machines

MDPI

Article

A Novel Hybrid Whale Optimization Algorithm for Flexible Job-Shop Scheduling Problem

Wenqiang Yang [1], Jinzhe Su [1], Yunhang Yao [1], Zhile Yang [2,*] and Ying Yuan [1]

1 School of Mechanical and Electrical Engineering, Henan Institute of Science and Technology, Xinxiang 453003, China; yangwqjsj@hist.edu.cn (W.Y.); jzsu0426@stu.hist.edu.cn (J.S.); yhyao0803@stu.hist.edu.cn (Y.Y.); yyuan@stu.hist.edu.cn (Y.Y.)
2 Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
* Correspondence: zl.yang@siat.ac.cn

Abstract: The flexible job shop scheduling problem (FJSP) is an extension of the classical job shop scheduling problem and one of the more well-known NP-hard problems. To get better global optima of the FJSP, a novel hybrid whale optimization algorithm (HWOA) is proposed for solving FJSP, in which minimizing the makespan is considered as the objective. Firstly, the uniformity and extensiveness of the initial population distribution are increased with a good point set (GPS). Secondly, a new nonlinear convergence factor (NCF) is proposed for coordinating the weight of global and local search. Then, a new multi-neighborhood structure (MNS) is proposed, within which a total of three new neighborhoods are used to search for the optimal solution from different directions. Finally, a population diversity reception mechanism (DRM), which ensures to some extent that the population diversity is preserved with iteration, is presented. Seven international benchmark functions are used to test the performance of HWOA, and the results show that HWOA is more efficient. Finally, the HWOA is applied to 73 FJSP and four Ra international instances of different scales and flexibility, and the results further verify the effectiveness and superiority of the HWOA.

Keywords: whale optimization algorithm; flexible job shop scheduling problem; good point set; nonlinear convergence factor; multi-neighborhood structure; diversity reception mechanism

check for updates

Citation: Yang, W.; Su, J.; Yao, Y.; Yang, Z.; Yuan, Y. A Novel Hybrid Whale Optimization Algorithm for Flexible Job-Shop Scheduling Problem. *Machines* **2022**, *10*, 618. https://doi.org/10.3390/machines10080618

Academic Editor: Pingyu Jiang

Received: 21 June 2022
Accepted: 25 July 2022
Published: 27 July 2022

1. Introduction

The scheduling problem is the rationalization of the allocation of limited resources under certain constraints with the aim of achieving one or more target values to the satisfaction of the operator. Job-shop scheduling is one of the most important issues in the planning and operation of modern production and manufacturing systems, which is the key to the operation of a manufacturing system and the heart of production planning and control [1]. Based on the development of economic globalization, manufacturing enterprises are bound to face more and more severe competition. Therefore, a reasonable and effective production scheduling solution is indispensable to raise enterprise competitiveness. Currently, many companies still rely on experienced workers for production scheduling planning. Although experience-based scheduling solves the problem of resource allocation in the production process of manufacturing companies to a certain extent, the method relies too much on personal experience and can only be used for some simple, small-scale production scheduling problems. Furthermore, the global optimization scheduling plans cannot be guaranteed. Therefore, in large-scale production tasks, such experience-dependent manual scheduling is no longer effective. Meanwhile, traditional industrial production is updating and striding forward in the direction of flexibility, digitalization and intelligence. Therefore, it is essential to carry out the study on the flexible job shop scheduling problem.

The job-shop scheduling problem (JSP) is a typical NP-hard problem and is very important in production scheduling management systems [2]. In 1959, Bowman et al. first gave the definition of job shop scheduling [3]. For this problem, there are m machines and n jobs to be machined, where each job has one or more machining operations and is machined in a fixed operation sequence. Each operation requires a designated machine, and the machine is always available and can operate only one operation at a time without interruption. The good decision is how the job sequence is determined under the idle condition of the machines, which minimizes the makespan.

The flexible job-shop scheduling problem (FJSP) is an extension of the classic JSP [4]. In 1990, Bruker et al. first considered machine operation flexibility and proposed the concept of flexible job shop scheduling [5]. On the basis of operation allocation, machine allocation should also be considered. Each operation of a job can be operated on multiple machines, and the operation time on each machine is not necessarily the same. Actual production can be flexible according to the resource load situation. However, the increase in flexibility causes the FJSP to become complex [6]. Therefore, FJSP is more difficult and more suited to the actual needs of modern production than traditional JSP [7]. There are two general methods for solving FJSP: Firstly, the exact algorithm can get an effective optimal solution when it is used to solve small-scale problems. However, it is difficult to obtain the optimal solution in an acceptable period of time for large-scale scheduling problems [8]. Secondly, intelligent algorithms having good search ability are usually used to deal with complex FJSP and can obtain the optimal solution in a reasonable time. Such algorithms generally include genetic algorithms (GA), simulated annealing (SA), ant colony optimization (ACO), particle swarm optimization (PSO), grey wolf optimization (GWO), etc.

Driss et al. proposed a new genetic algorithm, that adopted a new chromosome representation method and different crossover and mutation strategies to solve the FJSP [9]. For example, Jiang et al. proposed a discrete grey wolf algorithm to solve FJSP, in which a search operator based on crossover operation was designed, so that the algorithm could work directly in the discrete domain, and an adaptive mutation method and variable neighborhood search were utilized to enhance the search performance of the algorithm [10]. Xu et al. proposed a parallel adaptive hybrid immune algorithm (HIA) to solve FJSP, where hybrid coding method, adaptive crossover operator, mutation operator based on coding antibody method, and affinity calculation based on group matching were used. Furthermore, a hybrid algorithm based on a simulated annealing algorithm was added to avoid trapping in local optimum [11]. Caldeira et al. proposed an improved Jaya algorithm to solve FSJP, in which an efficient initialization mechanism, local search technology and acceptance criteria were introduced to improve the quality of the solution. The test on 203 benchmark instances proved that it was effective in solving the FJSP problem [12]. Wu et al. proposed a non-dominated sorting genetic algorithm optimization algorithm for solving FJSP with energy consumption as the goal [13]. Zhang et al. designed a machine selection method including a global selection strategy, a local selection strategy, and a random selection strategy [14]. Kacem et al. proposed a local search algorithm that comprehensively considered the size of machine load and operation load to solve the machine allocation problem and then used a genetic algorithm to solve the operation sequencing problem [15]. Li et al. proposed a new hybrid algorithm to solve the FJSP by combining the powerful global search ability of GA with the efficient local search ability of TS [16]. Zhang et al. proposed a quantum genetic algorithm and applied it to dual resource-constrained FJSP [17]. Singh et al. introduced mutation factors to speed up the convergence based on the particle swarm optimization algorithm and used a chaotic mapping approach to generate initialization values [18]. Wang et al. used an improved ant colony algorithm to optimize the FJSP, in which the initialization mechanism, pheromone guidance mechanism, and node selection method were improved [19]. Cruz-Chávez et al. proposed a simulated annealing algorithm which incorporated a new cooling mechanism to accelerate convergence [20]. Gao et al. proposed an improved simulated degradation algorithm to solve the FJSP, combining SA with PSO to improve the simulated annealing operation and using the better particles to guide the search [21].

The WOA was proposed by Mirjalili et al. and is widely used in many fields, such as facial recognition [22], power optimization [23], aerospace [24], shop scheduling [25], path planning [26], photovoltaic cells [27], fault diagnosis [28], etc. In the field of shop scheduling, Abdel et al. studied the permutation flow shop scheduling problem using WOA [29]. Liu et al. optimized the JSP using WOA and used the Lévy flight strategy and DE strategy to improve the performance of WOA [30]. Luan et al. proposed a discrete WOA to solve low-carbon FJSP, where a new coding mechanism to solve the allocation of machines and operations and a hybrid variable neighborhood search to generate high-quality populations were all used [31]. From the discussions above, the WOA algorithm has received the growing attention of scholars because of its simple principle, fewer parameters to be adjusted, and easy implementation. Furthermore, WOA has been proven by many scholars that its convergence speed and convergence accuracy are higher than some classical group intelligence optimization algorithms. Therefore, WOA has been widely used to solve continuous function optimization problems, but seldom in discrete combination optimization problems, especially in FJSP. Thus, in this paper, a new hybrid whale optimization algorithm (HWOA) is studied and adopted to solve the flexible job shop scheduling problem, which is important to expand its application. Specifically as follows: Firstly, according to the characteristics of the FJSP discrete combinatorial optimization problem, the continuous values of the whale positions are mapped to the discrete value; secondly, the good point set strategy is used to generate the uniformly distributed population in the initial stage; thirdly, a multi-neighborhood structure is added to explore the depth of the solution; finally, the population diversity reception mechanism is also used to ensure population diversity with iteration.

The rest of the paper is organized as follows. Section 2 describes the definition of the problem. Section 3 introduces the original whale optimization algorithm. Section 4, the improved HWOA is proposed. Section 5 analyzes the solution results of HWOA on various international benchmark instances of FJSP. Section 6 summarizes the conclusions of this paper and presents the research direction of future work.

2. Problem Description

Flexible job shop scheduling can be described as: multiple jobs that need to be operated on multiple machines, each with multiple operations. Compared to classic JSP, the number of machines that can operate each operation of a job is no longer a fixed number of machines; it can be operated by one or more machines [32]. Therefore, FJSP has two assignment tasks: machine selection and operation sorting. Notations and abbreviations used in the article are described in Table 1.

Table 1. Notations and abbreviations used in the article.

Symbols	Description
n	Total number of jobs
m	Total number of machines
O	Total number of operations
i	Job serial number, $i \in \{1, 2, \cdots, n\}$
l_i	Number of operations for each job
j	Operation serial number, $j \in \{1, 2, \cdots, O\}$
k	Machine serial number, $k \in \{1, 2, \cdots, m\}$
E_k	Number of operations assigned to machine k
$O_{i,j}$	The jth operation of job i
$M_{i,j}$	The set of available operation machines for the jth operation of job i
$P_{i,j,k}$	The operation time required for the jth operation of job i on machine k
$S_{i,j}$	The start time of the operation of the jth operation of job i
$C_{i,j}$	End time of the operation of the jth operation of job i
C_i	Completion time of job i
C_{max}	Makespan
$x_{i,j,k}$	Binary decision variable that specifies whether the jth operation of job i is operated at machine k.

Table 1. *Cont.*

Symbols	Description
OS	Operation sequence
MS	Machine sequence
len	The length of the core path set composed of core operations
GPS	Good point set initialization
NCF	Nonlinear convergence factor
MNS	Multiple neighborhood structure
DRM	Diversity receiving mechanism
$MRPD$	Self-deviation percentage
Pro	Relative lift percentage
LB	Lower bound of test instances
UB	Upper bound for test instances
RPD	Deviation percentage

FJSP can be divided into two main categories:

(1) In the Total Flexible Job Shop Scheduling Problem (T-FJSP), any operation in any job can choose any machine from all machines for operations, as shown in Table 2;

(2) Partial Flexible Job Shop Scheduling Problem (P-FJSP), where each operation of each job can be operated by some of all the machines shown in Table 3.

Table 2. 3 × 4 Instance of T-FJSP.

Job	Operation	Machine			
		M1	M2	M3	M4
i_1	O_{11}	6	3	7	3
	O_{12}	3	2	6	1
i_2	O_{21}	5	1	6	4
	O_{22}	3	6	3	7
i_3	O_{31}	2	7	1	2
	O_{32}	7	3	4	4

Table 3. 3 × 4 Instance of P-FJSP.

Job	Operation	Machine			
		M1	M2	M3	M4
i_1	O_{11}	-	3	-	3
	O_{12}	3	2	-	1
i_2	O_{21}	-	1	-	4
	O_{22}	3	-	3	-
i_3	O_{31}	2	5	1	-
	O_{32}	-	3	4	-

In this study, the optimization objective is to minimize the makespan C_{max}, which is the minimum time required for all jobs to be machined. The mathematical model can be described as follows:

$$minC_{max} = min\ max\{C_1, C_2, \cdots, C_I\} \tag{1}$$

$$S_{i,j} + x_{i,j,k} \cdot P_{i,j,k} < S_{i,j+1} \tag{2}$$

$$\sum_{i=1}^{m} x_{i,j,k} = 1 \tag{3}$$

$$C_i \le C_{max} \tag{4}$$

$$C_{i,j} \le S_{i(j+1)} \tag{5}$$

$$S_{i,j} \geq 0, C_{i,j} \geq 0 \tag{6}$$

$$E_k \geq 1, M_{i,j} \geq 1 \tag{7}$$

where Equation (1) is the optimization objective function. Equation (2) indicates that the preceding step of the current operation of the same job has been completed. Equation (3) means that an operation can only be operated by one machine. Equation (4) ensures that the completion time of any one job cannot exceed the total completion time. Equation (5) requires that the completion time of the current job cannot be greater than the operation starting time of the next operation of the job. Equation (6) guarantees that all parameter variables cannot be negative. Equation (7) shows that each machine can operate at least one operation of the job, with at least one machine available for each operation of the job.

3. Whale Optimization Algorithm

Mirjalili et al., Australian scholars, proposed WOA in 2016 [24]. It has been widely used in many fields because of its simple principle, easy implementation, few parameters, high convergence accuracy, and fast convergence. WOA focuses on solving optimization problems by simulating the group hunting behaviors of humpback whales in nature, such as the operation of searching, encircling, and chasing. During the search, each whale has two behaviors to choose from, encircling and bubble net attacking. During the process of encircling the prey, the whales choose to swim towards the best whale or one selected randomly.

3.1. Encircling the Prey

In the WOA, the location of the target prey is unknown beforehand, so the current optimal whale individual is assumed to be the location of the target prey, and the other whale individuals approach the optimal individual to form an encirclement. The mathematical description of this behavior is described as:

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}\cdot\vec{D} \tag{8}$$

$$\vec{D} = |\vec{C}\cdot\vec{X}^*(t) - \vec{X}(t)| \tag{9}$$

$$\vec{A} = 2a\cdot\vec{r} - a \tag{10}$$

$$\vec{C} = 2\cdot\vec{r} \tag{11}$$

$$a = 2 - \left(\frac{2t}{t_{max}}\right) \tag{12}$$

where t represents the current iteration, $\vec{A}, \vec{C}$ are the coefficient vectors, X^* is the position vector of the best individual obtained so far, $\vec{X}$ is the position vector, $|\cdot|$ expresses the absolute value, $\vec{r}$ is a random vector within [0,1], a decreases linearly from 2 to 0 as the number of iterations increases, and t_{max} indicates the maximum number of iterations.

3.2. Search for The Prey (Exploration Phase)

In the operation of encircling the prey, whales also have a certain probability to approach other whales, and the mathematical formula for this behavior is as follows:

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A}\cdot\vec{D} \tag{13}$$

$$\vec{D} = |\vec{C}\cdot\overrightarrow{X_{rand}} - \vec{X}(t)| \tag{14}$$

where $\overrightarrow{X_{rand}}$ is the position vector of any individual in the whales population.

Whether an individual whale moves closer to the optimal individual position is determined by the value of the control factor $|\vec{A}|$. The whale moves an individual at random when $|\vec{A}| > 1$, position selection is shown in Figure 1; otherwise, the whale moves closer to the optimal individual. The mathematical description is defined as:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A}\cdot\vec{D} & |\vec{A}| < 1 \\ \overrightarrow{X_{rand}} - \vec{A}\cdot\vec{D} & |\vec{A}| \geq 1 \end{cases} \tag{15}$$

$$\vec{D} = \begin{cases} |\vec{C}\cdot\vec{X}^*(t) - \vec{X}(t)| & |\vec{A}| < 1 \\ |\vec{C}\cdot\overrightarrow{X_{rand}} - \vec{X}(t)| & |\vec{A}| \geq 1 \end{cases} \tag{16}$$

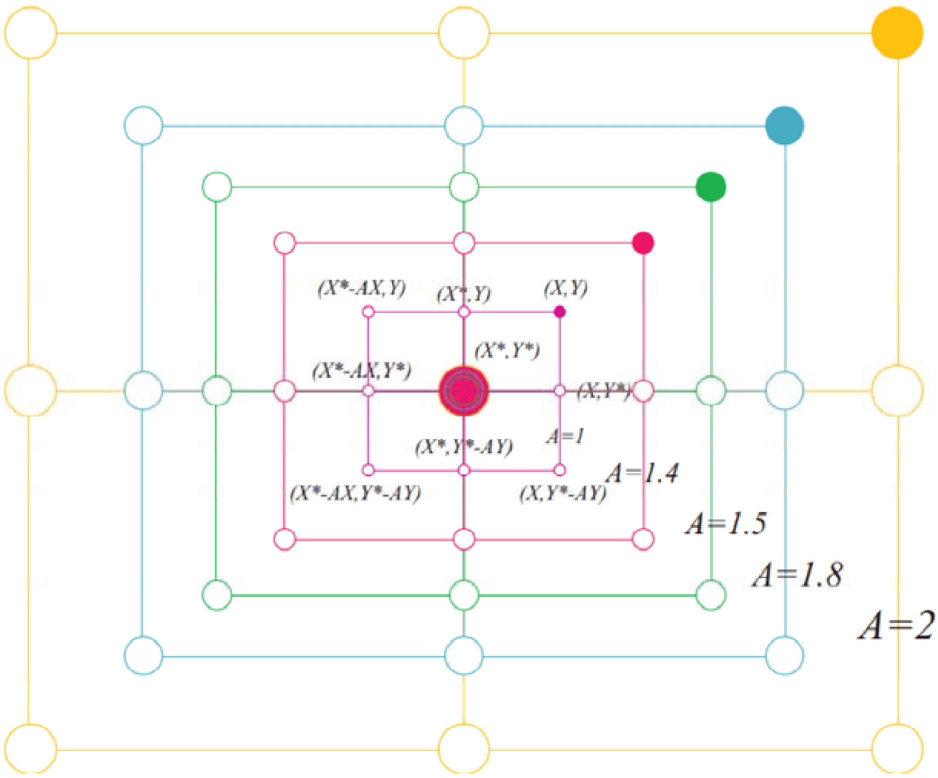


Figure 1. Effect on location selection by A value.

