A Discrete Multi-Objective Rider Optimization Algorithm for Hybrid Flowshop Scheduling Problem Considering Makespan, Noise and Dust Pollution

YANMING FU¹, ZHUOHANG LI¹, NINGJIANG CHEN¹, (Member, IEEE), AND CHIWEN QU²,³

¹School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China
²College of Mathematics and Statistics, Hunan Normal University, Changsha 410081, China
³School of Information Engineering, Baise University, Baise 533000, China

Corresponding authors: Zhuohang Li (1072694960@qq.com) and Chiwen Qu (quchiwen@163.com)

This work was supported in part by the Natural Science Foundation of China under Grant 61762008, in part by the National Key Research and Development Project of China under Grant 2018YFB1404404, and in part by the Guangxi Natural Science Foundation Project under Grant 2017GXNSFAA198141.

ABSTRACT Many optimization algorithms have been proposed to solve hybrid flowshop scheduling problem (HFSP). However, with the development of industry and society, labor right and labor safety have become important problem to consider in production scheduling. So the green HFSP considering makespan, noise and dust pollution becomes an urgent problem to be solved. In this paper, the rider optimization algorithm (ROA) is modified into the multi-objective rider optimization algorithm (MOROA) using Pareto archive and neighborhood sorting techniques. The Pareto archive and neighborhood sorting technology make the Pareto optimal solution set of MOROA have higher coverage and more solutions. Then MOROA is discretized into discrete MOROA (DMOROA) to solve the HFSP considering makespan, noise and dust pollution. DMOROA is tested on 10, 30 and 50 jobs HFSP considering makespan, noise and dust pollution. The test results are compared with two multi-objective algorithms to verify the performance of DMOROA. And the test results verify that the DMOROA is superior to the comparison algorithms in search accuracy, number of non-dominated solutions, diversity of solution set and stability. Therefore, DMOROA is effective in solving multi-objective HFSP considering makespan, noise and dust pollution.

INDEX TERMS Hybrid flowshop scheduling problem, multi-objective meta-heuristic algorithm, rider optimization algorithm, labor right.

I. INTRODUCTION

The hybrid flowshop scheduling problem (HFSP) is a kind of combinatorial optimization problem which integrates the traditional flowshop scheduling and parallel machine scheduling [1]. HFSP is a complex problem in flowshop scheduling and is difficult to be solved. Complex HFSP is not suitable to be solved by precise algorithm, so heuristic algorithm is put forward to solve HFSP [2]–[5]. The goal of HFSP is to promote the production efficiency. Therefore, HFSP usually uses minimum manufacturing time, total completion time, total delay or other criterions about production efficiency to establish the objective functions [6].

The single object HFSP considering efficiency has been researched widely. Reference [7] applied immune algorithm to solve HFSP of flexible printed circuit board and obtained good effect. But the algorithm was proposed for specific applications and not easy to generalize to other applications. Literature [8] proposed an artificial immune algorithm to minimize the makespan of HFSP, however the stability was poor. Reference [9] presented a tabu search optimization algorithm to deal with the multiprocessor HFSP. The algorithm was easy to fall into local optimum. Literature [10] used ant colony algorithm to solve HFSP. In reference [11], the iterative greedy algorithm and a variable block heuristic algorithm were fused to solve the HFSP with minimum total flow time, yet only a few solutions were given. In reference [12], an improved particle swarm optimization
algorithm was developed based on constructing heuristic algorithm to solve HFSP with minimum maximum completion time and total flow time. However the Pareto front end was not uniform enough.

