variant algorithms for the all random strategy, ABC_original. Table 5 shows the running results of GD, IGD, and RPI. ANOVA results are shown in Figure 9. As shown in Table 5, the GD value of the HMABC is lower than that of ABC_original in all calculation examples, and the IGD value of the HMABC is higher than that of ABC_original in 27 calculation examples. As can be seen from the results in Figure 9, the p values of the GD and IGD indexes are much less than 0.01, indicating that there are significant differences between HMABC and ABC_original, and the HMABC initialization strategy has better performance.

Table 5. The results of initialization strategy.

	Instance	GD		IGD		*RPI*			
		HMABC	*ABC_ original*	*HMABC*	*ABC_ original*	*HMABC (GD)*	*ABC_ original(GD)*	*HMABC (IGD)*	*ABC_ original(IGD)*
1	10_5_3	**2.293**	12.155	**141.749**	149.529	0.000	7.036	0.000	1.234
2	10_5_5	**3.039**	16.756	221.260	**177.896**	0.000	5.712	0.435	0.000
3	10_5_7	**3.288**	20.513	**204.594**	335.259	0.000	4.086	0.000	0.488
4	10_7_3	**3.315**	26.641	**140.763**	314.451	0.000	4.362	0.000	0.938
5	10_7_5	**3.224**	21.642	**211.624**	303.665	0.000	4.798	0.000	0.268
6	10_7_7	**4.362**	22.186	**245.825**	365.748	0.000	4.087	0.000	0.506
7	20_5_3	**4.627**	24.809	**254.671**	493.636	0.000	6.089	0.000	0.657
8	20_5_5	**4.406**	25.545	**242.600**	307.695	0.000	5.130	0.000	0.023
9	20_5_7	**4.197**	21.348	**173.637**	261.555	0.000	5.968	0.000	0.175
10	20_7_3	**3.363**	23.839	**199.333**	330.374	0.000	5.012	0.000	0.219
11	20_7_5	**3.524**	21.601	**218.485**	223.567	0.000	2.643	0.000	0.140
12	20_7_7	**3.743**	26.079	**339.711**	399.059	0.000	5.835	0.000	1.156
13	30_5_3	**5.754**	34.594	589.309	**483.425**	0.000	3.968	0.551	0.000
14	30_5_5	**5.248**	19.120	**387.378**	441.608	0.000	5.127	0.000	0.292
15	30_5_7	**4.078**	27.874	**321.487**	693.079	0.000	5.951	0.000	0.272
16	30_7_3	**8.625**	42.852	**524.120**	812.755	0.000	4.631	0.000	0.200
17	30_7_5	**6.790**	41.598	**561.952**	726.280	0.000	5.307	0.000	0.225
18	30_7_7	**6.119**	42.531	**635.945**	809.005	0.000	6.809	0.000	0.410
19	50_5_3	**9.050**	50.956	**642.789**	771.257	0.000	5.064	0.000	0.894
20	50_5_5	**8.395**	52.945	**918.886**	1125.780	0.000	4.592	0.000	0.320
21	50_5_7	**7.714**	60.235	**949.429**	1338.860	0.000	6.154	0.000	0.324
22	50_7_3	**9.642**	58.471	**725.451**	1374.210	0.000	5.974	0.000	0.105
23	50_7_5	**5.931**	33.167	**717.617**	947.390	0.000	2.695	0.000	0.171
24	50_7_7	**8.947**	64.009	**987.974**	1308.000	0.000	6.066	0.000	0.026
25	100_5_3	**11.326**	78.993	1256.360	**1137.180**	0.000	4.720	0.255	0.000
26	100_5_5	**8.755**	32.349	**1236.230**	1055.730	0.000	4.199	0.000	0.393
27	100_5_7	**10.990**	77.661	**1747.130**	1792.670	0.000	4.018	0.000	0.265
28	100_7_3	**15.061**	86.152	**1638.410**	3367.560	0.000	5.736	0.000	0.084
29	100_7_5	**9.608**	49.953	**1414.030**	1970.400	0.000	5.141	0.000	0.428
30	100_7_7	**10.760**	53.987	**1393.450**	1763.300	0.000	6.229	0.000	0.014

Note: best values in **bold**.



(**a**) the results of GD (**b**) the results of IGD

Figure 9. The comparison results of initialization strategy.

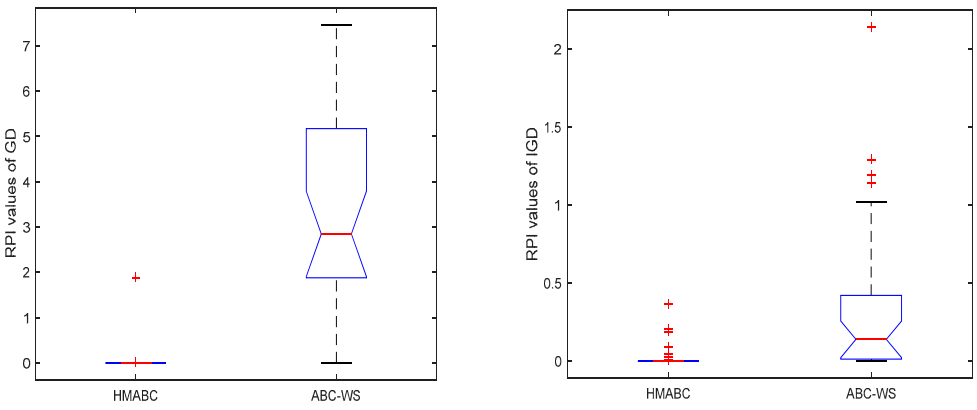
In order to test the effect of the mutation operation based on the critical path, the ABC_WS algorithm is designed to compare with the HMABC. The GD, IGD and RPI values of the HMABC and ABC_WS are shown in Table 6, and the ANOVA analysis results are

shown in Figure 10. The results show that the HMABC is significantly better than ABC_WS, so it can be concluded that the mutation operation based on the critical path is effective for the optimization target and can improve the overall efficiency of the algorithm.