3.3. *Bubble-Net Attacking (Exploitation Phase)*

When hunting, whales swim in a circle and spew out bubbles to form a bubble net to hunt prey, as in Figure 2. When forming a bubble net attack, the whale constantly updates its position in a spiral motion. The mathematical model is described as:

$$\vec{X}(t+1) = \vec{D^l}\cdot e^{bl}\cdot\cos(2\pi l) + \vec{X}^*(t) \tag{17}$$

$$\vec{D^l} = |\vec{X}^*(t) - \vec{X}(t)| \tag{18}$$

where b is a constant used to define the shape of the spiral usually taken as 1, l is a random number in $[-1, 1]$.

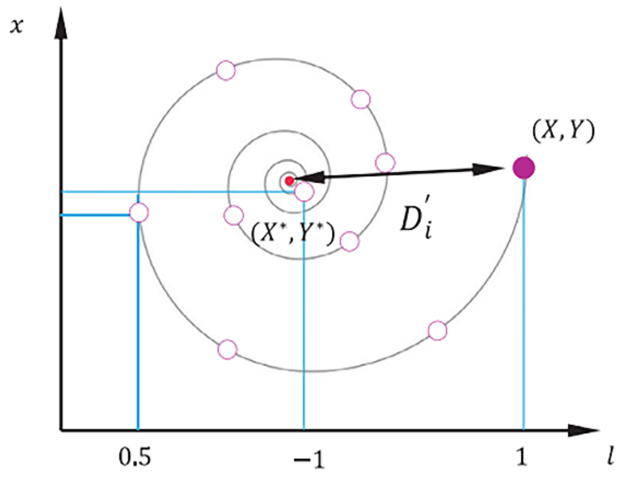
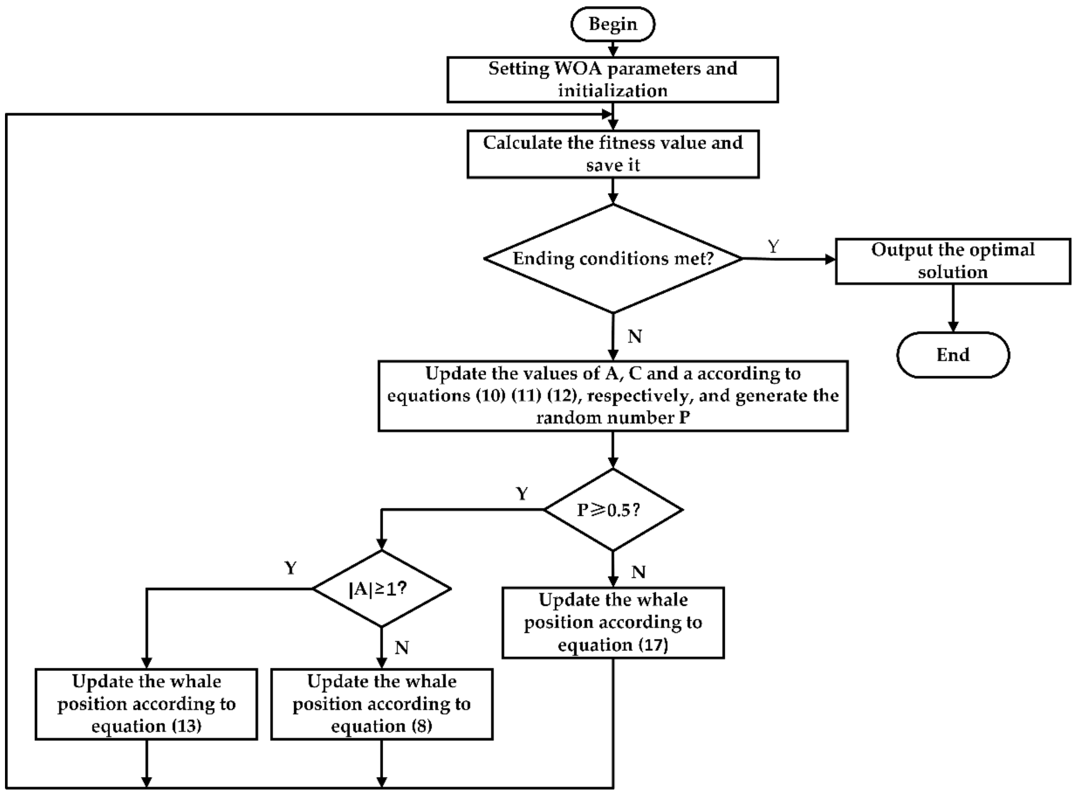


Figure 2. Spiral updating position.

The probability that the whale swims in a spiral form to continuously reduce the envelope is assumed to be P. Then the probability of enveloping its prey is $1 - P$, and usually P is set to 0.5. The mathematical model is as follows:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & P < 0.5 \\ \vec{D}^l \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & P \geq 0.5 \end{cases} \tag{19}$$

The detailed principle of WOA is shown in Figure 3.



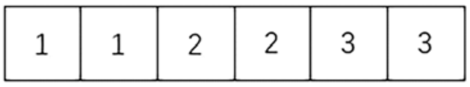
Begin
Setting WOA parameters and initialization
Calculate the fitness value and save it
Ending conditions met?
Y
Output the optimal solution
End
N
Update the values of A, C and a according to equations (10) (11) (12), respectively, and generate the random number P
P≥0.5?
Y
N
|A|≥1?
Y
N
Update the whale position according to equation (17)
Update the whale position according to equation (13)
Update the whale position according to equation (8)

Figure 3. The Flowchart of Basic WOA.

4. Proposed HWOA

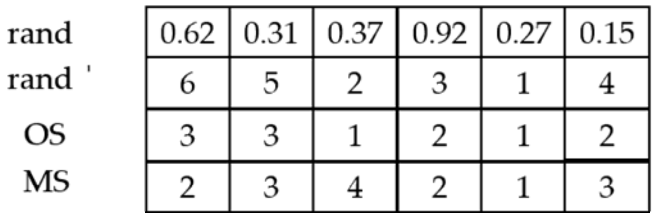
4.1. Encoding and Decoding

FJSP is a discrete combinatorial optimization problem, while WOA is an optimization algorithm used to solve the continuous optimization problem. Therefore, to solve the FJSP problem, WOA must be modified through a conversion mechanism. According to the characteristics of the FJSP problem, the operation sequencing problem and the machine assignment problem need to be ordered, so a two-layer coding approach including operation sequence (OS) and machine sequence (MS) is used. The principle can be described as follows: Firstly, generate the random numbers, of which the number is the same as the number of operations O; secondly, the generated random numbers are mapped from small to large, for instance, $rand = [0.62,\ 0.31,\ 0.37,\ 0.92,\ 0.27,\ 0.15]$ and the mapping sequence is $rand' = [6,\ 5,\ 2,\ 3,\ 1,\ 4]$, and the elements in the mapped rand' are used as the position index of the job number. Taking Table 2, described as 3×4 part of the flexible job shop scheduling, there are 3 jobs. Each job has 2 operations, each operation by multiple machinable machines, as an instance, of which encoding, and decoding are depicted in Figures 4 and 5.



1	1	2	2	3	3

Figure 4. Number of operations for each job.



rand	0.62	0.31	0.37	0.92	0.27	0.15
rand '	6	5	2	3	1	4
OS	3	3	1	2	1	2
MS	2	3	4	2	1	3

Figure 5. Schematic of encoding.

As shown in Figures 4 and 5, six random numbers are generated, and the OS and MS are determined based on the mapping sequence of the generated random numbers. The OS is ordered as follows: the first operation of job 3, the second operation of job 3, the first operation of job 1, the first operation of job 2, the second operation of job 1, and the second operation of job 2. According to the order of operation, the table of available operation machines for the current operation of the job is sequentially queried, and the machine selected for the current operation of each job is deemed as MS.

Furthermore, decoding can be described as, step 1: Determine the operation time required for each operation based on the operation sequencing table and the machine selection table; step 2 determine the end time of every operation, which is determined by the idle time and the processing time of the corresponding machine; and step 3 compare the complete times of all the jobs, and the smallest complete time of the job is chosen as the optimum.

4.2. Population Initialization

The quality of the initial population has a great impact on the performance of the algorithm to solve the problem. The initial population of WOA is generally generated randomly, which causes the initial population to be of low quality. Therefore, the theory of good point set is introduced to generate a uniformly distributed initial population, which effectively improves the diversity of the population and avoids premature convergence of the algorithm to a certain extent and is defined mathematically as follows [33]:

Let G be a unit cube in m-dimensional euclidean space, $x = (x_1, x_2, \cdots, x_m) \in G$, the point set is expressed as:

$$P_n(k) = \left\{ \left(\left\{ r_1^{(m)} * k \right\}, \left\{ r_2^{(m)} * k \right\}, \cdots \left\{ r_n^{(m)} * k \right\} \right), k = 1, 2, \cdots n \right\} \tag{20}$$

If the point set deviation satisfies:

$$\varphi(n) = C(r, \varepsilon) n^{-1+\varepsilon} \tag{21}$$

Then $P_n(k)$ is said to be a good point-set, r is a good point:

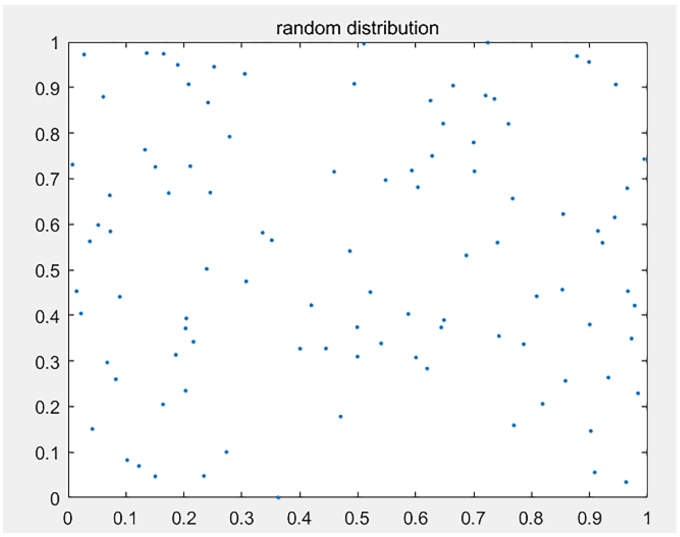
$$r_k = \{2\cos(2\pi k/p)\},\ 1 \le k \le m \tag{22}$$

In Equation (21), ε is an arbitrary positive number and $C(r, \varepsilon)$ is only a constant related to r, ε. In Equation (22), p is the smallest prime number that satisfies $2m + 3 \le p$.

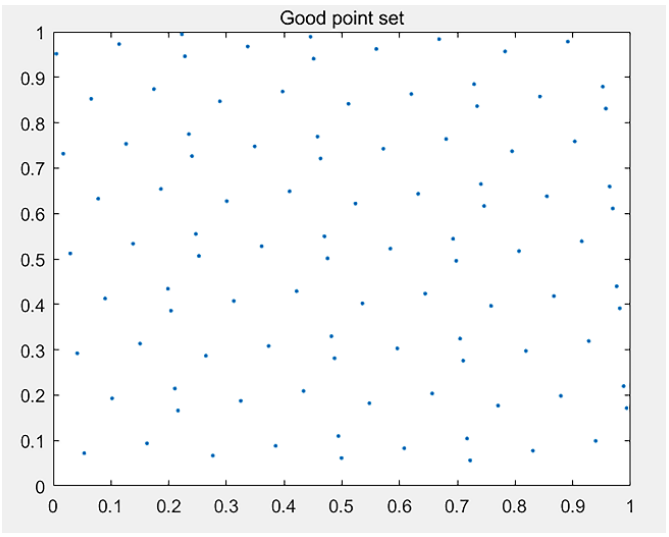
For the initial selection of the machine, a hybrid search strategy is used, of which 40% is generated with reference to the greedy algorithm and 60% is generated randomly to avoid falling into a local optimum during the iterations. Herein, taking the two-dimensional scatter plots in Figure 6 as an instance, it can be seen that the initial positions of the whale individuals initialized with the good point set are more uniformly distributed than those with random distribution, which can effectively ensure the diversity of the initial population solutions. The pseudo-code of the good point set initialization is described in Algorithm 1.

Algorithm 1. The population is initialized by the good point set.

```
1.Let the population size be G.
2. for n = 1:G do
3.   for k = 1:m do
4.     P takes the smallest prime number that satisfies p ≥ 2m + 3
5.     A matrix with n as rows, m as columns, and all elements in the columns as n
6.     Calculate the $r_k$ value by Equation (22)
7.     The initial population location $P$ is calculated by Equation (20)
8.   end for
9. end for
```



(**a**)



(**b**)

Figure 6. (**a**) Scatter plot of random distribution; (**b**) Scatter plot of good point set.

4.3. Nonlinear Convergence Factor

For swarm intelligence algorithms, how to coordinate the global search and the local search has a significant impact on the performance of the algorithm. The global search mechanism ensures that the algorithm can explore more solution space during iteration, thus avoiding falling into local optima. However, the local search mechanism ensures that the algorithm can exploit the solution as much as possible and thus speeds up the convergence of the algorithm. In the basic WOA, the global and local searches are mainly coordinated by the parameter $|\vec{A}|$, and it can be seen from Equation (10) that the value of $|\vec{A}|$ is linearly related to the convergence factor a, so the value of a can affect the algorithm's global and local search. The value of a in the basic WOA decreases linearly from 2 to 0 as the number of iterations increases. So, the parameter a leads to a slow convergence of the algorithm, and it is easy to fall into a local optimum. In fact, the algorithms should have a strong global search capability in the early search to ensure maintaining a faster convergence rate, and in the later search, local search should be used to enhance the accuracy of the solution. Based on the above analysis, it is necessary to introduce $\mathfrak{a}$ nonlinear convergence factor [34], whose mathematical expression is:

$$\mathfrak{a} = a_m(1 + \cos(\pi * (t-1)/(t_{max} - 1))) \tag{23}$$

where a_m is the initial value of the convergence factor $\mathfrak{a}$, t_{max} is the maximum number of iterations, and t is the current number of iteration. Meantime, comparison of before and after improvement of convergence factor $\mathfrak{a}$ is depicted in Figure 7.

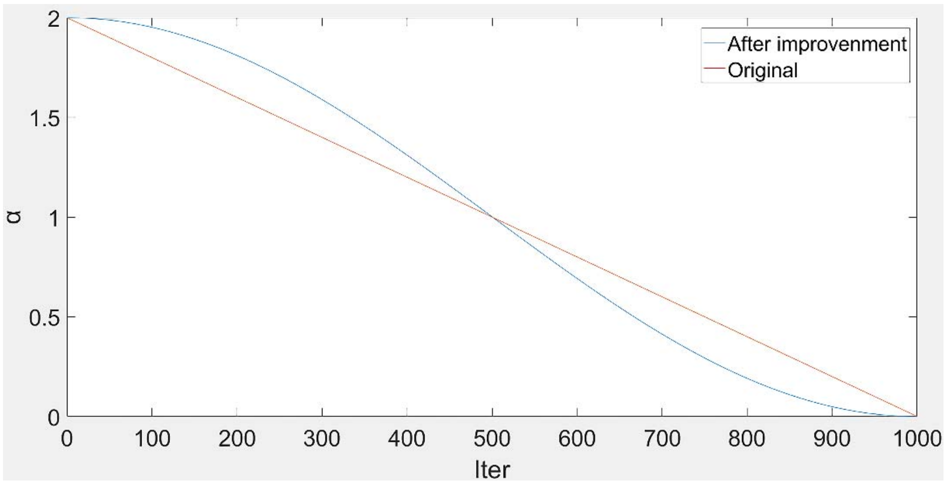


Figure 7. Comparison of before and after improvement of convergence factor $\mathfrak{a}$.

From Figure 7, it can be seen that the improved nonlinear convergence factor $\mathfrak{a}$ converges more slowly in the early and late stages of the algorithm and more quickly in the middle stage. Such a convergence approach can well coordinate the global search performance of the algorithm in the early stage with the local search performance in the latter stage and ensure the overall convergence speed of the algorithm.