The multi-objective HFSP usually employs the economic indexes such as efficiency or energy consumption to establish the objective functions. Reference [13] presented a discrete multi-objective firefly algorithm to solve the HFSP considering makespan, total flow time and machine idle time. But there was no other comparison metrics except for coverage metric in the experiment. In reference [14], a multi-objective discrete artificial bee colony algorithm was proposed to solve the green HFSP with the maximum completion time and the minimum total energy consumption. However the coding method was complex and the efficiency was low. Literature [15] applied a multi-objective gray wolf optimization algorithm based on cellular automata and variable neighborhood search to optimize HFSP considering production efficiency, energy consumption and noise pollution. The experiment was comprehensive, but lacked the complex test problem. Reference [16] enhanced the multi-objective evolutionary algorithm with problem-specific heuristics to deal with the lot-streaming HFSP. This method was a new research direction and close to the reality, yet the algorithm had high time complexity. In literature [17], a discrete multi-objective firework algorithm using cluster analysis and fuzzy correlation entropy analysis was introduced to handle the HFSP with total production cost, maximum completion time, average running time and average idle time. Unfortunately, this research focused on static situation and could not deal with dynamic situation. Reference [18] developed a multi-objective multi-verse optimizer algorithm based on Levy flight and chaos local search to solve reentrant HFSP with minimum completion time, delay and energy consumption. The method had strong ability of machine scheduling, but its performance were not outstanding in the case of continuous scheduling. In literature [19], a multi-objective evolutionary algorithm was introduced to search the optimal solutions of hot-rolling scheduling problem in the compact strip production. Reference [20] employed a dual-objective evolutionary algorithm to solve stochastic hybrid flowshop deteriorating scheduling problem considering makespan and total tardiness.

With the development of economic globalization and the spread of people-oriented concept, more and more countries pay attention to labor right and labor safety, but the loss caused by occupational disease and injury is still huge every year [21]–[23]. Occupational diseases mainly refer to the diseases caused by workers’ exposure to dust, noise or radioactive substance in their occupational activities. According to a report released by the international labor organization in 2019 [24], about 6500 people die of occupational disease every day in the world, and the economic loss caused by occupational safety and health problem accounts for 4% of the global GDP [25], [26]. The report suggested that countries could use new technologies to reduce occupational risks and create new jobs through sustainable development and green economy to continuously improve occupational safety.

There are two modes of labor right protection. One mode is to protect labor right legally through state legislative intervention or regulation [27]. The other mode relies on the joint participation of labor and capital in enterprise decision-making to implement enterprise autonomy and labor self-protection [28], [29]. Due to the different levels of economic and social development in different countries, the levels of labor protection in national law are uneven. Therefore, labor protection needs a long time and a lot of financial investment, and it is not easy to implement. In this study, new decision-making model and optimization technique are applied to production planning and scheduling to improve the protection of labor right without any change of production process and equipment. This paper is devoted to solving the green HFSP considering makespan, noise and dust pollution.

Rider optimization algorithm (ROA) is a new heuristic algorithm proposed by D. Binu and B. S kariyapa in 2019. ROA optimizes the problem by assuming that a group of riders drive towards the goal to compete for the championship [30]. ROA has unique structure and advantage in dealing with complex problem. The ROA is modified into the multi-objective rider optimization algorithm (MOROA) using Pareto archive and neighborhood sorting techniques. The Pareto archive and neighborhood sorting technology make the Pareto optimal solution set of MOROA have higher coverage and more solutions. Then MOROA is discretized into discrete MOROA (DMOROA) to solve the HFSP considering makespan, noise and dust pollution. DMOROA and HFSP considering labor safety are both novel issues, and this paper can provide an enlightening idea for these fields.

The rest of this paper is arranged as follows. The section II gives the definition of HFSP considering makespan, noise and dust pollution. The section III describes the ROA and gives the pseudo code. The section IV describes how to change ROA to MOROA and gives the pseudo code. In section V, MOROA will be discretized into discrete MOROA and DMOROA is used to solve the HFSP considering makespan, noise and dust pollution. In the section VI, the simulation experiment is carried out, and the experiment results are analyzed according to the evaluation metrics. The section VII summarizes the whole paper and discusses the future work.

II. DEFINITION OF HFSP CONSIDERING MAKESPAN, NOISE AND DUST POLLUTION

The definition of HFSP considering makespan, noise and dust pollution is given by model hypothesis and mathematical equations [31]–[34]. The schematic diagram of HFSP is shown in Figure 1.