Table 6. The results of mutation strategy.

	Instance	GD		IGD		*RPI*			
		HMABC	*ABC_WS*	*HMABC*	*ABC_WS*	*HMABC (GD)*	*ABC_WS (GD)*	*HMABC (IGD)*	*ABC_WS (IGD)*
1	10_5_3	**3.32**	28.07	**140.76**	442.33	0.000	7.466	0.000	2.142
2	10_5_5	13.60	**13.51**	**441.21**	502.06	0.007	0.000	0.000	0.138
3	10_5_7	**4.36**	13.60	376.33	**316.66**	0.000	2.118	0.188	0.000
4	10_7_3	**5.32**	18.30	**633.73**	900.57	0.000	2.440	0.000	0.421
5	10_7_5	**4.41**	13.45	772.89	**753.14**	0.000	2.052	0.026	0.000
6	10_7_7	**4.20**	22.60	**527.81**	660.15	0.000	4.385	0.000	0.251
7	20_5_3	**3.36**	22.70	**199.33**	402.30	0.000	5.750	0.000	1.018
8	20_5_5	**3.52**	14.40	**241.67**	376.72	0.000	3.086	0.000	0.559
9	20_5_7	**3.74**	16.43	**350.70**	355.06	0.000	3.390	0.000	0.012
10	20_7_3	**5.75**	16.57	**943.87**	1051.59	0.000	1.880	0.000	0.114
11	20_7_5	**5.25**	36.59	**387.38**	519.01	0.000	5.971	0.000	0.340
12	20_7_7	**4.08**	25.16	**321.49**	688.74	0.000	5.170	0.000	1.142
13	30_5_3	**8.62**	22.24	1257.16	**1045.34**	0.000	1.579	0.203	0.000
14	30_5_5	**6.79**	17.52	797.36	**732.68**	0.000	1.581	0.088	0.000
15	30_5_7	**6.12**	25.14	**650.45**	870.34	0.000	3.109	0.000	0.338
16	30_7_3	**9.50**	27.08	**2619.64**	2934.16	0.000	1.851	0.000	0.120
17	30_7_5	**8.39**	26.80	**1124.07**	1270.55	0.000	2.193	0.000	0.130
18	30_7_7	**7.71**	40.68	**949.43**	1464.85	0.000	4.274	0.000	0.543
19	50_5_3	**9.64**	40.91	2317.16	**1698.02**	0.000	3.243	0.365	0.000
20	50_5_5	**5.93**	42.19	**743.66**	1704.86	0.000	6.113	0.000	1.293
21	50_5_7	**8.95**	31.92	**987.97**	1156.88	0.000	2.567	0.000	0.171
22	50_7_3	**11.33**	30.27	**1370.16**	1561.69	0.000	1.673	0.000	0.140
23	50_7_5	**8.76**	63.63	**2134.78**	2788.95	0.000	6.267	0.000	0.306
24	50_7_7	**10.99**	76.96	1747.13	**1734.35**	0.000	6.003	0.007	0.000
25	100_5_3	41.99	**14.57**	**4443.15**	5164.68	1.882	0.000	0.000	0.162
26	100_5_5	**9.61**	59.63	**1414.03**	3099.30	0.000	5.206	0.000	1.192
27	100_5_7	**10.76**	41.06	**1578.73**	1662.72	0.000	2.816	0.000	0.053
28	100_7_3	**19.51**	75.68	**3849.80**	4032.16	0.000	2.880	0.000	0.047
29	100_7_5	**17.35**	43.03	**3178.86**	3520.00	0.000	1.480	0.000	0.107
30	100_7_7	**15.82**	46.03	2882.50	**2766.84**	0.000	1.910	0.042	0.000

Note: best values in **bold**.



(**a**) the results of GD (**b**) the results of IGD

Figure 10. The comparison results of mutation strategy.

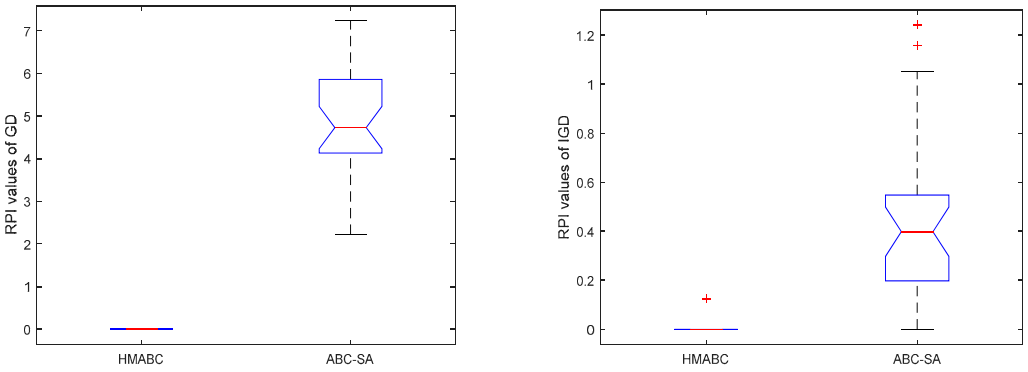
In the HMABC algorithm, the SA algorithm containing four kinds of neighborhood structures is introduced to improve the local search ability of the algorithm. In order to verify the performance of the embedded SA when the HMABC is used to solve the FAJSPLE problem, this section designs the ABC_SA algorithm, and it collects GD, IGD and RPI values and puts them in Table 7. The ANOVA is drawn as shown in Figure 11. By analyzing the comparison results, the embedded improved SA algorithm can improve the performance of the HMABC, which is feasible.

Table 7. The results of local search strategy.