4.4. Multi-Neighborhood Structure

A new multi-neighborhood structure for FJSP is proposed in which there are three new neighborhoods, N1, N2, and N3, and the optimization of FJSP is carried out from different directions.

Neighborhood structure N1: the scheduling solution is optimized in terms of different selection pairings of machines, and the machine selection is updated by a gene mutation mechanism.

Neighborhood structure N2: The scheduling solution is optimized in terms of the operation update mechanism, and a perturbation mechanism is proposed to disturb, in part, the position of whale individuals and avoid falling into a local optimum.

Neighborhood structure N3: Enhancing local search capability makes it easier for HWOA to break through the local optimum limit and lock the core operation for updating as the iteration proceeds.

4.4.1. Neighborhood Structure N1

During each iteration, the excellent machine-set gene sequences in the parent are retained to execute the machine gene mutation mechanism to obtain an alternative for the offspring. Let the set of parent machines be $M_i = (M_1, M_2, \cdots, M_n)$. Selection operation: b mutation points are selected from M_i after each iteration. The value of b decreases with the number of iterations. Machine gene mutation operation: for the selected b mutation points, the available operation machine set $M_{i,j}$ of the current operation is obtained and a new available operation machine set is generated from $M_{i,j}$. The new operation machine set is $M_i' = (x_{i1}', x_{i2}', \cdots, x_{in}')$. The machine mutation operation is shown in Figures 8 and 9, and the Gantt chart is shown in Figure 10. The available processing machines and processing times of each operation for each job are shown in Table 2. The meanings represented by the symbols $O_{i,j}$, $M_{i,j}$ in the figure are described in Table 3.

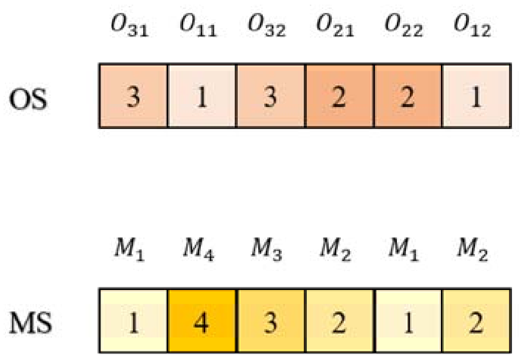

O_{31} O_{11} O_{32} O_{21} O_{22} O_{12}
OS
3 1 3 2 2 1
M_1 M_4 M_3 M_2 M_1 M_2
MS
1 4 3 2 1 2

Figure 8. Parental machine genes.

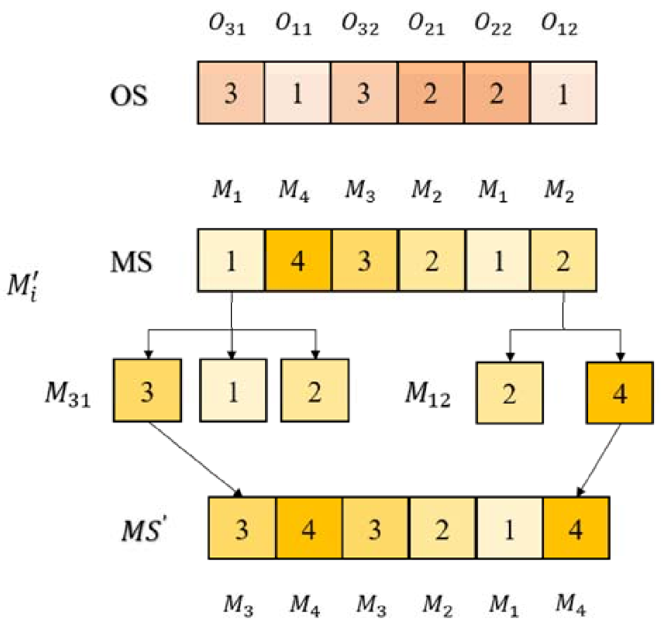

O_{31} O_{11} O_{32} O_{21} O_{22} O_{12}
OS
3 1 3 2 2 1
M_1 M_4 M_3 M_2 M_1 M_2
M_i'
MS
1 4 3 2 1 2
M_{31}
3 1 2
M_{12}
2 4
MS'
3 4 3 2 1 4
M_3 M_4 M_3 M_2 M_1 M_4

Figure 9. Machine gene after mutation.

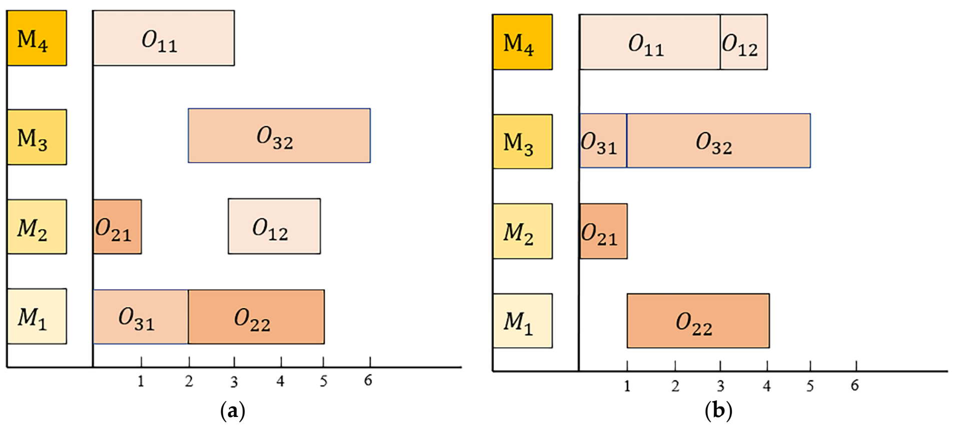


Figure 10. (**a**) Primitive Gantt chart; (**b**) Variant Gantt chart.

4.4.2. Neighborhood Structure N2

The initial population generated by using the good point set can fully ensure the diversity of the population in the early stage. However, in the late stage of the algorithm, the population diversity would be reduced due to the convergence of population individuals. Thus, the neighborhood structure N2 is designed as a perturbation to increase the population diversity. Let the population be $Popsize = (X_1, X_2, \cdots, X_N)$, the whale individuals be $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})$, and the adaptive interference probability be p. Generate a random number r during each iteration. If $r < p$ a disturbance operation is generated in this iteration, it generates a random perturbation factor β in the interval $[0, 10]$ to disturb the selected elements and generate a new individual $X_i' = (x_{i1}', x_{i2}', \cdots, x_{in}')$, as shown in Figure 11. Which is mathematically described as follows:

$$X_i' = \begin{cases} \beta * x_k & , r < p \\ x_{in} & , r > p \end{cases} \tag{24}$$

$$p = t/2 * t_{max} \tag{25}$$

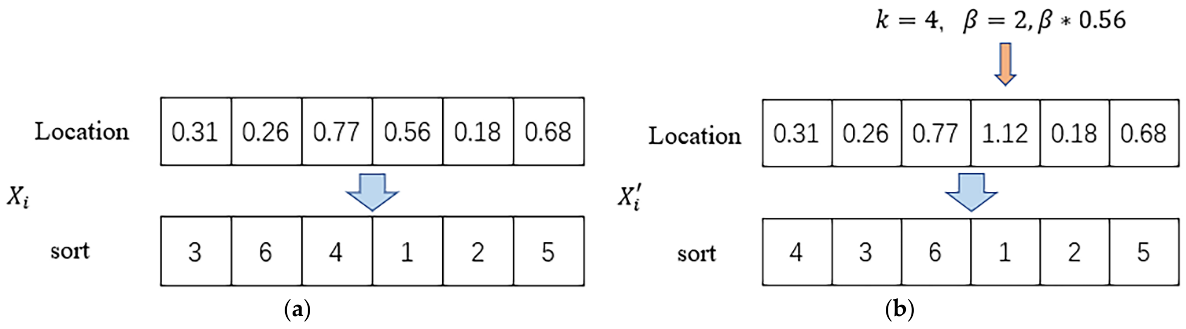


Figure 11. (**a**) Primitive Operation; (**b**) Disturbance Operation.

As shown in Figure 11, the perturbation operation occurs when $r < p$, the number of perturbations $z = 1$, the perturbation position $k = 4$, and the random perturbation factor $\beta = 2$. The number of perturbations and position of perturbations are determined randomly.

4.4.3. Neighborhood Structure N3

To enhance the local search capability of the algorithm, N3 uses the core operation across-machine gene reinforcement mechanism. Firstly, the sequence is determined which consumes the most time from the first operation to the last operation, as the start and end of core operations. Secondly, between the starting operation and the ending operation, starting from the ending operation and moving forward to find the operation sequence that is closely connected to the previous operation. Finally, all the operations found in the sequence can neither be advanced nor postponed until the starting operation is stopped, so that a core operation path is found. This means that any change to the sequence in the core operation would affect the makespan of the operation as well. To explain the neighborhood structure N3 in detail, Table 4 FJSP is taken as an instance, and the results are shown in Figures 12 and 13.

Table 4. 3 × 3 Instance of P-FJSP.

Job	**Operation**	**Machine**		
		M1	**M2**	**M3**
i_1	O_{11}	-	3	3
	O_{12}	3	2	1
i_2	O_{21}		1	4
	O_{22}	3	-	3
i_3	O_{31}	2	5	1
	O_{32}	-	3	4

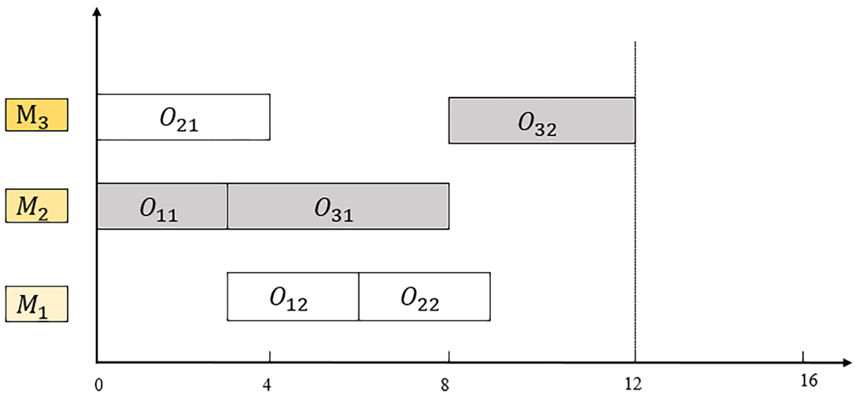


Figure 12. Original core operation Gantt chart.

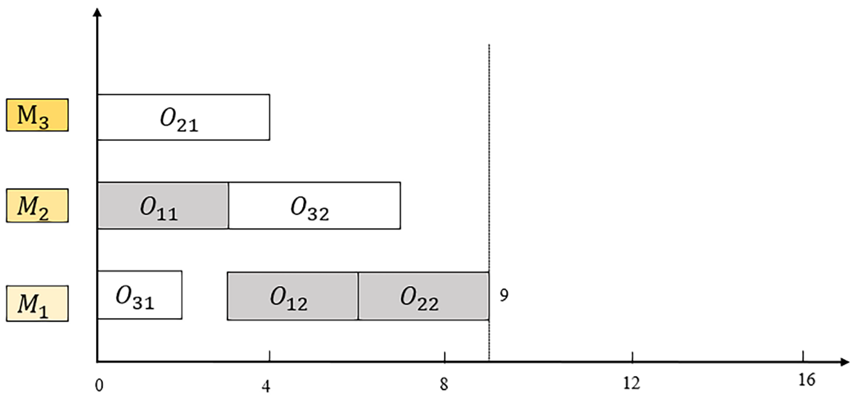


Figure 13. Gantt chart after across-machine gene reinforcement of core operation.

As shown in Figure 12, the gray part is the core operation, which is from the last gray part to the first gray part. It can be seen that any change in the arrangement of the core operation would directly affect the makespan. Adopting the core operation across-machine gene reinforcement mechanism, the core operation is reinforced with machine genes one by one without prolonging the makespan. During the execution of machine gene mutation operations in the core operation, only operations that produce positive optimization are retained, and the set of machine genes in the core operation is continuously enhanced. For sequences with reduced makespans due to machine set gene changes, which are preserved in the next generation. As shown in Figures 12 and 13, the set of core operations before updating are (O_{11}, O_{31}, O_{32}), with a makespan of 12, and the set of core operations after updating are (O_{11}, O_{12}, O_{22}), with a makespan of 9. The pseudo-code of the core operation, across-machine gene reinforcement mechanism, is described in Algorithm 2.

Algorithm 2. Core operation across-machine gene reinforcement mechanism.

```
1. Let the current core operation arrangement code be $OS_{i,j}$, the machine arrangement code be $MS_{i,j,k}$, the length of the core path set composed of core operations be $len$, and the fitness function be F
2.    for $i = 1 : len$
3.        Select the $i$ operation on the core path, and the set composed of its operation set $OS_{i,j}$ and machine set $MS_{i,j,k}$ is denoted as $S_i$.
4.        The across-machine gene reinforcement operation is performed on $MS_{i,j,k}$, and the newly generated machine set is noted as $machine_{i,j,k}$, and the set of newly generated machine set and operation set is noted as $Ups_i$
5.        if F($Ups_i$) < F($S_i$)
6.            $MS_{i,j,k} = machine_{i,j,k}$
7.        end if
8.    end for
```

4.5. Diversity Reception Mechanism

By using the diversity reception mechanism, the algorithm can accept poor individuals to improve the population diversity, which effectively avoid falling into local optimum. Its mathematical description is as follows.

$$P = \begin{cases} 1, & f_n \le f_o \\ 5 * ((f_n - f_o)/f_o), & f_n > f_o \end{cases} \tag{26}$$

where f_n is the fitness value of newly generated individuals in the current generation and f_o is the fitness value of individuals in the previous generation. When $P = 1$, the newly generated superior whale individual is received completely. Otherwise, a random number $q \in [0, 0.5]$ is generated, and if $P < q$ the newly generated inferior whale individual is received. The pseudo-code of HWOA is described in Algorithm 3.

Algorithm 3. HWOA.

```
1. Initialize the parameters and population
2. Calculate the fitness value and save the optimal individual position
3. while $t < t_{max}$ do
4.    for $i = 1 : N$ do
5.        Calculate the value of $A$ according to Equation (10), the value of $C$ according to Equation (11) and the value of the nonlinear convergence factor $\alpha$ according to Equation (23)
6.        Generate a random number $P$ within $[0, 1]$
7.        if $P \ge 0.5$ do
8.            if $|A| \ge 1$ do
9.                Update individual whale positions according to Equation (8)
10.           else if
11.               Update individual whale positions according to Equation (13)
12.           end if
13.       else if
14.           Update individual whale positions according to Equation (17)
15.       end if
16.   end for
17. Perform multi-neighborhood structure updates and retain the resulting improved solutions
18. Carry out the diversity reception mechanism
19. Update the optimal individual position
20. t = t + 1
21. end while
22. end
```

5. Experimental Analysis

5.1. Benchmark Functions Test

In order to verify the performance of HWOA, we chose 7 classic benchmark functions to test [35]. We list the functions in Table 5. Seven benchmark function tools have the following four characteristics: single mode function, multimode function, separable function, integral function, and to US, MS, the US, and UN tag in Table 5.

Table 5. Details of benchmark functions.

Name	Function	C	Search Range	Min
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	US	[−100, 100]	0
Sumsquare	$f_2(x) = \sum_{i=1}^{D} ix_i^2$	US	[−10, 10]	0
Schwefel2.22	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	UN	[−10, 10]	0
Rosenbrock	$f_4(x) = \sum_{i=1}^{D-1} \left\{100 \cdot \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right\}$	UN	[−5, 10]	0
Rastrigin	$f_5(x) = \sum_{i=1}^{D} [x_i^2 - 100\cos(2\pi x_i)] + 10D$	MS	[−5.12, 5.12]	0
Ackley	$f_6(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	MN	[−32, 32]	0
Levy	$f_7(x) = \sum_{i=1}^{D-1} (x_i - 1)^2 \left\lvert 1 + sin^2(3\pi x_{i+1}) \right\rvert + sin^2(3\pi x_1) + \lvert x_{D-1} \rvert \left[1 + sin^2(3\pi x_D)\right]$	MN	[−10, 10]	0

The computer used in this study is: Windows 10 operating system and an Intel Pentium Gold G5420 CPU, with a frequency of 3.80 GH_Z, and 8 GB of RAM memory using the MATLAB programming language to realize the following test.

In order to verify the superiority of HWOA, we compared it with the following seven algorithms: WOA, MFO, GWO, SCA, MWO, DA, and SSA. In order to ensure the fairness of the experiment, we set up the same experimental parameters and adopted three dimensions to test: dimension D = 30, 50, and 100. To reduce the influence of randomness, each algorithm was run 30 times independently.