A. MODEL ASSUMPTIONS

The model assumptions for HFSP considering makespan, noise and dust pollution are described below.

(1) The n jobs are processed in s working stages hypothetically. There are m parallel machines in each stage. The speed
of processing job, unit pollution of noise and dust may be different in each stage.

(2) In each stage, each job can only be processed by one machine, and each job must complete all stages. Moreover, the machine of a certain stage cannot handle operation of other stages.

(3) All machines and jobs are available at 0 time, and jobs follow the principle of first come first process.

(4) The buffer area between two consecutive stages is infinite. The preparation time and the transmission time are included in the processing time.

(5) Each machine can only process one job at one time. The work that is carried out must be completed without interruption.

(6) If at least one machine is available in a stage, the stage shall be started immediately after the completion of the previous stage.

**B. DESCRIPTION OF VARIABLES AND SYMBOLS**

Variables and symbols of HFSP considering makespan, noise and dust pollution are described as follows.

\( n \): Number of jobs, single job is marked as \( j(j = 1, \ldots, n) \).

\( s \): Number of stages, single stage is marked as \( k(k = 1, \ldots, s) \).

\( m \): Number of machines in each stage, single machine is marked as \( i(i = 1, \ldots, m) \).

\( M_{i,k}^{j} \): Job \( j \) selects machine \( i \) to work in stage \( k \). The values of \( M_{i,k}^{j} \) form the matrix

\[
M = [M_{1,k}^{1}, M_{1,k}^{2}, \ldots, M_{m,s}^{n}] = [(M_{1,k}^{1})_{s \times m}, (M_{1,k}^{2})_{s \times m}, \ldots, (M_{m,s}^{n})_{s \times m}].
\]

\( Z_{i,k}^{j} \): Binary variable. If job \( j \) selects machine \( i \) to work in stage \( k \), the value is 1, otherwise 0. The values of \( Z_{i,k}^{j} \) form the matrix

\[
Z = [Z_{1,k}^{1}, Z_{1,k}^{2}, \ldots, Z_{m,s}^{n}] = [(Z_{1,k}^{1})_{s \times m}, (Z_{1,k}^{2})_{s \times m}, \ldots, (Z_{m,s}^{n})_{s \times m}].
\]

\( S_{i,k}^{j} \): Job \( j \) starts to work in machine \( i \) at stage \( k \). The values of \( S_{i,k}^{j} \) form the matrix

\[
S = [S_{1,k}^{1}, S_{1,k}^{2}, \ldots, S_{m,s}^{n}] = [(S_{1,k}^{1})_{s \times m}, (S_{1,k}^{2})_{s \times m}, \ldots, (S_{m,s}^{n})_{s \times m}].
\]

\( C_{i,k}^{j} \): Job \( j \) completes work in machine \( i \) at time \( C_{i,k}^{j} \) and stage \( k \). The values of \( C_{i,k}^{j} \) form the matrix

\[
C = [C_{1,k}^{1}, C_{1,k}^{2}, \ldots, C_{m,s}^{n}] = [(C_{1,k}^{1})_{s \times m}, (C_{1,k}^{2})_{s \times m}, \ldots, (C_{m,s}^{n})_{s \times m}].
\]

\( N_{i,k}^{j} \): Job \( j \) works on machine \( i \) in stage \( k \) and produces per unit time noise of \( N_{i,k}^{j} \). The values of \( N_{i,k}^{j} \) form the matrix

\[
N = [(N_{1,k}^{1})_{m \times s}, (N_{1,k}^{2})_{m \times s}, \ldots, (N_{m,s}^{n})_{m \times s}].
\]

\( D_{i,k}^{j} \): Job \( j \) works on machine \( i \) in stage \( k \) and produces per unit time dust of \( D_{i,k}^{j} \). The values of \( D_{i,k}^{j} \) form the matrix

\[
D = [(D_{1,k}^{1})_{m \times s}, (D_{1,k}^{2})_{m \times s}, \ldots, (D_{m,s}^{n})_{m \times s}].
\]