	Instance	GD		IGD		RPI			
		HMABC	*ABC_SA*	*HMABC*	*ABC_SA*	*HMABC(GD)*	*ABC_SA (GD)*	*HMABC(IGD)*	*ABC_SA (IGD)*
1	10_5_3	**3.32**	27.29	**140.76**	315.71	0.000	4.330	0.000	1.243
2	10_5_5	**3.22**	24.12	**261.12**	303.15	0.000	4.238	0.000	0.161
3	10_5_7	**4.36**	20.65	**270.13**	365.75	0.000	5.043	0.000	0.354
4	10_7_3	**4.63**	24.21	**221.58**	436.85	0.000	7.233	0.000	0.972
5	10_7_5	**4.41**	24.77	**194.68**	309.63	0.000	6.480	0.000	0.590
6	10_7_7	**4.20**	18.90	**234.26**	337.21	0.000	3.733	0.000	0.439
7	20_5_3	**3.36**	23.07	**199.33**	302.96	0.000	4.233	0.000	0.520
8	20_5_5	**3.52**	20.09	**218.49**	286.10	0.000	4.622	0.000	0.309
9	20_5_7	**3.74**	23.79	**339.71**	382.57	0.000	3.504	0.000	0.126
10	20_7_3	**5.75**	33.15	**424.50**	589.31	0.000	5.862	0.000	0.388
11	20_7_5	**5.25**	24.39	**387.38**	466.23	0.000	4.700	0.000	0.204
12	20_7_7	**4.08**	27.61	**321.49**	659.56	0.000	5.356	0.000	1.052
13	30_5_3	**8.62**	44.98	**376.44**	812.76	0.000	4.761	0.000	1.159
14	30_5_5	**6.79**	39.91	**491.95**	724.75	0.000	3.647	0.000	0.473
15	30_5_7	**6.12**	42.57	**635.95**	761.86	0.000	5.770	0.000	0.198
16	30_7_3	**9.05**	46.39	**662.81**	1025.79	0.000	4.215	0.000	0.548
17	30_7_5	**8.39**	55.41	**901.06**	1030.43	0.000	4.878	0.000	0.144
18	30_7_7	**7.71**	60.05	**949.43**	1388.04	0.000	5.958	0.000	0.462
19	50_5_3	**9.64**	59.67	**774.61**	1420.97	0.000	4.126	0.000	0.834
20	50_5_5	**5.93**	30.45	**717.62**	1096.05	0.000	5.600	0.000	0.527
21	50_5_7	**8.95**	64.08	**987.97**	1228.82	0.000	6.785	0.000	0.244
22	50_7_3	**11.33**	78.91	**1082.87**	1335.06	0.000	5.189	0.000	0.233
23	50_7_5	**8.76**	28.13	1204.70	**1072.93**	0.000	4.134	0.123	0.000
24	50_7_7	**10.99**	77.36	**1747.13**	1928.21	0.000	6.163	0.000	0.104
25	100_5_3	**15.06**	83.55	**1550.98**	2851.96	0.000	5.967	0.000	0.839
26	100_5_5	**9.61**	46.16	**1414.03**	2038.97	0.000	2.214	0.000	0.442
27	100_5_7	**10.76**	46.56	**1393.45**	1816.44	0.000	6.039	0.000	0.304
28	100_7_3	**19.51**	132.06	**3849.80**	3851.80	0.000	4.548	0.000	0.066
29	100_7_5	**17.35**	108.95	**3154.11**	3158.11	0.000	3.805	0.000	0.409
30	100_7_7	**15.82**	117.16	**2397.02**	2610.24	0.000	3.327	0.000	0.089

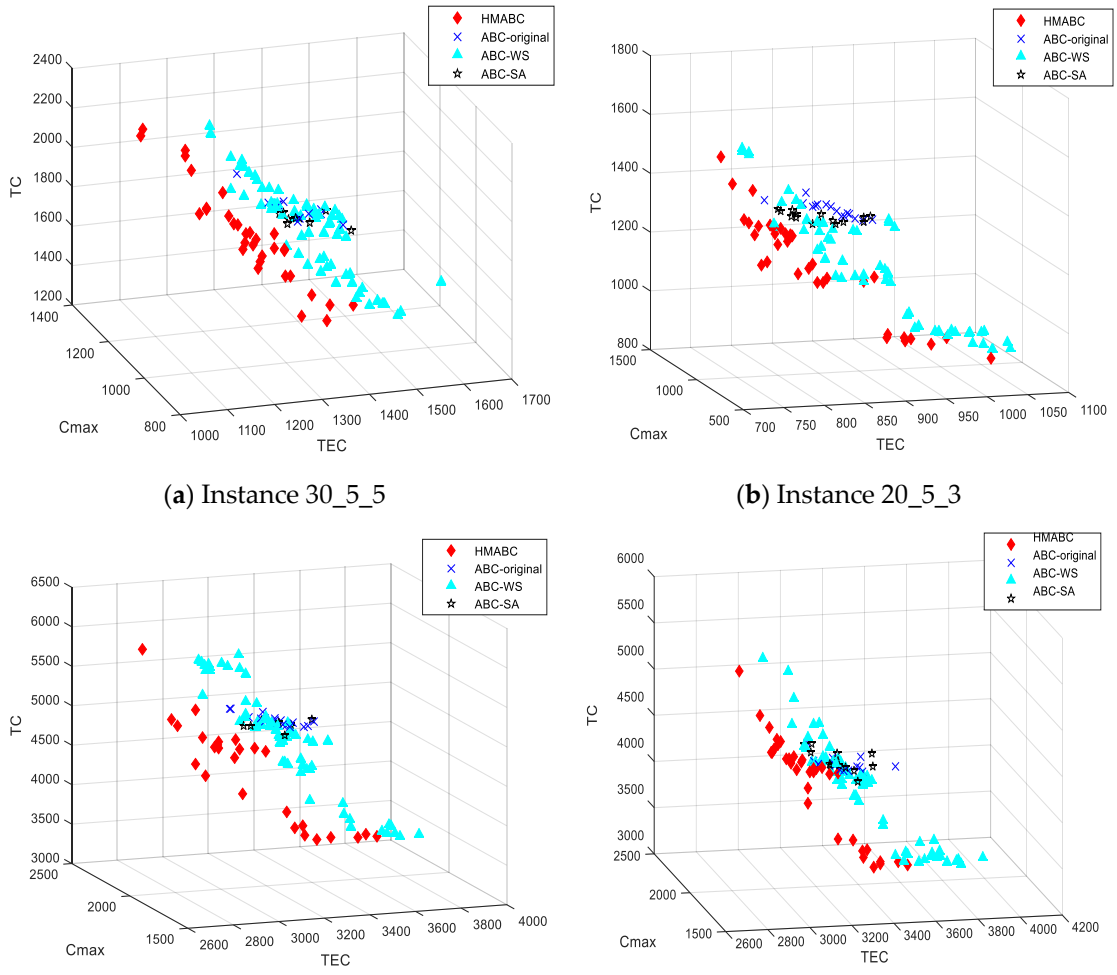
Note: best values in **bold.**

In order to more intuitively reflect the differences between the three comparison algorithms and the HMABC algorithm, four examples of Pareto frontier are given in Figure 12.