5.2. Performance Evaluation Compared with Other Algorithms

The results obtained by HWOA, and the other seven algorithms are listed in Tables 6–8. Table 6 shows the results of the maximum iteration $t_{max} = 500$ obtained by HWOA and the other seven algorithms under 30 dimensions. Table 7 shows the results of $t_{max} = 2000$ obtained by HWOA and the other seven algorithms under 50 dimensions. Table 8 shows the results of $t_{max} = 8000$ obtained by HWOA and the other seven algorithms under 100 dimensions. *Min* is the minimum value obtained from the eight algorithms, *Mean* is the mean value obtained by each algorithm independently running at 30, *Std* is the benchmark deviation obtained by each algorithm independently running at 30, and *Sig* is the Wilcoxon signed-rank test. Analysis is conducted under the significance level of 0.05. The results are marked as "+/=/−", corresponding to HWOA superior, equal, and worse than the comparison algorithm, respectively. The convergence curves of the algorithm in three dimensions are shown in Figure 14.

Table 6. Comparison of results for 30-dimension benchmark functions.

Algorithm		f_1	f_2	f_3	f_4	f_5	f_6	f_7
HWOA	Min	$\mathbf{3.42 \times 10^{-132}}$	$\mathbf{6.04 \times 10^{-132}}$	$\mathbf{2.61 \times 10^{-78}}$	$\mathbf{2.62 \times 10^{1}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{2.68 \times 10^{-2}}$
	Mean	$\mathbf{2.22 \times 10^{-120}}$	$\mathbf{4.45 \times 10^{-120}}$	$\mathbf{2.29 \times 10^{-71}}$	$\mathbf{2.68 \times 10^{1}}$	**0**	$\mathbf{3.14 \times 10^{-15}}$	$\mathbf{1.40 \times 10^{-1}}$
	Std	$\mathbf{1.13 \times 10^{-119}}$	$\mathbf{2.41 \times 10^{-119}}$	$\mathbf{7.84 \times 10^{-71}}$	$\mathbf{2.36 \times 10^{-1}}$	**0**	$\mathbf{1.98 \times 10^{-15}}$	$\mathbf{1.08 \times 10^{-1}}$
WOA	Mean	1.44×10^{-72}	4.41×10^{-78}	1.17×10^{-50}	2.79×10^{1}	1.89×10^{-15}	3.85×10^{-15}	5.19×10^{-1}
	Std	7.22×10^{-72}	1.58×10^{-77}	2.77×10^{-50}	4.79×10^{-1}	1.04×10^{-14}	2.48×10^{-15}	2.08×10^{-1}
	sig	+	+	+	+	+	=	+
MFO	Mean	2.34×10^{3}	6.04×10^{2}	3.64×10^{1}	5.35×10^{6}	1.59×10^{2}	1.46×10^{1}	2.24×10^{2}
	Std	5.04×10^{3}	8.47×10^{2}	1.99×10^{1}	2.03×10^{7}	4.07×10^{1}	7.66	1.09×10^{3}
	sig	+	+	+	+	+	+	+
GWO	Mean	1.65×10^{-27}	2.44×10^{-28}	1.08×10^{-16}	2.70×10^{1}	2.88	1.05×10^{-13}	6.53×10^{-1}
	Std	3.22×10^{-27}	3.80×10^{-28}	6.52×10^{-17}	7.85×10^{-1}	4.25	1.97×10^{-14}	1.98×10^{-1}
	sig	+	+	+	+	+	+	+
SCA	Mean	1.11×10^{1}	2.12	2.23×10^{-2}	6.33×10^{4}	2.64×10^{1}	1.45×10^{1}	1.13×10^{5}
	Std	3.32×10^{1}	4.69	3.06×10^{-2}	1.34×10^{5}	3.38×10^{1}	8.95	2.79×10^{5}
	sig	+	+	+	+	+	+	+
MVO	Mean	1.31×10	1.40	1.25×10^{1}	4.76×10^{2}	1.27×10^{2}	1.54	2.19×10^{-1}
	Std	3.71×10^{-1}	1.55	3.48×10^{1}	7.71×10^{2}	3.37×10^{1}	4.48×10^{-1}	1.57×10^{-1}
	sig	+	+	+	+	+	+	+
DA	Mean	1.69×10^{3}	2.39×10^{2}	1.60×10^{1}	3.10×10^{5}	1.67×10^{2}	1.09×10^{1}	3.23×10^{5}
	Std	6.59×10^{2}	1.95×10^{2}	5.52	3.06×10^{5}	3.77×10^{1}	1.90	3.98×10^{5}
	sig	+	+	+	+	+	+	+
SSA	Mean	4.79×10^{-7}	1.83	1.46	1.69×10^{2}	5.04×10^{1}	2.75	1.37×10^{1}
	Std	1.14×10^{-6}	1.54	1.02	2.63×10^{2}	1.38×10^{1}	6.48×10^{-1}	1.54×10^{1}
	sig	+	+	+	+	+	+	+

The best results are highlighted in boldface.

Table 7. Comparison of results for 50-dimension benchmark functions.

Algorithm		f_1	f_2	f_3	f_4	f_5	f_6	f_7
HWOA	Min	**0**	**0**	$\mathbf{2.17 \times 10^{-310}}$	$\mathbf{2.34 \times 10^{-2}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{9.69 \times 10^{-4}}$
	Mean	**0**	**0**	$\mathbf{4.92 \times 10^{-295}}$	$\mathbf{4.76 \times 10^{-1}}$	**0**	$\mathbf{3.02 \times 10^{-15}}$	$\mathbf{7.16 \times 10^{-2}}$
	Std	**0**	**0**	**0**	$\mathbf{4.90 \times 10^{-1}}$	**0**	$\mathbf{2.57 \times 10^{-15}}$	$\mathbf{1.01 \times 10^{-1}}$
WOA	Mean	9.86×10^{-305}	2.47×10^{-300}	2.26×10^{-209}	4.67×10^{1}	1.89×10^{-15}	4.32×10^{-15}	2.00×10^{-1}
	Std	0	0	0	6.10×10^{-1}	1.04×10^{-14}	2.72×10^{-15}	1.33×10^{-1}
	sig	+	+	+	+	+	=	+
MFO	Mean	8.00×10^{3}	2.19×10^{3}	6.97×10^{1}	1.07×10^{7}	2.96×10^{2}	1.89×10^{1}	2.73×10^{7}
	Std	8.47×10^{3}	1.93×10^{3}	3.59×10^{1}	2.77×10^{7}	4.65×10^{1}	2.85×10^{0}	1.04×10^{8}
	sig	+	+	+	+	+	+	+
GWO	Mean	1.70×10^{-91}	7.22×10^{-92}	1.28×10^{-53}	4.68×10^{1}	1.89×10^{-15}	1.58×10^{-14}	1.69×10
	Std	4.06×10^{-91}	2.82×10^{-91}	1.73×10^{-53}	7.93×10^{-1}	1.04×10^{-14}	2.54×10^{-15}	2.88×10^{-1}
	sig	+	+	+	+	+	+	+
SCA	Mean	1.58×10^{-1}	9.18×10^{-2}	1.45×10^{-5}	2.81×10^{5}	4.58×10^{1}	1.73×10^{1}	3.67×10^{5}
	Std	3.33×10^{-1}	2.64×10^{-1}	4.26×10^{-5}	1.19×10^{6}	3.91×10^{1}	7.02×10	1.29×10^{6}
	sig	+	+	+	+	+	+	+
MVO	Mean	6.06×10^{-1}	3.00×10	1.47×10^{1}	4.51×10^{2}	2.25×10^{2}	1.61×10	1.98×10^{-1}
	Std	1.54×10^{-1}	2.44×10	4.47×10^{1}	6.51×10^{2}	4.70×10^{1}	5.53×10^{-1}	1.38×10^{-1}
	sig	+	+	+	+	+	+	+
DA	Mean	2.78×10^{3}	5.71×10^{2}	2.64×10^{1}	4.02×10^{5}	2.98×10^{2}	9.83×10	1.61×10^{5}
	Std	1.17×10^{3}	2.44×10^{2}	8.71×10	2.53×10^{5}	5.00×10^{1}	1.16×10	1.65×10^{5}
	sig	+	+	+	+	+	+	+
SSA	Mean	3.08×10^{-8}	7.72×10^{-1}	2.34×10	1.04×10^{2}	1.01×10^{2}	2.75×10	3.99×10^{1}
	Std	5.12×10^{-9}	1.01×10	1.40×10	7.38×10^{1}	2.44×10^{1}	6.67×10^{-1}	2.93×10^{1}
	sig	+	+	+	+	+	+	+

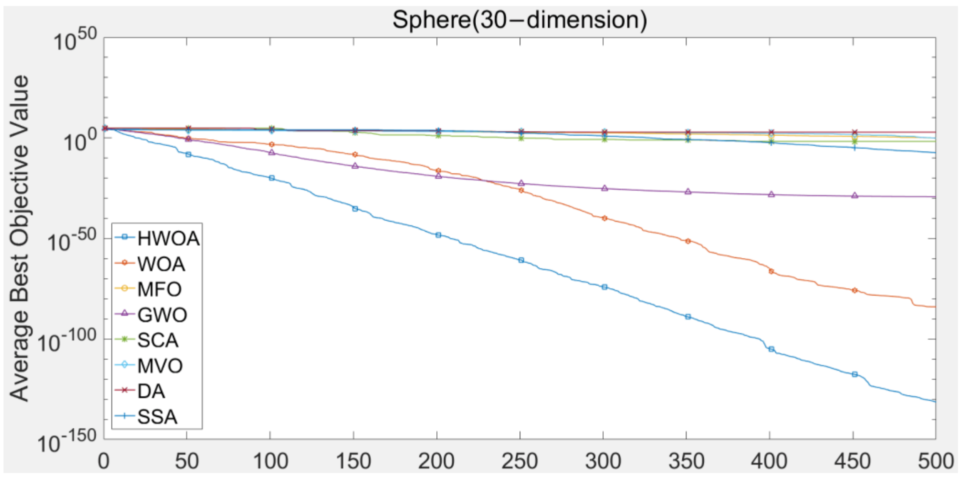
The best results are highlighted in boldface.

Table 8. Comparison of results for 100-dimension benchmark functions.

Algorithm		f_1	f_2	f_3	f_4	f_5	f_6	f_7
HWOA	Min	**0**	**0**	**0**	$\mathbf{2.22 \times 10^{-2}}$	**0**	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{4.47 \times 10^{-4}}$
	Mean	**0**	**0**	**0**	$\mathbf{3.10 \times 10^{-1}}$	**0**	$\mathbf{2.31 \times 10^{-15}}$	$\mathbf{1.96 \times 10^{-2}}$
	Std	**0**	**0**	**0**	$\mathbf{2.40 \times 10^{-1}}$	**0**	$\mathbf{2.40 \times 10^{-15}}$	$\mathbf{3.48 \times 10^{-2}}$
WOA	Mean	0	0	0	9.50×10^{1}	0	3.73×10^{-15}	2.76×10^{-2}
	Std	0	0	0	3.54×10^{-1}	0	2.54×10^{-15}	3.94×10^{-2}
	sig	=	=	=	+	=	+	+
MFO	Mean	1.95×10^{4}	1.47×10^{4}	1.25×10^{2}	4.58×10^{7}	6.40×10^{2}	1.99×10^{1}	1.39×10^{8}
	Std	1.61×10^{4}	7.70×10^{3}	5.56×10^{1}	6.95×10^{7}	7.41×10^{1}	1.39×10^{-1}	2.45×10^{8}
	sig	+	+	+	+	+	+	+
GWO	Mean	1.98×10^{-265}	8.93×10^{-266}	3.26×10^{-156}	9.72×10^{1}	0	1.51×10^{-14}	5.71
	Std	0	0	0	7.60×10^{-1}	0	1.87×10^{-15}	3.36×10^{-1}
	sig	+	+	+	+	=	+	+
SCA	Mean	1.84×10^{2}	5.11×10^{1}	4.13×10^{-7}	4.21×10^{6}	1.03×10^{2}	1.94×10^{1}	2.10×10^{7}
	Std	2.70×10^{2}	1.19×10^{2}	1.82×10^{-6}	5.18×10^{6}	6.26×10^{1}	4.38	2.58×10^{7}
	sig	+	+	+	+	+	+	+
MVO	Mean	6.05×10^{-1}	1.04×10^{1}	7.14×10^{4}	4.12×10^{2}	5.81×10^{2}	4.07	1.13
	Std	1.10×10^{-1}	5.81	3.90×10^{5}	6.34×10^{2}	8.02×10^{1}	6.19	2.73
	sig	+	+	+	+	+	+	+
DA	Mean	3.08×10^{3}	1.68×10^{3}	4.46×10^{1}	4.45×10^{5}	6.17×10^{2}	8.11	1.17×10^{5}
	Std	1.36×10^{3}	7.42×10^{2}	1.66×10^{1}	3.44×10^{5}	1.49×10^{2}	1.93	2.29×10^{5}
	sig	+	+	+	+	+	+	+
SSA	Mean	8.03×10^{-8}	1.49	6.24	1.81×10^{2}	2.06×10^{2}	3.78	1.27×10^{2}
	Std	8.91×10^{-9}	1.79	3.27	1.61×10^{2}	4.48×10^{1}	7.02×10^{-1}	2.63×10^{1}
	sig	+	+	+	+	+	+	+

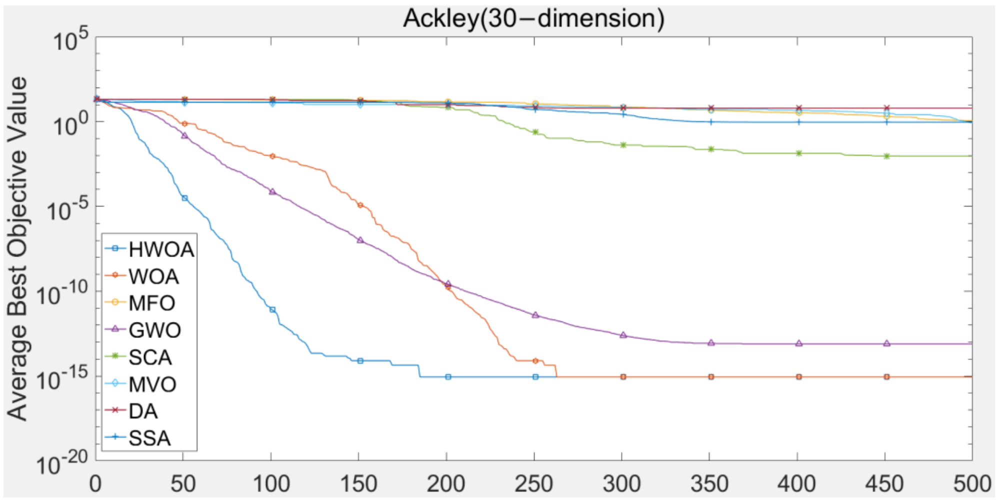
The best results are highlighted in boldface.

As can be seen in Tables 6–8, HWOA outperforms the other seven algorithms in both mean results and standard deviation at 30- and 50-dimensions, and in the Wilcoxon signed-rank test results, HWOA and WOA achieve the same performance results in the benchmark function f_6, but the other six criterion functions, and our proposed HWOA in the Wilcoxon signed-rank test results all outperformed the other algorithms, and at 50-dimensions, HWOA found the optimal values in the benchmark functions f_1, f_2, and f_5. In 100 dimensions, HWOA finds the optimal values in the benchmark functions f_1, f_2, f_3, and f_5. Although WOA also finds the optimal values in the benchmark functions f_1, f_2, f_3, and f_5, it can be seen from the data in Tables 6–8 and Figure 14 that the proposed HWOA outperforms the other seven algorithms in terms of convergence speed for different dimensions. These results show that HWOA shows strong robustness for different dimensional problems and also proves the superiority of our proposed HWOA algorithm.