\( T_{i,k}^{j} \): Job \( j \) takes time \( T_{i,k}^{j} \) to work in machine \( i \) in stage \( k \). The values of \( T_{i,k}^{j} \) form the matrix

\[
T = [(T_{1,k}^{1})_{m \times s}, (T_{1,k}^{2})_{m \times s}, \ldots, (T_{m,s}^{n})_{m \times s}].
\]

**III. RIDER OPTIMIZATION ALGORITHM**

Optimization algorithm can be roughly divided into two categories: natural heuristic algorithm and artificial computation algorithm [35, 36]. Natural heuristic algorithm seeks solution to optimization problem based on inspiration from biological or physical phenomena [37]. Artificial computation algorithm is a guiding way to solve complex problem based on human thought. In 2019, D. Binu and B. S kariyapa proposed a new computing method that is called virtual computing [38, 39]. Virtual computing follows an idea based on imagination to solve optimization problem. At the same time, they put forward ROA based on the idea of virtual computing.

The ROA assumes that there are four groups of riders with the same number of riders moving towards the same goal to compete for the championship. The four groups of riders are the bypass riders, followers, overtakers and
attacker. The bypass riders take the lead by making a detour. The followers follow the leader most of the time. The overtakers overtake by judging the leaders and their positions. The attackers use the fastest speed to rush to the target point. In the ROA, each rider adjusts his own direction, gear, accelerator, brake and other parameters in real time to update his position. At the end of the competition (usually to reach the maximum number of iterations), the leader wins.

The schematic diagram of ROA is shown in Figure 2.

At time \( t \), the gear of the \( i \)-th rider is selected from \( \{0,1,2,3,4\} \), and the initial gear value of each rider is set to 0. The formula is equation (15).

\[
E = \{E_i\}; 1 \leq i \leq R
\] (15)

The value range of accelerator and decelerator of each rider belongs to \([0,1]\). Equation (16) represents accelerator, which is initialized to 0. Equation (17) represents decelerator, which is initialized to 1.

\[
e = \{e_i\}; 1 \leq i \leq R
\] (16)

\[
K = \{K_i\}; 1 \leq i \leq R
\] (17)

The maximum speed limit of each gear is shown as expression (18).

\[
V_i^E = \frac{V^i_{\text{max}}}{\| E \|}
\] (18)

\[
V_i^i = \frac{X_i^L - X_i^i}{T_{\text{OFF}}}
\] (19)

\( \| E \| \) represents the number of gears. \( V^i_{\text{max}} \) is the maximum speed that \( i \)-th rider can reach. \( X_i^L \) and \( X_i^i \) are the upper and lower bounds of the rider position values, respectively. \( T_{\text{OFF}} \) is the end time of the competition or the maximum number of iterations.

The success rate is defined as the reciprocal of the distance from the current position to the leader position. \( X_i \) is the current position of the \( i \)-th rider. \( L_T \) is the leader position. The rider individual with the best fitness value is the leader. The equation of success rate is equation (20).

\[
r_i = \frac{1}{|X_i - L_T|}
\] (20)

**C. LOCATION UPDATE EQUATIONS**

1) **BYPASS RIDER LOCATION UPDATE**

Bypass rider location is updated by equation (21).

\[
X_{i+1}^B(i,j) = \delta[X_i(\eta,j) \ast \beta + X_i(\xi,j) \ast (1 - \beta)]
\] (21)

where \( \delta \) and \( \beta \) belong to the random numbers of 0-1 uniform distribution. \( \eta \) and \( \xi \) belong to the random integers of 1-R uniform distribution.