(**a**) the results of GD (**b**) the results of IGD

Figure 11. The comparison results of local search strategy.



(**a**) Instance 30_5_5 (**b**) Instance 20_5_3

(**c**) Instance 50_7_3 (**d**) Instance 50_7_5

Figure 12. Pareto results of instances.

5.3. Comparative Analysis with Other Algorithms

To further verify the effectiveness of the HMABC algorithm in solving the FAJSPLE problem, four algorithms were selected: namely, PSO_GA [49], MOGA [50], EGA [19] and memetic NSGA-II [53]. In order to make a fair comparison, the four comparison algorithms are consistent with the HMABC in the aspects of the partial initialization strategy, encoding, decoding and test examples. The results of GD, IGD and the C-Metric are shown in

Tables 8 and 9. According to the data in the table, the following can be concluded: (1) for the GD and IGD values, the HMABC is significantly superior to the other comparison algorithms in the 30 test examples; (2) for the C-Metric, the HMABC obtained 26, 22, 27 and 30 dominant results, respectively.

Table 8. The results of GD and IGD.

Instance	GD					IGD				
	HMABC	**PSO_GA**	**MOGA**	**EGA**	**NSGA-II**	**HMABC**	**PSO_GA**	**MOGA**	**EGA**	**NSGA-II**
10_5_3	**4.37**	22.80	25.39	24.12	23.72	349.94	263.98	**205.77**	324.28	368.96
10_5_5	**2.79**	22.17	20.76	23.49	21.95	**144.38**	359.31	296.16	415.14	397.65
10_5_7	**3.22**	19.97	19.45	17.34	20.06	**213.79**	324.16	255.99	336.53	453.64
10_7_3	**3.11**	25.94	21.81	26.75	26.07	**143.64**	374.68	278.35	368.48	503.63
10_7_5	**4.36**	33.27	32.51	36.03	34.82	**401.59**	825.35	605.05	942.28	1023.85
10_7_7	**4.53**	30.05	28.34	31.64	30.26	440.87	426.78	**378.72**	555.72	624.34
20_5_3	**2.82**	19.43	19.38	21.54	19.41	**166.83**	270.16	256.88	300.27	335.54
20_5_5	**3.36**	15.99	14.46	17.75	15.84	310.63	279.25	268.53	**253.77**	272.18
20_5_7	**4.18**	22.95	23.36	23.08	23.14	305.85	298.31	260.88	**258.91**	407.85
20_7_3	**5.34**	41.21	38.64	41.37	39.43	**532.27**	864.07	727.88	901.42	883.44
20_7_5	**4.00**	38.12	35.51	39.18	35.77	**330.53**	873.60	722.42	926.32	929.42
20_7_7	**5.03**	38.23	36.00	39.20	38.57	**354.79**	611.49	491.04	620.85	669.22
30_5_3	**6.51**	51.81	48.21	51.11	48.81	**607.98**	985.44	795.49	958.26	1169.06
30_5_5	**4.79**	42.47	40.77	43.45	42.23	**245.48**	577.69	500.30	617.93	625.39
30_5_7	**6.04**	54.26	50.85	54.42	54.82	**650.77**	1544.43	1199.88	1551.18	1604.49
30_7_3	**6.39**	48.06	45.83	48.72	46.88	**766.70**	1623.89	1315.66	1571.76	1678.64
30_7_5	**7.76**	67.57	64.05	68.32	64.23	**1255.27**	2659.39	2315.85	2643.06	2733.84
30_7_7	**7.43**	67.31	65.36	67.43	68.56	**513.91**	1062.29	985.53	1038.10	1130.54
50_5_3	**8.06**	71.43	70.67	71.21	72.34	**1009.00**	2276.26	2024.25	2142.57	2243
50_5_5	**7.14**	58.90	57.35	59.57	58.52	**852.99**	1815.14	1657.73	1929.28	1874.77
50_5_7	**8.72**	69.13	67.21	70.36	65.14	**1087.23**	1999.11	1824.04	1962.75	1917.96
50_7_3	**12.51**	91.45	89.40	91.94	91.96	**1846.28**	2635.69	2522.75	2595.38	2667.69
50_7_5	**6.40**	92.13	89.21	92.26	92.22	**470.16**	2872.91	2726.18	2871.87	2934.8
50_7_7	**9.09**	106.53	104.07	104.89	105.56	**1218.31**	4302.17	4136.74	4226.80	4363.43
100_5_3	**15.07**	110.85	109.26	110.78	110.73	**2491.30**	3585.06	3196.22	3318.53	3361.83
100_5_5	**11.31**	103.95	100.41	102.06	101.39	**1708.33**	3827.34	3521.87	3566.03	3860.15
100_5_7	**10.89**	108.18	108.36	109.14	108.07	**1269.50**	3328.00	3261.61	3259.70	3398.42
100_7_3	**17.33**	153.63	150.95	152.35	149.63	**2689.83**	5169.45	5030.77	5142.87	5247.48
100_7_5	**12.31**	128.20	126.65	128.31	130.8	**1328.98**	3919.76	3840.55	3855.55	3923.66
100_7_7	**16.36**	143.45	142.54	142.48	144.14	**3475.67**	6744.54	6914.76	6670.63	6778.14
Average	7.37	63.31	61.56	63.68	62.84	906.09	1889.99	1750.59	1870.87	1946.10

Note: best values in **bold.**

Table 9. The results of C-Metric.