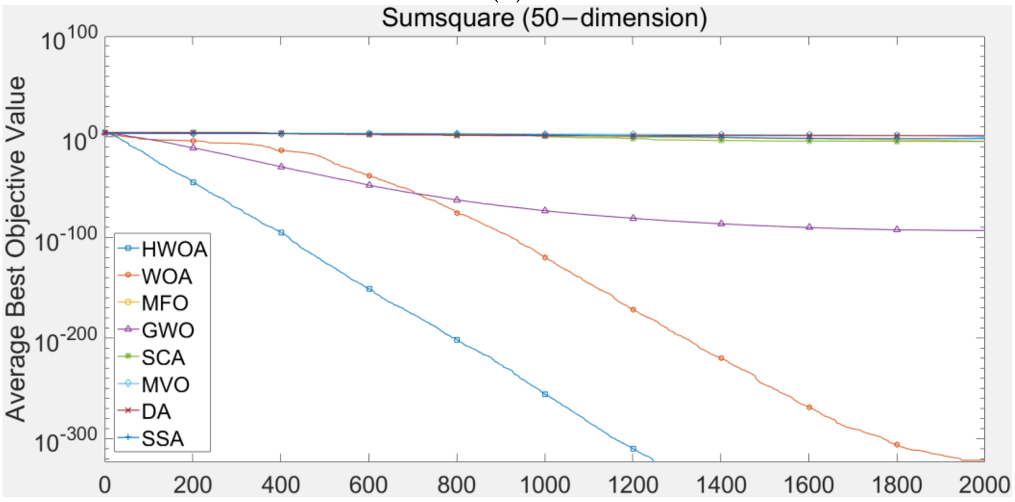


(**a**)

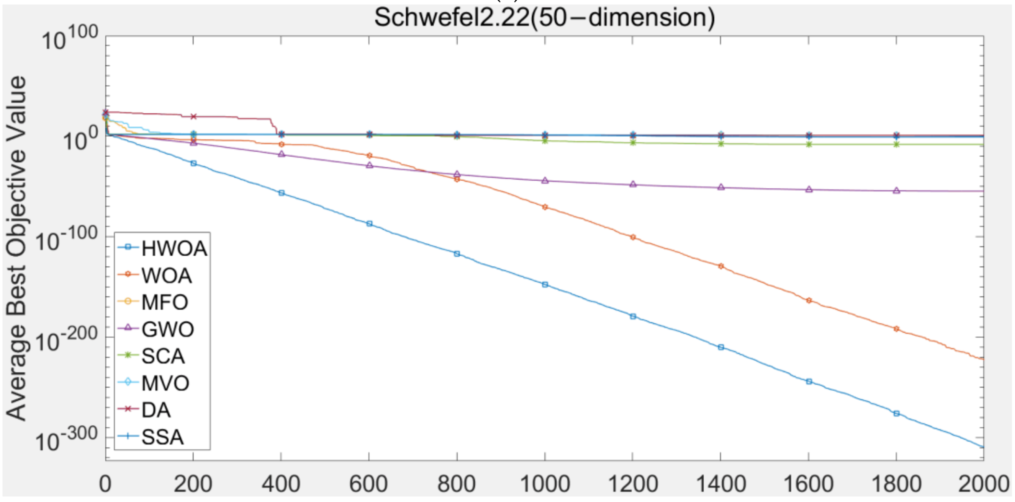
Figure 14. *Cont.*



(**b**)

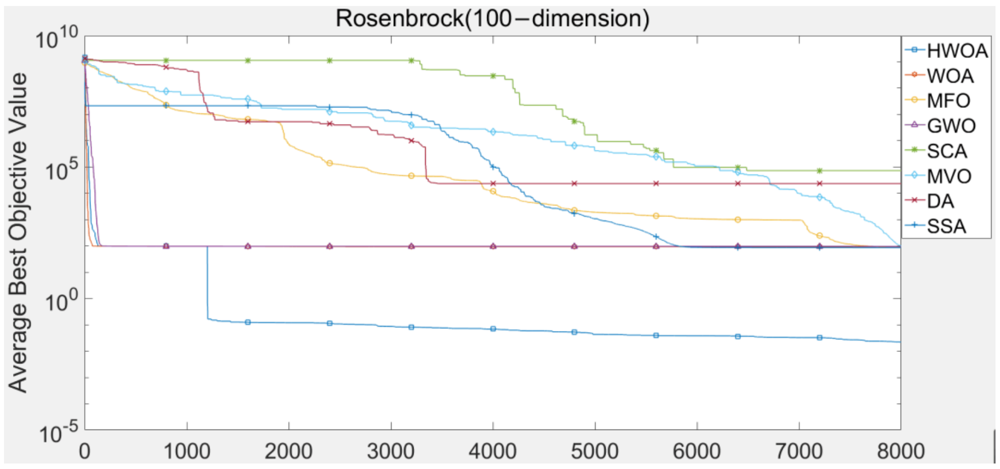


(**c**)

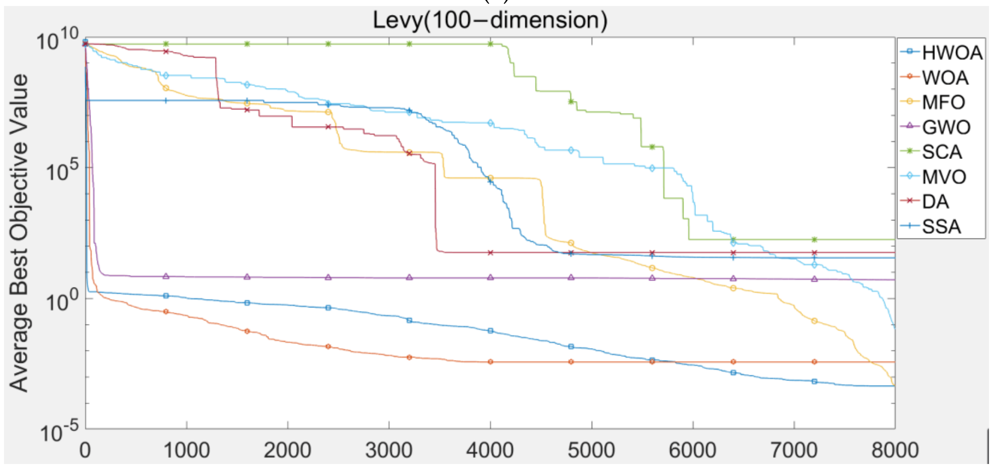


(**d**)

Figure 14. *Cont*.



(**e**)



(**f**)

Figure 14. (**a**) Sphere 30−dimension Convergence Curve; (**b**) Ackley 30−dimension Convergence Curve; (**c**) Sumsquare 50−dimension Convergence Curve; (**d**) Schwefel2.22 50−dimension Convergence Curve. (**e**) Rosenbrock 100−dimension Convergence Curve. (**f**) Levy 100−dimension Convergence Curve.

5.3. Application to Flexible Job Shop Scheduling

5.3.1. Experimental Settings

The parameters of HWOA are set as follows: population size of 100, vector dimension n , spiral coefficient $b = 1$, maximum number of iterations $t_{max} = 500$, and $a_m = 1$.

In order to verify the feasibility and effectiveness of the HWOA, four sets of internationally known instances and a set of randomly generated instances are selected. The first set of FJSP instances proposed by Brandimate [36] includes 10 representative FJSP problems, named Mk01-Mk10, which are one of the standard instances for studying FJSP at present. The second set of FJSP instances proposed by Kacem [15] contains five instances of FJSP of different sizes. The third set of instances is the Fattahin instances, which consists of a total of 18 small and medium-sized FJSPs. The fourth set of instances is the Hurink base instance (vdata) with a high degree of flexibility, which has 40 FJSPs of different sizes. The fifth group is the randomly generated FJSP instances, with four large-scale FJSPs. The values of the instance data obey a uniform distribution within a certain range, the number of operations for each job is the same, the available operation machines are randomly generated within the total number of machines, and the operation corresponding time is generated within $[1, 100]$. In order to eliminate the randomness in the experimental process, multiple tests are conducted on each instance.

5.3.2. Strategy Validity Verification

Four strategies are used to improve the efficiency of the HWOA algorithm for FJSP, which include good point set initialization (GPS), nonlinear convergence factor (NCF), multiple neighborhood structure (MNS), and diversity receiving mechanism (DRM). The strategy of adding GPS and NCF in WOA is named HWOA-1, the policy of adding N1 and N2 neighborhoods in HWOA-1 is named HWOA-2, and the approach of adding N3 neighborhoods in HWOA-2 is named HWOA. HWOA-1, HWOA-2, and HWOA all contain DRM in the three algorithms. For a fair comparison, each algorithm has 10 dependent runs for each trial. LB and UB are the lower and upper limits of the test case, respectively. C_{min} is the minimum value of each algorithm in 10 runs, and C_{mean} is the mean value. O_{min} is the minimum value of all algorithms compared in 10 runs, $MPRD$ is the relative deviation percentage of O_{min}, which is calculated as in Equation (27), and the relative lift percentage Pro is calculated by Equation (28).

$$MRPD = \frac{C_{min} - O_{min}}{O_{min}} \times 100 \tag{27}$$

$$Pro = \frac{Oar - Crr}{Oar} \times 100\% \tag{28}$$

where Crr and Oar are the best values obtained in 10 runs by HWOA and the original WOA, respectively.

As for the data in Table 9, the first column is the name of the instance, the second column is the size of the instance, the third column is the internationally published optimal lower bound for the instance, and the fourth column is the internationally published upper bound for the solution of the instance.

Table 9. BRdata Instance Verification.

BRdata	n × m	LB	UB	WOA		HWOA-1		HWOA-2		HWOA	
				C_{min}	C_{mean}	C_{min}	C_{mean}	C_{min}	C_{mean}	C_{min}	C_{mean}
MK01	10 × 6	36	42	42	45.7	42	44.3	40	41.8	**40**	**41.3**
MK02	10 × 6	24	32	33	39.9	33	36.5	27	28.4	**26**	**27.5**
MK03	15 × 8	204	211	231	235.1	223	228.3	204	213.1	**204**	**207.4**
MK04	15 × 8	48	81	73	76.3	73	74.7	64	67.3	**62**	**63.8**
MK05	15 × 4	168	186	175	181.5	175	181.9	173	177.3	**173**	**174.5**
MK06	10 × 15	33	86	98	101.7	98	100.7	65	69.5	**62**	**65.4**
MK07	20 × 5	133	157	154	158.7	155	160.7	145	147.6	**143**	**145.2**
MK08	20 × 10	523	523	531	541.8	531	539.3	523	526.1	**523**	**523**
MK09	20 × 10	299	369	379	392.5	383	391.5	312	327.9	**310**	**319.9**
MK10	20 × 15	165	296	271	302.4	265	296.5	224	233.3	**216**	**220.3**

The best results are highlighted in boldface.

From Table 9, it can be seen that the HWOA algorithm shows superior capability in the BRdata standard instance, and all the results are better than those of WOA, HWOA-1, and HWOA-2. Although some of the results do not reach the lower limit of the instance, they are within reasonable limits. HWOA-1 and WOA obtain the same minimum value C_{min} in MK02, MK04, MK06, MK09, and MK10, but the overall results of HWOA-1 are better than those of WOA under the synergistic effect of GN and NCF strategies.

From Figure 15, it can be concluded that HWOA achieves a better solution in the MK01 instance compared to WOA and HWOA-1, and converges to a superior solution earlier compared to HWOA-2. In Figure 16, it can be seen that the mean value obtained by HWOA-1 with GPS and NCF strategies for solving the BRdata standard instance is superior to those obtained by WOA. HWOA-2, with the addition of N1 and N2 neighborhood structures, has significantly better results than WOA and HWOA-1. Furthermore, the HWOA algorithm with the addition of the N3 neighborhood structure strategy does have the best results relative to the other three algorithms.

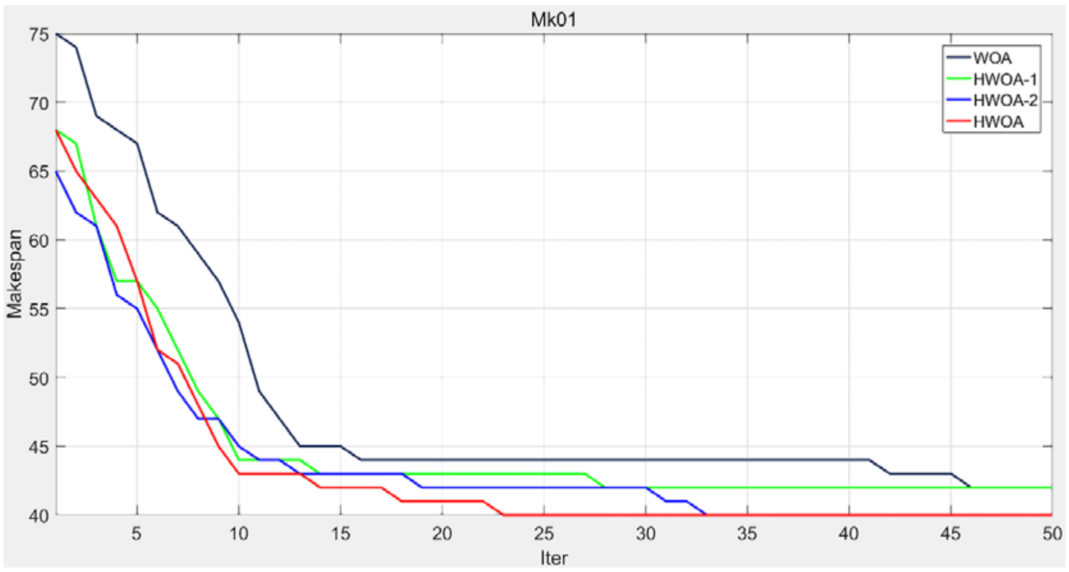


Figure 15. MK01 strategy comparison Convergence Curve.

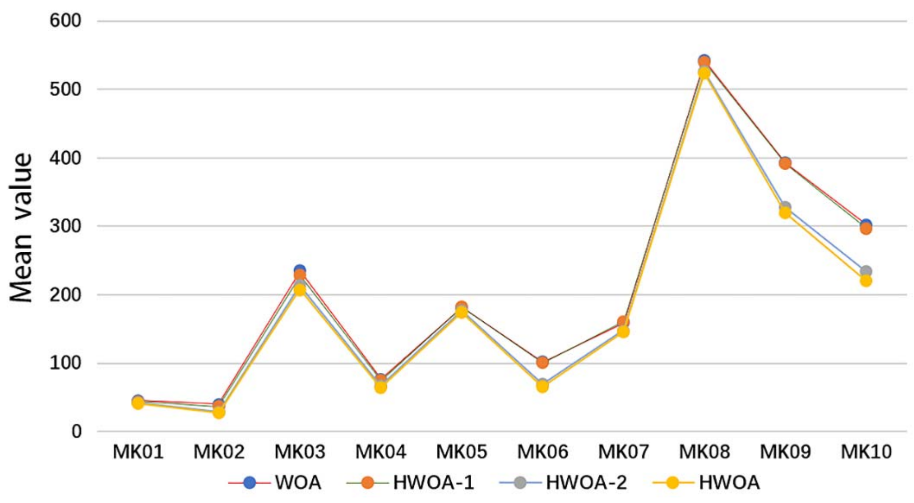


Figure 16. Comparison of mean values of four algorithms for BRdata instance.

It can be seen in Table 10 that the HWOA algorithm achieves the best results for 15 international benchmark instances of different sizes. For the five international benchmark instances of Kacem and the 10 international benchmark instances of BRdata, the percentage improvement of the results obtained by HWOA relative to those obtained by WOA reaches 16.19%, HWOA-2 14.81%, and HWOA-1 2.95%, all of which show an improvement.

Table 10. Comparison of four algorithms.

Kacem, BRdata	**n × m**	**LB**	**WOA**		**HWOA-1**		**HWOA-2**		**HWOA**	
			MPRD	***Pro*(%)**	***MPRD***	***Pro*(%)**	***MPRD***	***Pro*(%)**	***MPRD***	***Pro*(%)**
Kac01	4 × 5	11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Kac02	8 × 8	14	57.14	0.00	14.28	27.27	0.00	36.36	0.00	36.36
Kac03	10 × 7	11	27.27	0.00	27.27	0.00	0.00	21.42	0.00	21.42
Kac04	10 × 10	7	14.28	0.00	0.00	12.50	0.00	12.50	0.00	12.50
Kac05	15 × 10	11	27.27	0.00	27.27	0.00	9.09	14.28	0.00	21.42
MK01	10 × 6	36	5.00	0.00	5.00	0.00	0.00	4.76	0.00	4.76
MK02	10 × 6	24	26.92	0.00	26.92	0.00	3.84	18.18	0.00	21.21
MK03	15 × 8	204	13.23	0.00	9.31	3.46	0.00	11.68	0.00	11.68

Table 10. *Cont.*

Kacem, BRdata	**n × m**	**LB**	**WOA**		**HWOA-1**		**HWOA-2**		**HWOA**	
			MPRD	***Pro*(%)**	***MPRD***	***Pro*(%)**	***MPRD***	***Pro*(%)**	***MPRD***	***Pro*(%)**
MK04	15 × 8	48	17.74	0.00	17.74	0.00	3.22	12.32	0.00	15.06
MK05	15 × 4	168	1.15	0.00	1.15	0.00	0.00	1.14	0.00	1.14
MK06	10 × 15	33	58.06	0.00	58.06	0.00	4.83	33.67	0.00	36.73
MK07	20 × 5	133	7.69	0.00	8.39	−0.64	1.39	5.84	0.00	7.14
MK08	20 × 10	523	1.53	0.00	1.53	0.00	0.00	15.06	0.00	15.06
MK09	20 × 10	299	22.25	0.00	19.06	−1.05	0.64	17.67	0.00	18.20
MK10	20 × 15	165	25.46	0.00	22.68	2.21	3.70	17.34	0.00	20.29
Mean	-	-	20.33	0.00	15.91	2.95	1.78	14.81	0.00	16.19

We show two Gantt charts for solving the Kacem instance and the BRdata instance using HWOA in Figures 17 and 18, respectively. These two instances are the Kacem03 instance and the MK10 instance, with problem sizes of 10 machines, seven jobs, and 15 machines, 20 jobs, respectively.