2) **FOLLOWER LOCATION UPDATE**

Follower location is updated by expression (22).

\[
X_{i+1}^F(i,k) = X^L(L,K) + [\cos(T_{i,k}^L) \ast X^L(L,K) \ast d_i^l]
\] (22)

where \( X^L(L,K) \) represents the \( k \)-th dimension of the leader position. \( T_{i,k}^L \) represents the \( k \)-th dimension turning angle of the \( i \)-th rider at \( t \) time. \( d_i^l \) represents the distance that the rider will travel next. \( d_i^l \) is described by equation (23).

\[
d_i^l = \frac{v_i^l}{T_{\text{OFF}}}
\] (23)

\[
v_i^l = \frac{1}{3}[E_i^l \ast V_i^E + V_i^i \ast e_i^l + (1 - K_i^l) \ast V_{\text{max}}]
\] (24)
\( v_i^J \) refers to the speed of the \( i \)-th rider at time \( t \), which depends on the rider parameters. \( V_{\text{max}}^{ij} \) is the maximum speed that the \( i \)-th rider can reach. \( E_i^J, e_i^J, K_{ij}^J \) and \( V_i^E \) respectively represent the gear, accelerator, brake and gear speed limit of the rider.

The dimension \( k \) that is determined during location update depends on the real-time probability \( P_{ON}^i \). Equation (25) is used to calculate the real-time probability \( P_{ON}^i \). \( P_{ON}^i \) is multiplied by the corresponding dimension index \( j \). Then \( j \cdot P_{ON}^i \) is compared with \( Q \) to determine the coordinate that is considered for the change of position. And this comparison process is described in equation (26).

\[
P_{ON}^i = \left( \frac{1}{T_{OFF}} \right) * Q \quad (25)
\]

\[
k \ll (j \cdot P_{ON}^i); \text{if}(j \cdot P_{ON}^i) < Q \forall j \quad (26)
\]

3) OVERTAKER LOCATION UPDATE

Equation (27) represents the update equation of overtaker position, and the updated positions are selected according to equations (30) and (31). The difference between the leader ripple and the overtaker is calculated according to equation (30), and the dimensions less than the average value are selected.

Then, the direction of each rider is calculated according to the relative success rate of equation (29). Finally, equation (27) is employed to update the overtaker position according to the direction of travel, the position of the leading driver, the position of the current rider and the selected dimension.

\[
X_{i+1}^O(i, k) = X_i(i, k) + [D_i^J(i) + X^L(L, K)] \quad (27)
\]

\( D_i^J(i) \) is the direction indicator of the \( i \)-th rider at time \( t \). \( D_i^J(i) \) is calculated using the relative success rate \( S_i^R(i) \). The relative success rate is the ratio of the success rate of the corresponding rider to the maximum success rate of rider. The relative success rate is calculated by equation (29), where \( r_i(i) \) is the success rate of the \( i \)-th rider at time \( t \).

\[
D_i^J(i) = \left[ \frac{2}{1 - \log(S_i^R(i))} \right] - 1 \quad (28)
\]

\[
S_i^R(i) = \frac{r_i(i)}{\max_{r_i}^R} \quad (29)
\]

In equation (30), \( l(i, j) \) is the distance between the current rider position and the leader position in the \( j \)-th coordinate. The value of \( l(i, j) \) that is smaller than its average value is selected from \( j \) according to equation (31). In the equation (31), \( \mu_j \) is the average value of the standardized distance vector of the \( i \)-th rider.

\[
l(i, j) = \| X_i(i, j) - X_i^L(L, j) \| \quad (30)
\]

\[
k \ll \{ j; \text{if}(l(i, j) < \mu_j) \forall j \quad (31)
\]

4) ATTACKER LOCATION UPDATE

The attacker follows a similar update process as the follower. However, the attacker will update all the values in the coordinate instead of only the selected values. The equation (32) is the update equation of attacker position.

\[
X_{i+1}^A(i, j) = X_i^L(L, j) + [\cos(T_{i+1}^A(i, j)) * X_i^L(L, j)] + d_i^J \quad (32)
\]

D. RIDER PARAMETER UPDATE

In addition to the rider parameters shown in the above, here add a new parameter: the activity counter. The direction and gear are updated according to the activity counter. The activity counter is updated based on the success rate. The mathematical representation of the activity counter is given in equation (33).

\[
A_c^{i+1}(i) = \begin{cases} 1; & \text{if } r_{i+1}(i) > r_i(i) \\ 0; & \text{otherwise} \end{cases} \quad (33)
\]

The activity counter value of the \( i \)-th rider at time \( t+1 \) is \( A_c^{i+1}(i) \), which depends on \( r_i(i) \) and \( r_{i+1}(i) \). If \( r_{i+1}(i) > r_i(i) \) then the \( A_c^{i+1}(i) \) value is 1, otherwise 0.