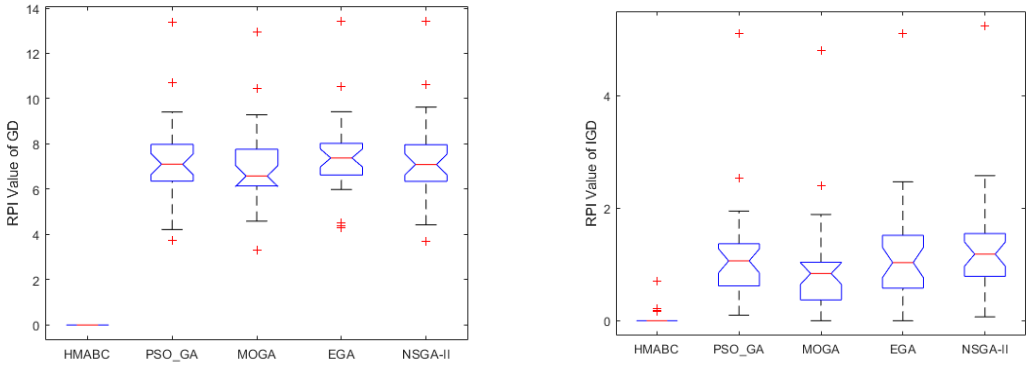
Instance	*C*(*HMABC*, PSO_GA)	*C*(PSO_G, *HMABC*)	*C*(*HMABC*, MOGA)	*C*(MOGA, *HMABC*)	*C*(*HMAC*, EGA)	*C*(EGA, *HMABC*)	*C*(*HMAB*, HDMICA)	*C*(HDMIC, *HMABC*)
10_5_3	**0.8429**	0.0053	**1.0000**	0.0000	**0.2000**	0.0000	**0.0333**	0.0105
10_5_5	0.0250	**0.0267**	0.1000	**0.1512**	**0.2000**	0.0000	**0.1222**	0.0133
10_5_7	**1.0000**	0.0000	**0.2000**	0.0875	**0.2000**	0.0000	**0.1200**	0.0412
10_7_3	**0.2000**	0.1125	**0.2000**	0.1188	**0.2000**	0.0000	**0.1429**	0.0118
10_7_5	**0.2000**	0.0846	**0.2000**	0.1077	**0.2000**	0.0000	**0.1750**	0.0000
10_7_7	**0.1636**	0.1371	0.0741	**0.1371**	**0.1571**	0.0000	**0.1400**	0.0000
20_5_3	**0.1846**	0.0000	**0.1500**	0.0000	**0.2000**	0.0000	**0.0571**	0.0222
20_5_5	**0.1294**	0.1231	0.1053	**0.1231**	**0.0667**	0.0148	**0.1800**	0.0148
20_5_7	**0.1846**	0.1500	0.1053	**0.1500**	**0.1818**	0.0000	**0.1077**	0.0308
20_7_3	**0.1474**	0.1152	0.0240	**0.1212**	**0.1565**	0.0000	**0.0842**	0.0343
20_7_5	**0.2000**	0.1086	**0.1875**	0.1486	**0.2000**	0.0000	**0.1474**	0.0000
20_7_7	0.0857	**0.1091**	**0.2000**	0.1182	**0.1909**	0.0000	**0.1333**	0.0182
30_5_3	**0.2000**	0.0308	**0.0889**	0.0769	**0.1895**	0.0000	**0.1250**	0.0074
30_5_5	**0.2000**	0.0600	**0.1333**	0.1200	0.0207	**0.0625**	**0.0889**	0.0188

Table 9. *Cont.*

Instance	*C*(*HMABC*, PSO_GA)	*C*(PSO_G, *HMABC*)	*C*(*HMABC*, MOGA)	*C*(MOGA, *HMABC*)	*C*(*HMAC*, EGA)	*C*(EGA, *HMABC*)	*C*(*HMAB*, HDMICA)	*C*(HDMIC, *HMABC*)
30_5_7	0.0313	**0.0865**	**0.2000**	0.1135	**0.2000**	0.0000	**0.1077**	0.0059
30_7_3	**0.2000**	0.0533	**0.1846**	0.1333	0.0000	**0.1056**	**0.1571**	0.0056
30_7_5	**0.0541**	0.0286	1.0000	0.0000	**0.1500**	0.0000	**0.1158**	0.0000
30_7_7	**0.0118**	0.1000	**0.2000**	0.1273	**0.2000**	0.0000	**0.0750**	0.0222
50_5_3	0.0400	0.0500	**0.2000**	0.1111	**0.1500**	0.0000	**0.1200**	0.0182
50_5_5	**0.0632**	0.0444	**0.2000**	0.1000	**0.2000**	0.0000	**0.0714**	0.0429
50_5_7	**0.1130**	0.0714	**0.2000**	0.1286	**0.2000**	0.0000	**0.1500**	0.0333
50_7_3	**0.1188**	0.0000	**0.2000**	0.0759	**0.1714**	0.0000	**0.1429**	0.0077
50_7_5	**0.1313**	0.0211	1.0000	0.0000	**0.2000**	0.0000	**0.0667**	0.0571
50_7_7	**0.1034**	0.0071	**0.2000**	0.0429	**0.2000**	0.0000	**0.0824**	0.0158
100_5_3	**0.0706**	0.0400	**0.2000**	0.0467	0.0000	**0.0848**	**0.1412**	0.0000
100_5_5	**0.1100**	0.0129	**0.2000**	0.0710	**0.2000**	0.0000	**0.1833**	0.0150
100_5_7	**0.1455**	0.0074	**0.2000**	0.0741	**0.2000**	0.0000	**0.0500**	0.0182
100_7_3	**0.1565**	0.0056	**0.2000**	0.0278	**0.2000**	0.0000	**0.1556**	0.0000
100_7_5	**0.1209**	0.0364	**0.2000**	0.0485	**0.2000**	0.0000	**0.1556**	0.0057
100_7_7	**0.0909**	0.0200	1.0000	0.0000	**0.2000**	0.0000	**0.1333**	0.0148

Note: best values in **bold.**

In order to prove the significance of differences between the HMABC and the four comparison algorithms, variance analysis was performed on the GD and IGD values of the algorithms, respectively. As shown in Figure 13, the P-values are all less than 0.01. In addition, the Pareto frontier results for four examples are given in Figure 14. In summary, it can be concluded that the HMABC has better performance than the other algorithms.