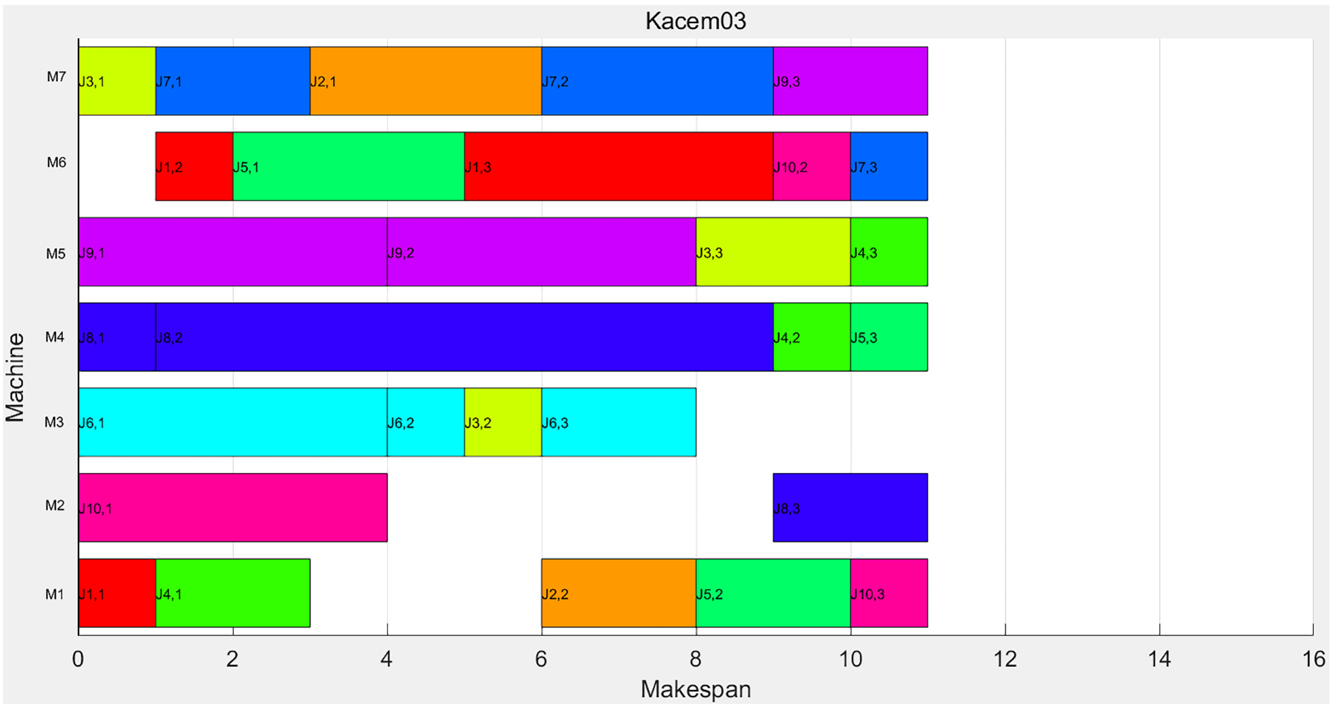


Figure 17. Kacem03 Gantt chart.

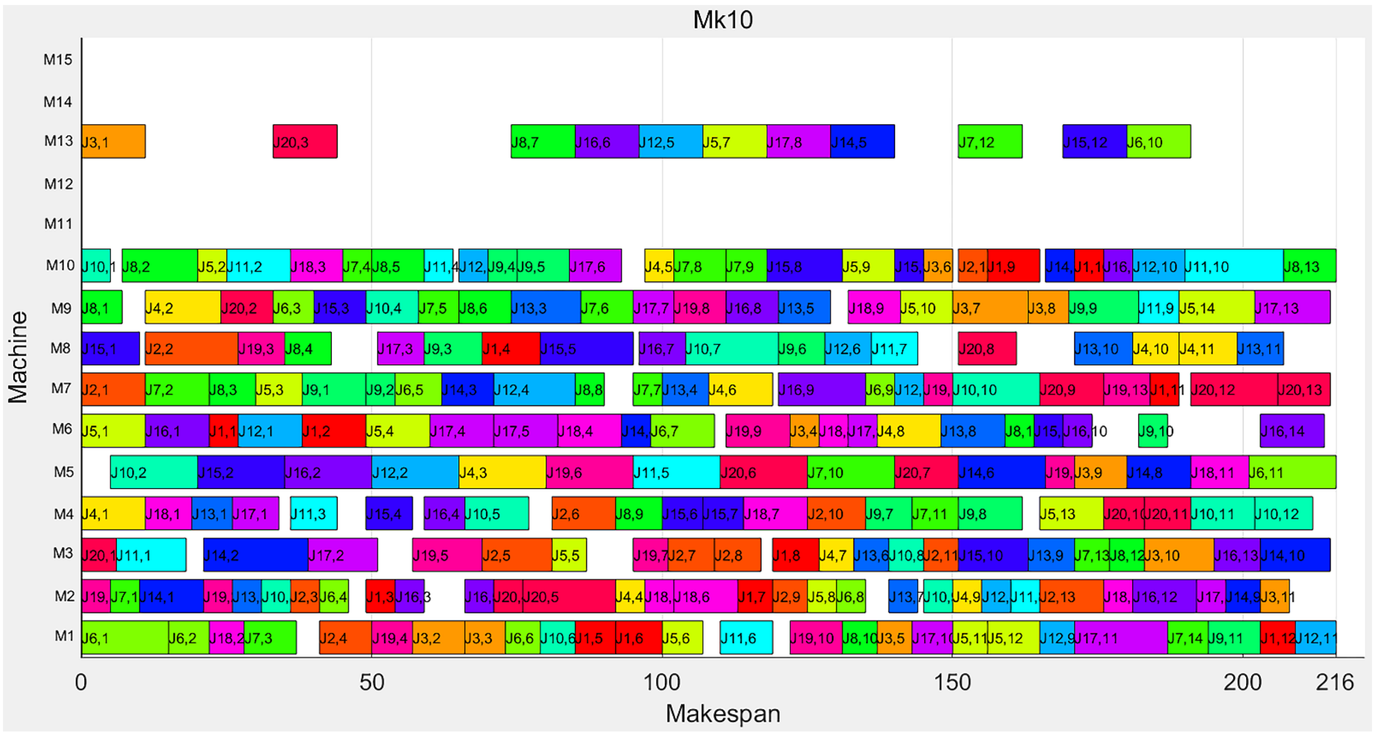


Figure 18. MK10 Gantt chart.

5.3.3. Performance Verification

In the following tests, the HWOA is compared with several algorithms from the literature on different international benchmark instances. In addition to the comparison with the better solutions obtained by each algorithm, the *RPD* tests on the results of the different algorithms for all instances are also performed, which is calculated by Equation (29), where *LB* is the lower limit of the test instance and C_{min} is the minimum makespan of the instance obtained from multiple runs of the current algorithm.

$$RPD = \frac{C_{min} - LB}{LB} \times 100 \tag{29}$$

In order to minimize the influence of the randomness of the algorithm, the following comparison experiments were repeated over 20 runs.

Comparison Experiment 1

To further validate the performance of the HWOA, a set of FJSP instances proposed by Brandimate are used in Experiment 1, which has a total of 10 medium-sized FJSPs. Algorithms were introduced by several scholars for comparison, such as OPSO [6], PPSO [37], KBACO [38], Heuristic [39], IWOA [40], and NIMASS [41]. The results are presented in Tables 11 and 12.

As can be seen from Table 11, HWOA achieves optimal solutions for five of the 10 instances compared with the other six algorithms, and the remaining five achieve suboptimal solutions, which is excellent overall. As for the *PRD* in Table 12, it can be concluded that the HWOA proposed in this paper has the smallest overall mean *PRD* value among the results obtained by the seven algorithms, which can also be seen in that the international known lower bound solutions are found in MK03 and MK08, and the deviations of the better solutions found by HWOA from the internationally known lower bound solutions of the four instances of MK02, MK05, MK07, and MK09 are also small, at between 2% and 9%. A BRdata instances box plot of HWOA and the other six algorithms is shown in Figure 19.

Table 11. Comparison of different algorithms in BRdata.

BRdata	n × m	OPSO	PPSO	KBACO	Heuristic	IWOA	NIMASS	HWOA
		C_{min}	C_{min}	C_{min}	C_{min}	C_{min}	C_{min}	C_{min}
MK01	10 × 6	41	40	39	42	40	40	40
MK02	10 × 6	26	29	29	28	26	28	26
MK03	15 × 8	207	204	204	204	204	204	204
MK04	15 × 8	65	66	65	75	60	65	62
MK05	15 × 4	171	175	173	179	175	177	173
MK06	10 × 15	61	77	67	69	63	67	62
MK07	20 × 5	173	145	144	149	144	144	143
MK08	20 × 10	523	523	523	555	523	523	523
MK09	20 × 10	307	320	311	342	339	312	310
MK10	20 × 15	312	239	229	242	242	229	216

Table 12. RPD value validation for comparison algorithm in BRdata.

BRdata	n × m	OPSO	PPSO	KBACO	Heuristic	IWOA	NIMASS	HWOA
		RPD	*RPD*	*RPD*	*RPD*	*RPD*	*RPD*	*RPD*
MK01	10 × 6	11.11	11.11	8.33	16.67	11.11	11.11	11.11
MK02	10 × 6	12.5	20.83	20.86	16.67	8.33	20.86	8.33
MK03	15 × 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MK04	15 × 8	29.16	37.50	35.41	56.25	25.00	35.41	29.16
MK05	15 × 4	5.95	4.16	2.97	6.54	4.16	5.35	2.97
MK06	10 × 15	136.36	133.33	103.03	109.09	90.90	103.03	87.87
MK07	20 × 5	10.52	9.02	8.27	12.03	8.27	8.27	7.51
MK08	20 × 10	0.00	0.00	0.00	6.12	0.00	0.00	0.00
MK09	20 × 10	14.04	7.02	4.01	14.38	13.37	4.34	3.67
MK10	20 × 15	50.91	44.84	38.78	46.66	46.66	38.78	30.90
Mean	-	27.05	26.78	22.16	28.44	20.78	22.71	18.15

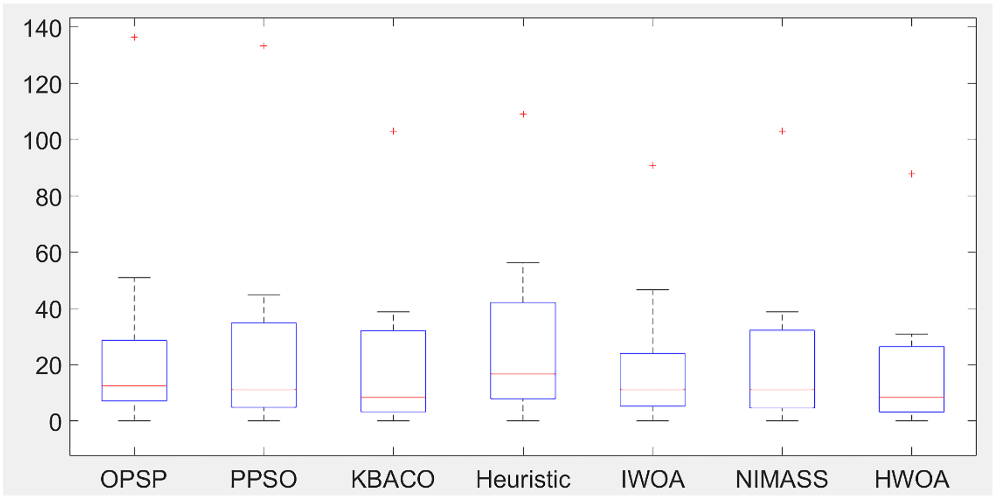


Figure 19. BRdata instances of seven algorithms box plot.

Comparison Experiment 2

The Kacem instances are introduced to reflect the effectiveness of the HWOA algorithm, which is compared with the four algorithms KBACO [38], HGWO [42], IWOA [40], and EQEA [4]. The experiment results are listed in Tables 13 and 14. In HWOA, KBACO, and EQEA, the three algorithms perform well in both C_{min} and RPD.

Table 13. Comparison of different algorithms in Kacem.

Kacem	n × m	KBACO	HGWO	IWOA	EQEA	HWOA
		C_{min}	C_{min}	C_{min}	C_{min}	C_{min}
Kac01	4 × 5	11	11	11	11	11
Kac02	8 × 8	14	14	14	14	14
Kac03	10 × 7	11	11	13	11	11
Kac04	10 × 10	7	7	7	7	7
Kac05	15 × 10	11	13	14	11	11

Table 14. RPD value validation for comparison algorithm in Kacem.

Kacem	n × m	KBACO	HGWO	IWOA	EQEA	HWOA
		RPD	*RPD*	*RPD*	*RPD*	*RPD*
Kac01	4 × 5	0.00	0.00	0.00	0.00	0.00
Kac02	8 × 8	0.00	0.00	0.00	0.00	0.00
Kac03	10 × 7	0.00	0.00	18.18	0.00	0.00
Kac04	10 × 10	0.00	0.00	0.00	0.00	0.00
Kac05	15 × 10	0.00	18.18	27.27	0.00	0.00

Comparison Experiment 3

Fattahin instances are selected to verify the performance of the HWOA, which contains a total of 18 FJSPs, and the results of comparing HWOA with the following six algorithms: FPA [43], M2 [44], HA [16], GOA [45], AIA [46], and EPSO [47] are presented in Tables 15 and 16. As can be seen, HWOA achieves known lower bound solutions in 10 instances, and achieves well-known solutions together with HA and EPSO in eight instances. Overall, HWOA outperforms the four algorithms FPA, M2, GOA, and AIA in the Fattahin instances test.

Table 15. Comparison of different algorithms in Fattahin.

Fattahin	n × m	LB	FPA	M2	HA	GOA	AIA	EPSO	HWOA
			C_{min}	C_{min}	C_{min}	C_{min}	C_{min}	C_{min}	C_{min}
SFJS01	2 × 2	66	66	66	66	66	66	66	66
SFJS02	2 × 2	107	107	107	107	107	107	107	107
SFJS03	3 × 2	221	221	221	221	221	221	221	221
SFJS04	3 × 2	355	355	355	355	355	355	355	355
SFJS05	3 × 2	119	119	119	119	119	119	119	119
SFJS06	3 × 3	320	320	320	320	320	320	320	320
SFJS07	3 × 5	397	397	397	397	397	397	397	397
SFJS08	3 × 4	253	253	253	253	253	253	253	253
SFJS09	3 × 3	210	210	210	210	210	210	210	210
SFJS10	4 × 5	516	516	516	516	533	516	516	516
MFJS01	5 × 6	396	469	468	468	469	468	468	468
MFJS02	5 × 7	396	446	446	446	457	448	446	446
MFJS03	6 × 7	396	470	466	466	538	468	466	466
MFJS04	7 × 7	496	554	564	554	610	554	554	554
MFJS05	7 × 7	414	516	514	514	613	527	514	514
MFJS06	8 × 7	469	636	634	634	721	635	634	634
MFJS07	8 × 7	619	879	928	879	1092	879	879	879
MFJS08	9 × 8	619	884	-	884	1095	884	884	884

- Means the result was not given by the author.

Table 16. RPD value validation for comparison algorithm in Fattahin.

Fattahin	**n × m**	**LB**	**FPA**	**M2**	**HA**	**GOA**	**AIA**	**EPSO**	**HWOA**
			RPD	***RPD***	***RPD***	***RPD***	***RPD***	***RPD***	***RPD***
SFJS01	2 × 2	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS02	2 × 2	107	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS03	3 × 2	221	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS04	3 × 2	355	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS05	3 × 2	119	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS06	3 × 3	320	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS07	3 × 5	397	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS08	3 × 4	253	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS09	3 × 3	210	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SFJS10	4 × 5	516	0.00	0.00	0.00	3.29	0.00	0.00	0.00
MFJS01	5 × 6	396	18.43	18.18	18.18	18.43	18.18	18.18	18.18
MFJS02	5 × 7	396	12.62	12.62	12.62	15.40	13.13	12.62	12.62
MFJS03	6 × 7	396	18.68	17.67	17.67	35.85	18.18	17.67	17.67
MFJS04	7 × 7	496	11.69	13.71	11.69	22.98	11.69	11.69	11.69
MFJS05	7 × 7	414	24.63	24.15	24.15	48.06	27.29	24.15	24.15
MFJS06	8 × 7	469	35..60	35.18	35.18	53.73	35.39	35.18	35.18
MFJS07	8 × 7	619	42.00	49.92	42.00	76.41	42.00	42.00	42.00
MFJS08	9 × 8	619	42.81	-	42.81	76.89	42.81	42.81	42.81

- Means the result was not given by the author.

Comparison Experiment 4

Forty instances of the Hurink benchmark with high flexibility are selected. Since no lower limit is published for this instance, the well-known solutions obtained by other scholars to solve these instances are used as the lower limit for comparison. This experiment uses the results obtained for these instances in the HA [15] published by Li and Gao in 2016 as the effective lower bound benchmark. In Table 17, the results of the comparison with the following three algorithms are included: N1-1000 [48], N2-1000 [48], and IJA [12].

Table 17. RPD value validation for comparison algorithm in Hurink (Vdata).