Equation (34) updates the direction matrix with the value of the activity counter.

\[
T_{i+1}^{i+1} = \begin{cases} T_{i+1}^1; & \text{if } A_c^{i+1}(i) = 1 \\ T_{i+1}^0; & \text{otherwise} \end{cases} \quad (34)
\]

Equation (35) updates the direction matrix with the value of the activity counter. Equations (36) and (37) determine the values of the accelerator and and the brake based on the newly selected gear.

\[
E_{i+1}^{i+1} = \begin{cases} E_i^1 + 1; & \text{if } A_c^{i+1}(i) = 1 \text{ and } E_i^1 \neq |E| \\ E_i^1 - 1; & \text{if } A_c^{i+1}(i) = 0 \text{ and } E_i^1 \neq 0 \\ E_i^1; & \text{otherwise} \end{cases} \quad (35)
\]

\[
e_i^{i+1} = \frac{E_i^{i+1}}{|E_i|} \quad (36)
\]

\[
K_i^{i+1} = [1 - \frac{E_i^{i+1}}{|E_i|}] \quad (37)
\]

E. THE PSEUDO CODE OF RIDER OPTIMIZATION ALGORITHM

The pseudo code of ROA is described in algorithm 1.

IV. MULTI-OBJECTIVE RIDER OPTIMIZATION ALGORITHM

ROA was originally developed to solve the continuous single-objective optimization problem. It cannot be directly used to solve the multi-objective optimization problem. The Pareto archive and neighborhood sorting technique are applied to change ROA into MOROA.

A. DEFINITION OF MULTI-OBJECTIVE OPTIMIZATION PROBLEM

In single-objective optimization, because the objective function is unitary, the solutions are easy to compare. For the minimization problem, if \( x < y \), then the solution \( x \) is better than \( y \). However, the multi-objective optimization problem has multiple objective values. The solutions in the solution space cannot be compared by using common relational operators. At this time, the concept of Pareto optimality is given: when a solution shows better or equal objective values on all objectives, and provides a better objective value in at least
Algorithm 1 Rider Optimization Algorithm

1. Input: Random positions of the riders \(X_t\)
2. Output: Leader \(X^L\)
3. Begin
4. Initialize the population
5. Initialize the rider parameters: Steering angle \(T\), Gear \(E\), Accelerator \(e\), and Break \(K\)
6. Find the success rate \(r_i\)
7. while \(t < T_{OFF}\)
8. for \(i = 1\) to \(R\)
9. Update the position of the bypass rider using Eq. (21)
10. Update the position of the follower using Eq. (22)
11. Update the position of the overtaker using Eq. (27)
12. Update the position of the attacker using Eq. (32)
13. Rank the riders based on \(r_i\) as the leader
14. Select the rider having the maximum \(r_i\) as the leader
15. Update \(T\), \(E\), \(e\), and \(K\)
16. Return \(X^L\)
17. \(t = t + 1\)
18. end for
19. end while
20. Terminate

one objective function, the solution is superior (dominate) to the other solution [40]–[43]. The mathematical definition of Pareto optimality for minimization problem is as follows [44]–[46].

Let \(x = (x_1, x_2, \ldots, x_k) \in X\) be a solution vector. \(X\) is the solution space. Let \(F(x) = (f_1(x), f_2(x), \ldots, f_k(x))\) be a multi-objective function with \(k\) objective functions.