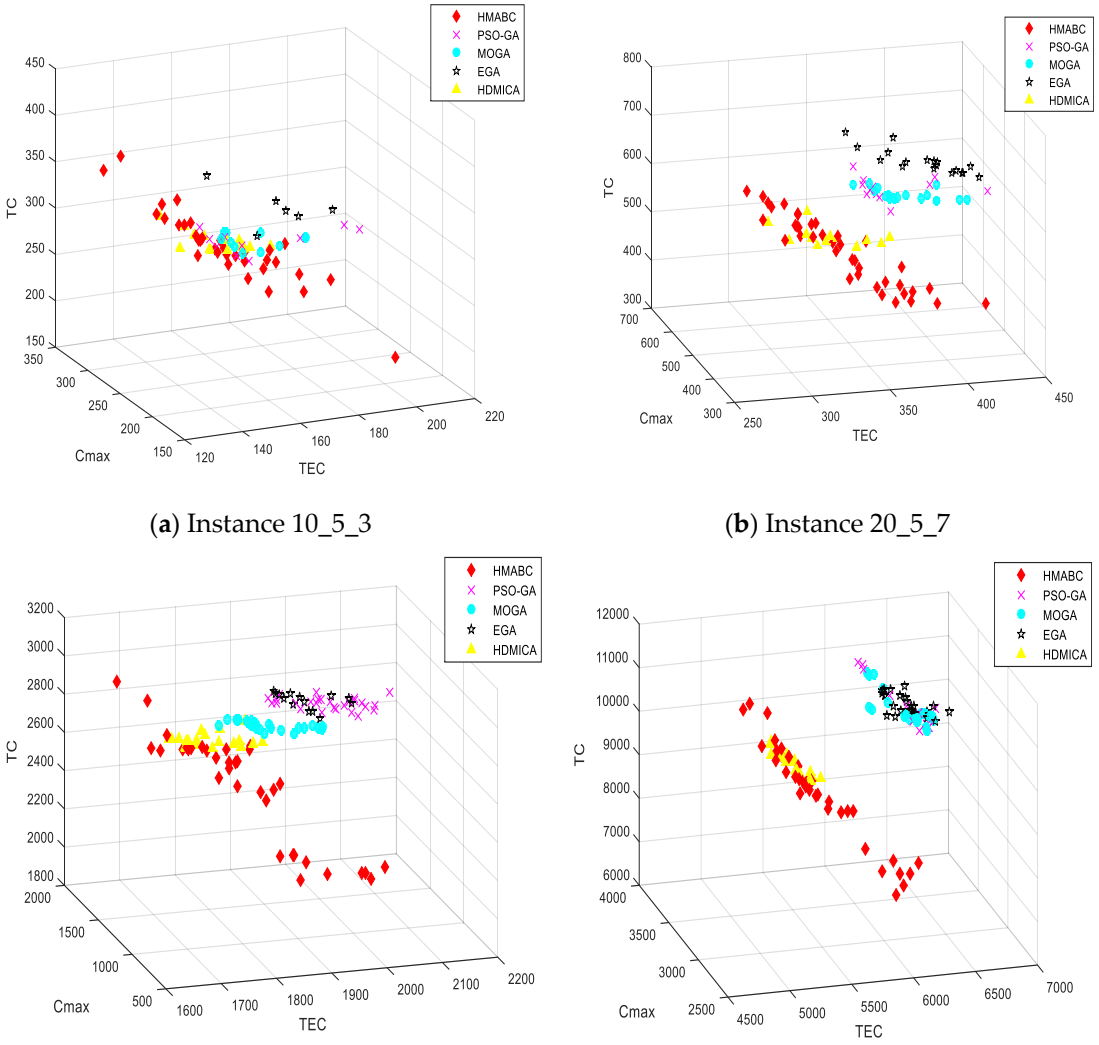


(**a**) RPI of GD values (**b**) RPI of IGD values

Figure 13. ANOVA results of algorithms comparison.

5.4. Experimental Analysis

The results show that the HMABC can effectively solve the FAJSPLE. The favorable performance of the HMABC mainly results from (1) the effective encoding and decoding mechanism, and it is necessary to take advantage of the problem-specific characteristics for developing two kinds of knowledge in the decoding stage; (2) the initialization phase generates effective 3D encoding schemes to improve the quality of the initial population; (3) the employed bee phase executes well-designed crossover and mutation operations to enhance the search capacity of the algorithm; (4) the SA algorithm is embedded into the HMABC algorithm to reduce the possibility of falling into the local optimal.



(**a**) Instance 10_5_3 (**b**) Instance 20_5_7

(**c**) Instance 30_7_5 (**d**) Instance 100_7_3

Figure 14. Pareto results of multi-algorithm comparison.

6. Conclusions and Future Research Directions

In this article, the HMABC is employed to address the FAJSPLE. The simultaneous optimization of the makespan, TEC and TC is achieved by building a mathematical model of the FAJSPLE. According to the characteristics of the problem, a critical path-based mutation strategy is designed, aiming at dynamically adjusting the proficiency of workers and minimally avoiding the waste of resources. In order to avoid falling into the local optimum, a local search algorithm is designed by combining the SA and four neighborhood search operators. The introduction of this neighborhood search strategy in the HMABC makes the algorithm always find the optimal solution. In contrast, the mutation strategy proposed in this paper allows the algorithm to obtain a more efficient solution, and the proposed neighborhood search operator can find the local optimum effectively. Three metrics are used to validate the effectiveness of the HMABC on different sized instances, and the HMABC achieves lower GD, IGD and higher C-Metric values compared to the comparison algorithms.