Vdata	**N1-1000**	**N2-1000**	**IJA**	**HWOA**
	RPD	***RPD***	***RPD***	***RPD***
La01–La05	0.78	0.59	0.00	0.00
La06–La10	0.20	0.15	0.00	0.00
La11–La15	0.20	0.40	0.00	0.00
La16–La20	0.00	0.00	0.00	0.00
La21–La25	6.90	1.97	0.77	0.64
La26–La30	0.26	0.25	0.13	0.09
La31–La35	0.06	0.01	0.01	0.01
La36–La40	0.00	0.00	0.00	0.00
Mean	1.050	0.421	0.114	0.093

As can be seen from the results in Table 17, HWOA achieves good results in the Hurink benchmark instances with a high degree of flexibility. The results obtained in most instances are consistent with the lower bound value, while the percentage deviation of the solution obtained from the HWOA solution with respect to the well-known solution is smaller than that obtained by the other three algorithms. Based on the comparison results of these instances, it can be concluded that HWOA performs equally well in solving highly flexible problems.

Comparison Experiment 5

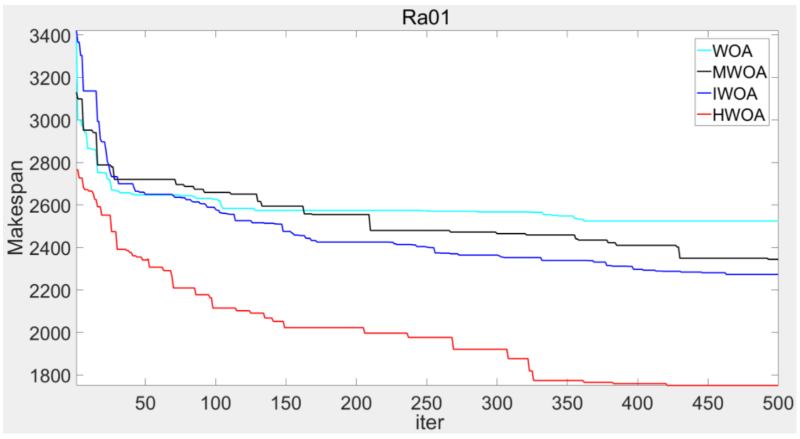
Four randomly generated large-scale FJSP instances are tested by HWOA, which is compared with WOA [24], MWOA [49], and IWOA [40]. The results of the comparative test for this instance are presented in Table 18.

Table 18. Comparison of different algorithms in Ra.

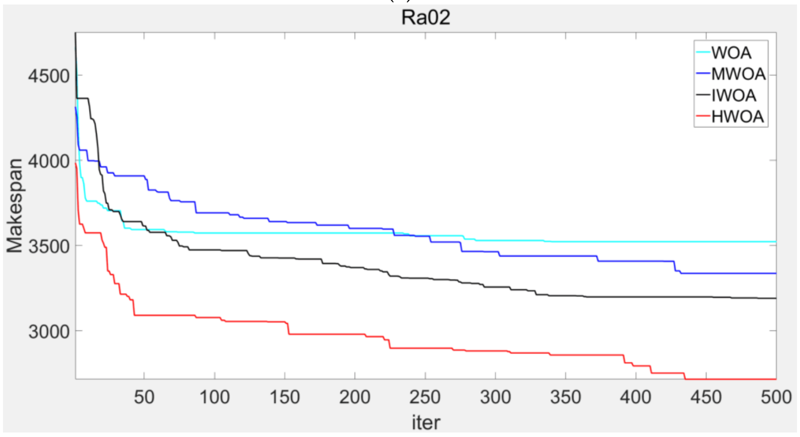
Ra	**n × m**	**WOA**			**MWOA**			**IWOA**			**HWOA**		
		C_{min}	C_{mean}	*CPU*	C_{min}	C_{mean}	*CPU*	C_{min}	C_{mean}	*CPU*	C_{min}	C_{mean}	*CPU*
Ra01	30 × 15	2524	2664.6	152.77	2344	2419.2	106.13	2273	2397.2	169.36	**1751**	**1857.7**	**96.81**
Ra02	45 × 15	3522	3691.2	203.04	3336	3417.5	**114.91**	3190	3354.1	223.03	**2715**	**3939.2**	116.45
Ra03	50 × 10	3269	3382.4	152.77	3090	3168.2	**95.82**	3007	3193.4	169.36	**2734**	**2878.4**	98.84
Ra04	60 × 15	4454	4660.6	283.36	4261	4318.2	153.72	3947	4231.9	310.01	**3669**	**3836.3**	**137.17**

The best results are highlighted in boldface.

The convergence curve of the four algorithms, WOA, MWOA, IWOA, and HWOA solving Ra instances is depicted in Figure 20. As can be seen, HWOA achieves the best results among the four algorithms. The reasons for this are as follows: First, the good point set population initialization mechanism assists HWOA in finding a good solution space at the beginning of the iteration, resulting in rapid convergence and reduced running time; second, the multi-neighborhood structure in the iterative operation ensures that HWOA keeps breaking through the dilemma to avoid falling into a local optimum; and third, the diversity reception mechanism also ensures the population diversity also of HWOA, which improves computational accuracy. The scheduling Gantt chart for Ra instances is shown in Figure 21.

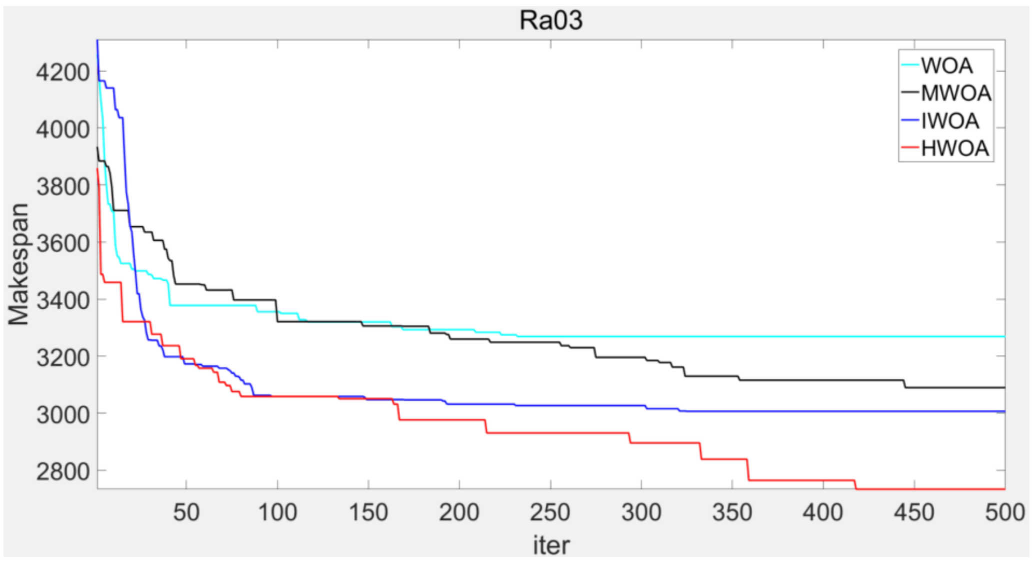


(**a**)

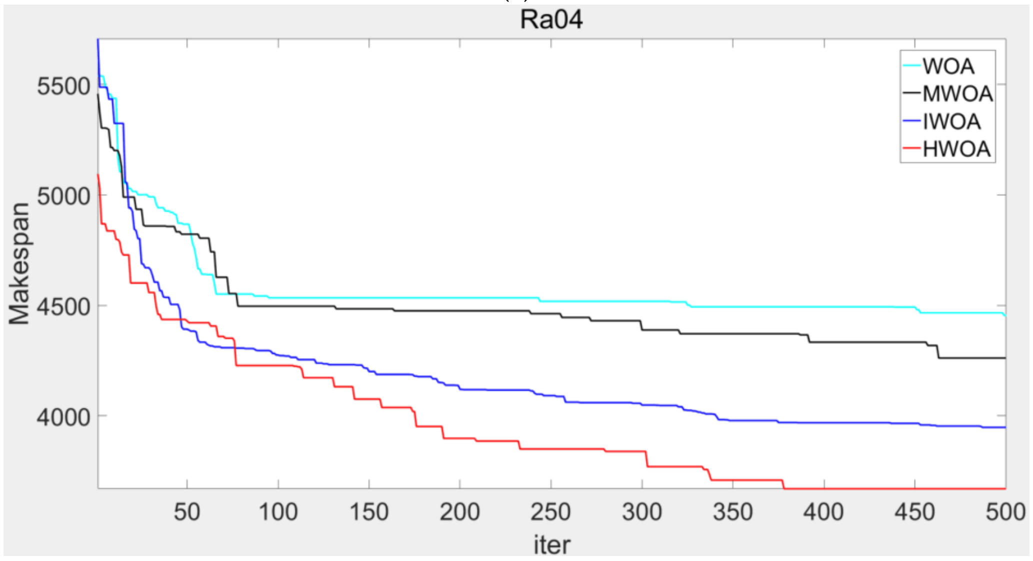


(**b**)

Figure 20. *Cont.*

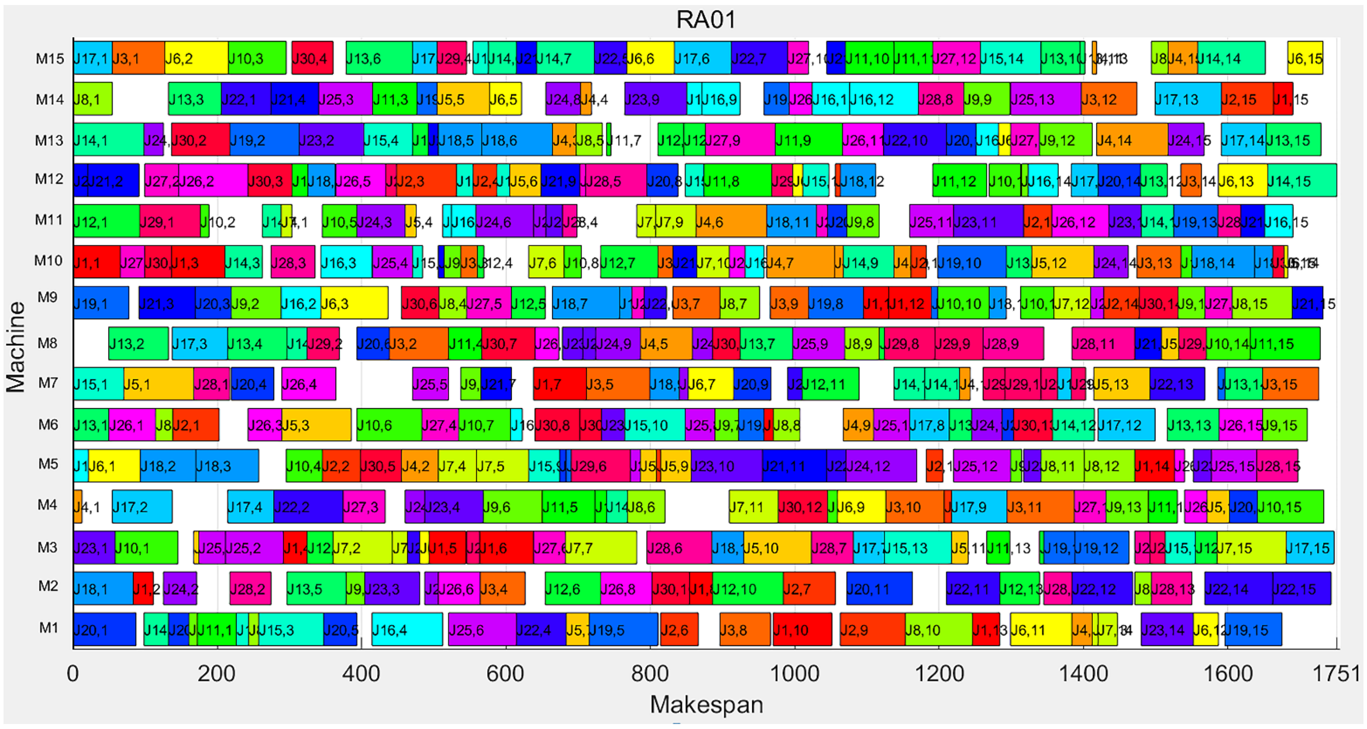


(**c**)

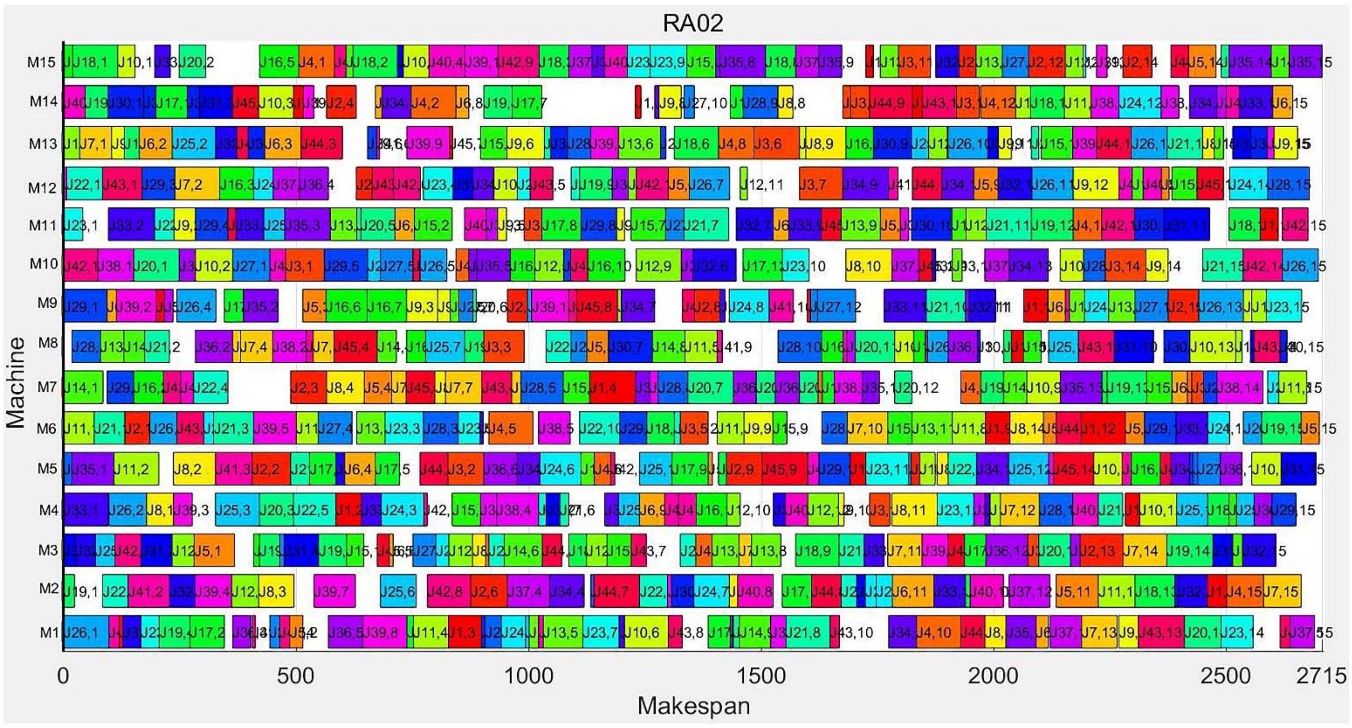


(**d**)

Figure 20. (**a**) Ra01 Convergence Curve; (**b**) Ra02 Convergence Curve; (**c**) Ra03 Convergence Curve; (**d**) Ra04 Convergence Curve.