1) Pareto dominance. Suppose there are two vectors \(x_1, x_2 \in X\). If the vector \(x_1\) dominates vector \(x_2\), that is \((x_1 > x_2)\), then it can be expressed by equation (38).

\[
\forall i \in \{1, 2, \ldots, k\}, [f_i(x_1) \leq f_i(x_2)] \land \exists j \in \{1, 2, \ldots, k\}, [f_j(x_1) < f_j(x_2)]
\]

(38)

2) When the solution vector \(x^*\) is Pareto optimality, it can be expressed as equation (39).

\[
\exists y \in X|F(y) > F(x^*)(x^* \in X)
\]

(39)

3) A group of Pareto optimal solutions consist Pareto optimal set, the Pareto optimal set is expressed by equation (40).

\[
P_s := \{x^* \in X|\exists y \in X: y > x^*\}
\]

(40)

4) The set of objective function values corresponding to Pareto optimal solution set is called Pareto optimal front, which is expressed as equation (41).

\[
P_f := \{F(x)|x \in P_s\}
\]

(41)

When solving multi-objective problem, the goal of algorithm is to find a set of Pareto optimal solutions. This set of Pareto optimal solutions needs to be closest to the real Pareto front, large number and even distribution.

B. PARETO ARCHIVE AND NEIGHBORHOOD SORTING TECHNIQUE

1) PARETO ARCHIVE

The multi-objective heuristic algorithm will give a set of Pareto optimal solutions when the maximum number of iterations is reached. If the Pareto optimal solution set has a higher coverage rate and more solution number as far as possible, the Pareto optimal solution set can try to approach the real Pareto front. MOROA applies Pareto archive technique to maintain the Pareto optimal solution set in the optimization process.

Pareto archive is an independent multi-dimensional solution space with predefined size. The non-dominated solutions generated on each iteration will be added to the Pareto archive according to certain rules. The non-dominated solutions will be carefully maintained in Pareto archive. The non-dominated solutions will be added to Pareto archive or delete from Pareto archive followed a series of rules, and these rules are listed below.

(a) Addition of non-dominated solutions. After the new non-dominated solutions are produced in the current iteration, they should be added to the Pareto archive for maintenance.

(b) Deletion of non-dominated solutions. The size of the Pareto archive is limited, and the solutions in the Pareto archive should be dominated by each other. In the following cases, the deletion program of non-dominated solutions is executed.

(b) After the above procedures are executed, the number of solutions in the Pareto archive exceeds the upper limit of the Pareto archive. The excess non-dominated solutions need to be deleted. It is necessary to select the deleted solutions reasonably to ensure that the obtained Pareto front end is uniform and highly approximate to the real Pareto front end. Therefore MOROA uses neighborhood sorting technique to select the deleted solutions and ensure to get optimal Pareto solution set.

2) NEIGHBORHOOD SORTING TECHNIQUE

In order to make the Pareto front as evenly distributed as possible, Neighborhood sorting technique is used to delete the non-dominated solutions that exceed the upper limit of Pareto archive. And neighborhood sorting technique is applied to select the leaders and guide the population position.
update. Neighborhood sorting technique calculates the density around a non-dominated solution. The density of the solution is employed as the probability to determine whether the solution is deleted or selected as the leader.

Neighborhood sorting technique defines a hypersphere for each solution in the archive. The radius of the hypersphere is determined by the prior knowledge of the problem. The prior knowledge is related to the background of the practical problem. Each hypersphere will contain other solutions (0, 1 or multiple solutions) around the solution. The probability of each hypersphere is equation (42).

\[ P_i = \frac{N_i}{c} \] (42)

The \( c \) is a constant greater than 1, \( N_i \) is the number of non-dominated solutions in the \( i \)-th hypersphere. The roulette mechanism is used to choose the solutions to be deleted, and the interval of roulette will be determine by probability \( P_i \). The equation (42) means that the algorithm has a higher probability to select deleted solutions from areas with higher density. The purpose of this choice is to make the non-dominated solutions distribute evenly in the Pareto front end. Therefore, the solutions exceeding the Pareto archive capacity will be deleted according to roulette mechanism and equation (42).