Further research can concern the following topics: (1) designing better coding methods to improve the overall effectiveness of the algorithms; (2) reducing the complexity of the algorithms to further shorten the running time; and (3) considering combining reinforcement

learning or deep reinforcement learning algorithms to have solved the problems studied, for example, using reinforcement learning algorithms to improve the ability of local search.

Author Contributions: Z.D.: Conceptualization, Methodology, Validation, Formal Analysis, Writing—Original Draft, Writing—Review and Editing, Supervision. J.L. (Junqing Li): Methodology, Validation, Writing—Review and Editing, Supervision, Funding Acquisition. J.L. (Jiake Li): Investigation, Formal Analysis, Visualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the Key projects of Yunnan Province Basic Research Program (202401AS070036), Yunnan Key Laboratory of Modern Analytical Mathematics and Applications (No. 202302AN360007).

Data Availability Statement: Available on request.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Li, X.; Guo, X.; Tang, H.; Wu, R.; Wang, L.; Pang, S.; Liu, Z.; Xu, W.; Li, X. Survey of integrated flexible job shop scheduling problems. *Comput. Ind. Eng.* **2022**, *174*, 108786. [CrossRef]
2. Hansmann, R.-S.; Rieger, T.; Zimmermann, U.-T. Flexible job shop scheduling with blockages. *Math. Methods Oper. Res.* **2014**, *79*, 135–161. [CrossRef]
3. Kress, D.; Müller, D.; Nossack, J. A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. *OR Spectr.* **2019**, *41*, 179–217. [CrossRef]
4. Homayouni, S.M.; Fontes, D.B. Production and transport scheduling in flexible job shop manufacturing systems. *J. Glob. Optim.* **2021**, *79*, 463–502. [CrossRef]
5. Fattahi, P.; Mehrabad, M.S.; Jolai, F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *J. Intell. Manuf.* **2007**, *18*, 331–342. [CrossRef]
6. Wang, J.-F.; Du, B.-Q.; Ding, H.-M. A genetic algorithm for the flexible job-shop scheduling problem. In *Proceedings of the Advanced Research on Computer Science and Information Engineering: International Conference, CSIE 2011, Zhengzhou, China, 21–22 May 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 332–339.
7. Xia, W.; Wu, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Comput. Ind. Eng.* **2005**, *48*, 409–425. [CrossRef]
8. Zhang, S.; Li, X.; Zhang, B.; Wang, S. Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system. *Eur. J. Oper. Res.* **2020**, *283*, 441–460. [CrossRef]
9. Wu, C.-C.; Wang, D.-J.; Cheng, S.-R.; Chung, I.-H.; Lin, W.-C. A two-stage three-machine assembly scheduling problem with a position-based learning effect. *Int. J. Prod. Res.* **2018**, *56*, 3064–3079. [CrossRef]
10. Kazemi, H.; Mazdeh, M.M.; Rostami, M. The two stage assembly flow-shop scheduling problem with batching and delivery. *Eng. Appl. Artif. Intell.* **2017**, *63*, 98–107. [CrossRef]
11. Mozdgir, A.; Ghomi, S.F.; Jolai, F.; Navaei, J. Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times. *Int. J. Prod. Res.* **2013**, *51*, 3625–3642. [CrossRef]
12. Navaei, J.; Ghomi, S.; Jolai, F.; Shiraqai, M.; Hidaji, H. Two-stage flow-shop scheduling problem with non-identical second stage assembly machines. *Int. J. Adv. Manuf. Technol.* **2013**, *69*, 2215–2226. [CrossRef]
13. Fattahi, P.; Hosseini, S.M.H.; Jolai, F. A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2013**, *65*, 787–802. [CrossRef]
14. Liao, C.-J.; Lee, C.-H.; Lee, H.-C. An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Comput. Ind. Eng.* **2015**, *88*, 317–325. [CrossRef]
15. Chan, F.; Wong, T.; Chan, L. Lot streaming for product assembly in job shop environment. *Robot. Comput. -Integr. Manuf.* **2008**, *24*, 321–331. [CrossRef]
16. Wong, T.-C.; Chan, F.-T.; Chan, L. A resource-constrained assembly job shop scheduling problem with lot streaming technique. *Comput. Ind. Eng.* **2009**, *57*, 983–995. [CrossRef]
17. Zhang, S.; Wang, S. Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. *IEEE Trans. Eng. Manag.* **2018**, *65*, 487–504. [CrossRef]
18. Hurink, J.; Knust, S. Tabu search algorithms for job-shop problems with a single transport robot. *Eur. J. Oper. Res.* **2005**, *162*, 99–111. [CrossRef]