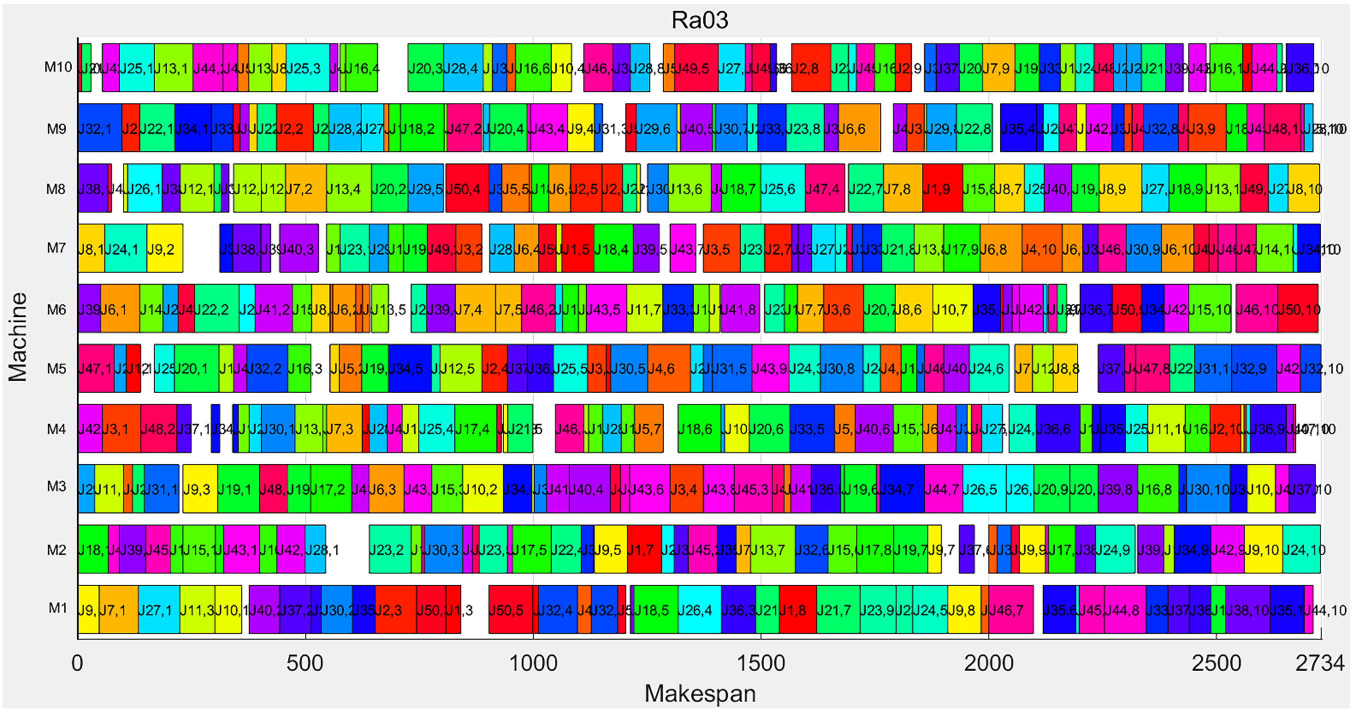


(**a**)

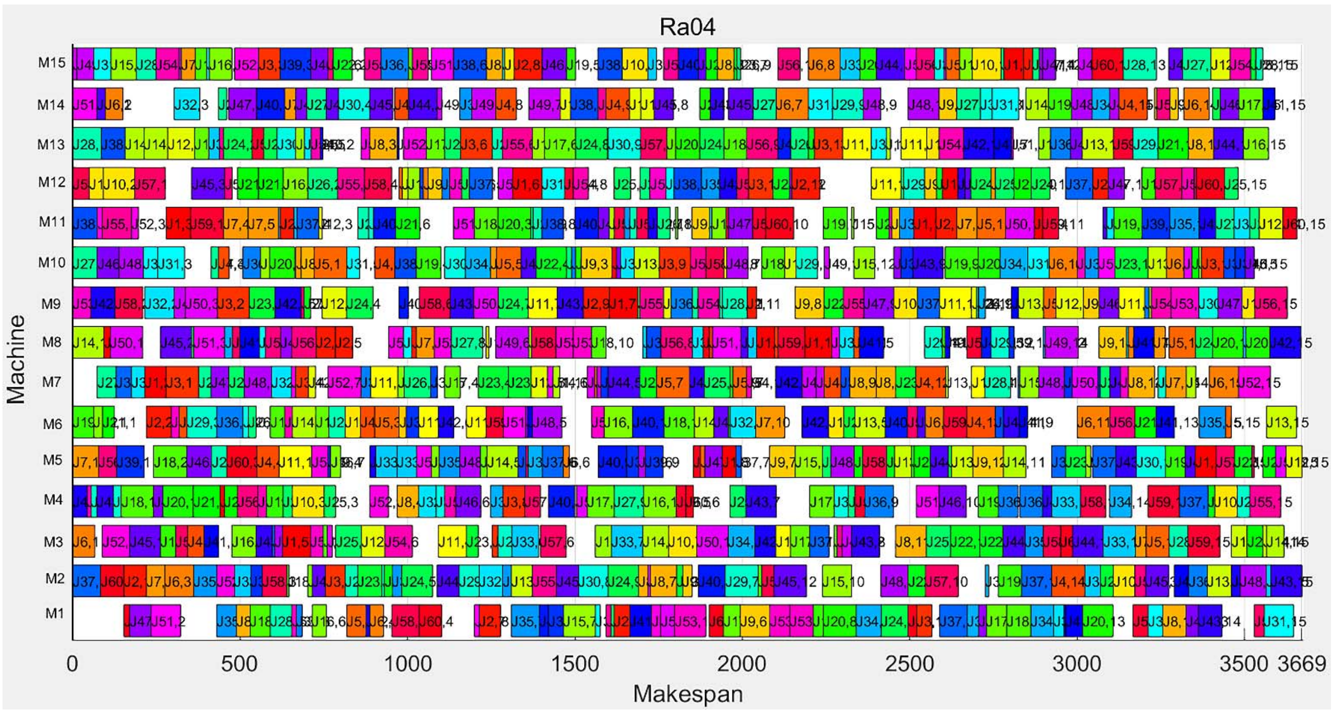


(**b**)

Figure 21. *Cont.*



(**c**)



(**d**)

Figure 21. (**a**) Ra01 Gantt chart; (**b**) Ra02 Gantt chart; (**c**) Ra03 Gantt chart; (**d**) Ra04 Gantt chart.

6. Conclusions

In this study, a novel hybrid whale optimization algorithm (HOWA) is proposed to solve the flexible job shop scheduling problem with the objective of minimizing the maximum makespan. The Whale Optimization Algorithm (WOA) is a new intelligent algorithm commonly used to solve optimization problems, and the WOA has been validated by most scholars in solving continuous problems. In this study, WOA is used to solve the discrete FJSP by a certain transformation. A variety of effective strategies are added to WOA, namely HWOA, to make it better for solving FJSP. Firstly, the initial population is generated by introducing the theory of good point set (GPS) to make the distribution of the initial population more uniform and have more comprehensive coverage in the initial stage. Secondly, in order to make the convergence factor a in WOA play a better coordination role, a nonlinear convergence factor (NCF) a is proposed to coordinate the global search and local search in WOA. Thirdly, a new multi-neighborhood structure (MNS) is proposed, within which a total of three new neighborhoods are included. The N1 neighborhood structure optimizes the scheduling solution in terms of different selections of machine pairings, the N2 neighborhood structure optimizes the scheduling solution in terms of the operation update mechanism; and the N3 neighborhood structure is used to enhance the local search capability of HWOA. Finally, a population diversity reception mechanism (DRM) that ensures to some extent that populations do not lose population diversity with iteration. Finally, seven international standard functions, including 30-, 50-, and 100-dimensions, are used to test the HWOA, and the results show that HWOA has a strong search ability on almost all the tested functions. 73 international benchmark instances of FJSP of different sizes and flexibility, and four randomly generated large-scale FJSP instances are used to perform performance tests on HWOA and verify the effectiveness of HWOA, which further demonstrates that HWOA shows strong competitiveness. Especially in the BRdata instance, HWOA improves by 36.73% relative to WOA, and in the Kacem instance, HWOA improves by 36.36% relative to WOA, with an average improvement rate of 16.19% in both instances. With reference to the experimental results, the maximum Pro of HWOA relative to OPSP, PPSO, KBACO, Heuristic, IWOA, and NIMASS is 44.44%, 24.19%, 11.54%, 20.97%, 12.04%, and 8.06%, respectively. To sum up, HWOA has strong robustness. For future research, multiple objectives of FJSP would be considered to better adapt to the complicated scenarios of enterprise production.

Author Contributions: Writing—review & editing, Z.Y.; Methodology, W.Y.; Writing—original draft, J.S.; Validation, Y.Y. (Yunhang Yao); Investigation, Y.Y. (Ying Yuan). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Project of China (2018YFB1700500) and the Scientific and Technological Project of Henan Province (222102110095), Higher Learning Key Development Project of Henan Province (22A120007).

Data Availability Statement: Reasonable requests to access the datasets should be directed to zl.yang@siat.ac.cn.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, H.; Ihlow, J.; Lehmann, C. A genetic algorithm for flexible job-shop scheduling. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Detroit, MI, USA, 10–15 May 1999; pp. 1120–1125.
2. Du, X.; Li, Z.; Xiong, W. Flexible Job Shop scheduling problem solving based on genetic algorithm with model constraints. In Proceedings of the 2008 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore, 8–11 December 2008; pp. 1239–1243.
3. Bowman, E.H. The schedule-sequencing problem. *Oper. Res.* **1959**, *7*, 621–624. [CrossRef]
4. Wu, X.; Wu, S. An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem. *J. Intell. Manuf.* **2017**, *28*, 1441–1457. [CrossRef]
5. Brucker, P.; Schlie, R. Job-shop scheduling with multi-purpose machines. *Computing* **1990**, *45*, 369–375. [CrossRef]
6. Nouiri, M.; Bekrar, A.; Jemai, A.; Niar, S.; Ammari, A.C. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *J. Intell. Manuf.* **2015**, *29*, 603–615. [CrossRef]
7. Huang, X.; Chen, S.; Zhou, T.; Sun, Y. A new neighborhood structure for solving the flexible job-shop scheduling problem. *Syst. Eng.-Theory Pract.* **2021**, *41*, 2367–2378.
8. Xue, L. Block structure neighborhood search genetic algorithm for job-shop scheduling. *Comput. Integr. Manuf. Syst.* **2021**, *27*, 2848–2857.

9. Driss, I.; Mouss, K.N.; Laggoun, A. A new genetic algorithm for flexible job-shop scheduling problems. *J. Mech. Sci. Technol.* **2015**, *29*, 1273–1281. [CrossRef]
10. Jiang, T.; Zhang, C. Application of Grey Wolf Optimization for Solving Combinatorial Problems: Job Shop and Flexible Job Shop Scheduling Cases. *IEEE Access* **2018**, *6*, 26231–26240. [CrossRef]
11. Liang, X.; Huang, M.; Ning, T. Flexible job shop scheduling based on improved hybrid immune algorithm. *J. Ambient Intell. Humaniz. Comput.* **2016**, *9*, 165–171. [CrossRef]
12. Caldeira, R.H.; Gnanavelbabu, A. Solving the flexible job shop scheduling problem using an improved Jaya algorithm. *Comput. Ind. Eng.* **2019**, *137*, 106064. [CrossRef]
13. Wu, M.; Yang, D.; Zhou, B.; Yang, Z.; Liu, T.; Li, L.; Wang, Z.; Hu, K. Adaptive population nsga-iii with dual control strategy for flexible job shop scheduling problem with the consideration of energy consumption and weight. *Machines* **2021**, *9*, 344. [CrossRef]
14. Zhang, G.; Gao, L.; Shi, Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Syst. Appl.* **2011**, *38*, 3563–3573. [CrossRef]
15. Kacem, I.; Hammadi, S.; Borne, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Trans. Syst. Man Cybern. Part C* **2002**, *32*, 1–13. [CrossRef]
16. Li, X.; Gao, L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int. J. Prod. Econ.* **2016**, *174*, 93–110. [CrossRef]
17. Zhang, S.; Du, H.; Borucki, S.; Jin, S.; Hou, T.; Li, Z. Dual resource constrained flexible job shop scheduling based on improved quantum genetic algorithm. *Machines* **2021**, *9*, 108. [CrossRef]
18. Singh, M.R.; Mahapatra, S.S. A quantum behaved particle swarm optimization for flexible job shop scheduling. *Comput. Ind. Eng.* **2016**, *93*, 36–44. [CrossRef]
19. Wang, L.; Cai, J.; Li, M.; Liu, Z. Flexible Job Shop Scheduling Problem Using an Improved Ant Colony Optimization. *Sci. Program.* **2017**, *2017*, 9016303. [CrossRef]
20. Cruz-Chávez, M.A.; Martínez-Rangel, M.G.; Cruz-Rosales, M.H. Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *Int. Trans. Oper. Res.* **2017**, *24*, 1119–1137. [CrossRef]
21. Chenyang, G.; Yuelin, G.; Shanshan, L. Improved simulated annealing algorithm for flexible job shop scheduling problems. In Proceedings of the 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, China, 28–30 May 2016; pp. 2191–2196.
22. Vijaya Lakshmi, A.; Mohanaiah, P. WOA-TLBO: Whale optimization algorithm with Teaching-learning-based optimization for global optimization and facial emotion recognition. *Appl. Soft Comput.* **2021**, *110*, 107623. [CrossRef]
23. Medani, K.b.O.; Sayah, S.; Bekrar, A. Whale optimization algorithm based optimal reactive power dispatch: A case study of the Algerian power system. *Electr. Power Syst. Res.* **2018**, *163*, 696–705. [CrossRef]
24. Huang, X.; Wang, R.; Zhao, X.; Hu, K. Aero-engine performance optimization based on whale optimization algorithm. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 11437–11441.
25. Yan, Q.; Wu, W.; Wang, H. Deep Reinforcement Learning for Distributed Flow Shop Scheduling with Flexible Maintenance. *Machines* **2022**, *10*, 210. [CrossRef]
26. Dao, T.-K.; Pan, T.-S.; Pan, J.-S. A multi-objective optimal mobile robot path planning based on whale optimization algorithm. In Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 6–10 November 2016; pp. 337–342.
27. Oliva, D.; Abd El Aziz, M.; Hassanien, A.E. Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm. *Appl. Energy* **2017**, *200*, 141–154. [CrossRef]
28. Liang, R.; Chen, Y.; Zhu, R. A novel fault diagnosis method based on the KELM optimized by whale optimization algorithm. *Machines* **2022**, *10*, 93. [CrossRef]
29. Abdel-Basset, M.; Manogaran, G.; El-Shahat, D.; Mirjalili, S. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Gener. Comput. Syst.* **2018**, *85*, 129–145. [CrossRef]
30. Liu, M.; Yao, X.; Li, Y. Hybrid whale optimization algorithm enhanced with Lévy flight and differential evolution for job shop scheduling problems. *Appl. Soft Comput.* **2020**, *87*, 105954.
31. Luan, F.; Cai, Z.; Wu, S.; Liu, S.Q.; He, Y. Optimizing the Low-Carbon Flexible Job Shop Scheduling Problem with Discrete Whale Optimization Algorithm. *Mathematics* **2019**, *7*, 688. [CrossRef]
32. Yazdani, M.; Amiri, M.; Zandieh, M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Syst. Appl.* **2010**, *37*, 678–687. [CrossRef]
33. Hua, L.; Wang, Y. *Application of Number Theory in Approximate Analysis*; Science Press: Bejing, China, 1978; p. 248.
34. Lv, H.; Feng, Z.; Wang, X.; Zhou, W.; Chen, B. Structural damage identification based on hybrid whale annealing algorithm and sparse regularization. *J. Vib. Shock.* **2021**, *40*, 85–91.
35. Yang, W.; Yang, Z.; Chen, Y.; Peng, Z. Modified Whale Optimization Algorithm for Multi-Type Combine Harvesters Scheduling. *Machines* **2022**, *10*, 64. [CrossRef]
36. Brandimarte, P. Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.* **1993**, *41*, 157–183.
37. Ding, H.; Gu, X. Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. *Comput. Oper. Res.* **2020**, *121*, 104951. [CrossRef]
38. Xing, L.-N.; Chen, Y.-W.; Wang, P.; Zhao, Q.-S.; Xiong, J. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. *Appl. Soft Comput.* **2010**, *10*, 888–896. [CrossRef]

39. Ziaee, M. A heuristic algorithm for solving flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2013**, *71*, 519–528. [CrossRef]
40. Luan, F.; Cai, Z.; Wu, S.; Jiang, T.; Li, F.; Yang, J. Improved Whale Algorithm for Solving the Flexible Job Shop Scheduling Problem. *Mathematics* **2019**, *7*, 384. [CrossRef]
41. Xiong, W.; Fu, D. A new immune multi-agent system for the flexible job shop scheduling problem. *J. Intell. Manuf.* **2015**, *29*, 857–873. [CrossRef]
42. Jiang, T. Flexible job shop scheduling problem with hybrid grey wolf optimization algorithm. *Control Decis.* **2018**, *33*, 503–508.
43. Phuang, A. The flower pollination algorithm with disparity count process for scheduling problem. In Proceedings of the 2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE), Phuket, Thailand, 12–13 October 2017; pp. 1–5.
44. Demir, Y.; İşleyen, S.K. Evaluation of mathematical models for flexible job-shop scheduling problems. *Appl. Math. Model.* **2013**, *37*, 977–988. [CrossRef]
45. Feng, Y.; Liu, M.; Yang, Z.; Feng, W.; Yang, D. A Grasshopper Optimization Algorithm for the Flexible Job Shop Scheduling Problem. In Proceedings of the 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Zhanjiang, China, 16–18 October 2020; pp. 873–877.
46. Bagheri, A.; Zandieh, M.; Mahdavi, I.; Yazdani, M. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Gener. Comput. Syst.* **2010**, *26*, 533–541. [CrossRef]
47. Teekeng, W.; Thammano, A.; Unkaw, P.; Kiatwuthiamorn, J. A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization. *Artif. Life Robot.* **2016**, *21*, 18–23. [CrossRef]
48. Hurink, J.; Jurisch, B.; Thole, M. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Oper.-Res.-Spektrum* **1994**, *15*, 205–215. [CrossRef]
49. Wei, F.; Cao, C.; Zhang, H. An Improved Genetic Algorithm for Resource-Constrained Flexible Job-Shop Scheduling. *Int. J. Simul. Model.* **2021**, *20*, 201–211. [CrossRef]