19. Zhang, Q.; Manier, H.; Manier, M.-A. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *Int. J. Prod. Res.* **2014**, *52*, 985–1002. [CrossRef]
20. Rossi, A.; Dini, G. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robot. Comput. -Integr. Manuf.* **2007**, *23*, 503–516. [CrossRef]
21. Zhang, Q.; Manier, H.; Manier, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Comput. Oper. Res.* **2012**, *39*, 1713–1723. [CrossRef]
22. Rossi, A. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *Int. J. Prod. Econ.* **2014**, *153*, 253–267. [CrossRef]
23. Karimi, S.; Ardalan, Z.; Naderi, B.; Mohammadi, M. Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Appl. Math. Model.* **2017**, *41*, 667–682. [CrossRef]
24. Qi, R.; Li, J.-Q.; Wang, J.; Jin, H.; Han, Y.-Y. QMOEA: A Q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows. *Inf. Sci.* **2022**, *608*, 178–201. [CrossRef]
25. Zhang, G.; Hu, Y.; Sun, J.; Zhang, W. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm Evol. Comput.* **2020**, *54*, 100664. [CrossRef]
26. Liu, Z.; Guo, S.; Wang, L. Integrated green scheduling optimization of flexible job shop and crane transportation considering comprehensive energy consumption. *J. Clean. Prod.* **2019**, *211*, 765–786. [CrossRef]
27. Dai, M.; Tang, D.; Giret, A.; Salido, M.A. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robot. Comput. Integr. Manuf.* **2019**, *59*, 143–157. [CrossRef]
28. Azzouz, A.; Ennigrou, M.; Said, L.B. A hybrid algorithm for flexible job-shop scheduling problem with setup times. *Int. J. Prod. Manag. Eng.* **2017**, *5*, 23–30. [CrossRef]
29. Mousakhani, M. Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. *Int. J. Prod. Res.* **2013**, *51*, 3476–3487. [CrossRef]
30. Li, J.-Q.; Deng, J.-W.; Li, C.-Y.; Han, Y.-Y.; Tian, J.; Zhang, B.; Wang, C.-G. An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowl. Based Syst.* **2020**, *200*, 106032. [CrossRef]
31. Saidi-Mehrabad, M.; Fattahi, P. Flexible job shop scheduling with tabu search algorithms. *Int. J. Adv. Manuf. Technol.* **2007**, *32*, 563–570. [CrossRef]
32. Du, Y.; Li, J.-Q.; Chen, X.-L.; Duan, P.-Y.; Pan, Q.-K. Knowledge-Based Reinforcement Learning and Estimation of Distribution Algorithm for Flexible Job Shop Scheduling Problem. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *7*, 1036–1050. [CrossRef]
33. Du, Y.; Li, J.; Li, C.; Duan, P. A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 5695–5709. [CrossRef]
34. Li, Z.; Qian, B.; Hu, R.; Chang, L.; Yang, J. An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups. *Knowl. Based Syst.* **2019**, *173*, 83–112. [CrossRef]
35. Sun, J.; Zhang, G.; Lu, J.; Zhang, W. A hybrid many-objective evolutionary algorithm for flexible job-shop scheduling problem with transportation and setup times. *Comput. Oper. Res.* **2021**, *132*, 105263. [CrossRef]
36. Zou, W.; Chen, Y.Z.; Zhang, B. Solving multiobjective optimization problems using artificial bee colony algorithm. *Discret. Dyn. H. Nat. Soc.* **2011**, *2011*, 569784. [CrossRef]
37. Hedayatzadeh, R.; Hasanizadeh, B.; Akbari, R.; Ziarati, K. A multi-objective artificial bee colony for optimizing multi-objective problems. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China, 20–22 August 2010*; IEEE: New York, NY, USA, 2010; pp. V5-277–V5-281.
38. Kishor, A.; Singh, P.K.; Prakash, J. NSABC: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering. *Neurocomputing* **2016**, *216*, 514–533. [CrossRef]
39. Akay, B. Synchronous and asynchronous pareto-based multi-objective artificial bee colony algorithms. *J. Glob. Optim.* **2013**, *57*, 415–445. [CrossRef]
40. Toktas, A.; Ustun, D.; Erdogan, N. Pioneer Pareto artificial bee colony algorithm for three-dimensional objective space optimization of composite-based layered radar absorber. *Appl. Soft Comput.* **2020**, *96*, 106696. [CrossRef]
41. Li, J.-Q.; Wang, J.-D.; Pan, Q.-K.; Duan, P.-Y.; Sang, H.-Y.; Gao, K.-Z.; Xue, Y. A hybrid artificial bee colony for optimizing a reverse logistics network system. *Soft Comput.* **2017**, *21*, 6001–6018. [CrossRef]
42. Gao, W.-F.; Liu, S.-Y.; Huang, L.-L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Trans. Cybern.* **2013**, *43*, 1011–1024. [PubMed]
43. Shan, J.; Lu, R. Multi-objective economic optimization scheduling of CCHP micro-grid based on improved bee colony algorithm considering the selection of hybrid energy storage system. *Energy Rep.* **2021**, *7*, 326–341. [CrossRef]
44. Chen, H.; Bo, M.L.; Zhu, Y. Multi-hive bee foraging algorithm for multi-objective optimal power flow considering the cost, loss, and emission. *Int. J. Electr. Power Energy Syst.* **2014**, *60*, 203–220. [CrossRef]
45. Li, J.-Q.; Han, Y.-Q. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust. Comput.* **2020**, *23*, 2483–2499. [CrossRef]

46. Alrezaamiri, H.; Ebrahimnejad, A.; Motameni, H. Parallel multi-objective artificial bee colony algorithm for software requirement optimization. *Requir. Eng.* **2020**, *25*, 363–380. [CrossRef]
47. Baradaran, V.; Shafaei, A.; Hosseinian, A.H. Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Comput. Ind. Eng.* **2019**, *131*, 187–199. [CrossRef]
48. Li, Y.; Huang, W.; Wu, R.; Guo, K. An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem. *Appl. Soft Comput.* **2020**, *95*, 106544. [CrossRef]
49. Xie, J.; Gao, L.; Pan, Q.-K.; Tasgetiren, M.F. An effective multi-objective artificial bee colony algorithm for energy efficient distributed job shop scheduling. *Procedia Manuf.* **2019**, *39*, 1194–1203. [CrossRef]
50. Xu, X.; Hao, J.; Zheng, Y. Multi-objective artificial bee colony algorithm for multi-stage resource leveling problem in sharing logistics network. *Comput. Ind. Eng.* **2020**, *142*, 106338. [CrossRef]
51. Azzouz, A.; Ennigrou, M.; Said, L.B. A self-adaptive evolutionary algorithm for solving flexible job-shop problem with sequence dependent setup time and learning effects. In *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017*; IEEE: New York, NY, USA, 2017; pp. 1827–1834.
52. Li, Y.; Li, X.; Gao, L.; Fu, L.; Wang, C. An efficient critical path based method for permutation flow shop scheduling problem. *J. Manuf. Syst.* **2022**, *63*, 344–353. [CrossRef]
53. Burmeister, S.-C.; Guericke, D.; Schryen, G. A memetic NSGA-II for the multi-objective flexible job shop scheduling problem with real-time energy tariffs. *Flex. Serv. Manuf. J.* **2023**, *36*, 1–41.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.