The neighborhood sorting technique will be used to choose the leader to guide the Pareto front end more evenly. Similarly, the hypersphere for each solution will be defined and the probability of each hypersphere is expressed by equation (43).

\[ P_i' = \frac{c}{N_i} \] (43)

where \( c \) is a constant greater than 1, and \( N_i \) is the number of solutions in the \( i \)-th hypersphere. In the roulette mechanism, the hypersphere with the lowest density assigns the maximum probability to be selected as the leader.

**Algorithm 2 Multi-Objective Rider Optimization Algorithm**

1. Input: Random positions of the riders \( X_t \)
2. Output: Archive
3. Begin
4. Initialize the population
5. Initialize the rider parameters: Steering angle T, Gear E, Accelerator e, and Break K
6. Initialize the Archive, find the leader
7. Find the success rate \( r_t \)
8. while \( t < T_{OFF} \)
9. for \( i = 1 \) to \( R \)
10. Update the position of the bypass rider using Eq. (21)
11. Update the position of the follower using Eq. (22)
12. Update the position of the overtaker using Eq. (27)
13. Update the position of the attacker using Eq. (32)
14. Update the Archive using Eq. (42) and roulette mechanism
15. Find the leader using Eq. (43) and roulette mechanism
16. Calculate the \( r_t \) of riders
17. Update T, E, e, and K
18. Return Archive
19. \( t = t + 1 \)
20. end for
21. end while
22. Terminate

**V. DMOROA FOR HFSP CONSIDERING MAKESPAN, NOISE AND DUST POLLUTION**

In literature [30], ROA has been employed to diagnose analog circuit faults, and proven to be effective in handling complex engineering problem. ROA has the advantages of fast search speed, high search accuracy and good at dealing with complex problem. Therefore, MOROA will be discretized into DMOROA. DMOROA is used to solve the HFSP considering makespan, noise and dust pollution. The data of industrial production is difficult to obtain and lacks of universality, so the experimental data is randomly generated. The randomly generated data is usually used to test HFSP [15]–[17]. Therefore, the value of matrix \( S, C, N, D, T \) is determined by random initialization in this paper. The initialization of matrix \( S \) and \( C \) should satisfy the conditions of equations (6) and (7).

Each HFSP is identified as “number of jobs - number of stages - number of parallel machines in each stage”. Suppose that \( n \) jobs and \( s \) stages are needed, and \( m \) parallel machines of these two hyperspheres have the maximum probability of being deleted or selected as the leader respectively.

**C. PSEUDO CODE OF MOROA**

The pseudo code of MOROA is explained in algorithm 2.

Figure 3 shows an example of the above two selection processes. The red circle represents the hypersphere with the highest density, and the black circle represents the hypersphere with the lowest density. The corresponding solutions
can be selected in each stage, then the HFSP can be named as \( n-s-m \) problem. A specific \( 3-3-2 \) problem will be applied as an example to illustrate the process of population initialization and discretization in DMOROA. The time sequence Gantt chart of problem \( 3-3-2 \) is Figure 4.

**B. POPULATION DISCRETIZATION**

The discretization method of reference [13] is used to discretize the population. The components of vector \( Q \) are numbered in order from small to large. After numbering, the vector \( Q \) is discretized into vector \( Q' \). For \( 3-3-2 \) problem, the first row of Table 2 is converted to the second row of Table 2. The processing sequence of the jobs 1, 2, 3 is 1, 2, 3.

The smallest position value rule [27] is used to determine machine selection. The \( j \)-th job select one of the \( m \) machines to process in stage \( k \). The \( m \) values of machine selection are \((M_{j1}^k, M_{j2}^k, \ldots, M_{jm}^k)\). The minimum value of these \( m \) values is marked with 1, indicating that the corresponding machine is selected. The other values are marked with 0, indicating that the corresponding machines are not selected. For example, values \((0.26744, 0.33730)\) in Table 3 represent that job \( n_1 \) selects machines \( m_1 \) or \( m_2 \) in stage \( s_1 \). The minimum value is marked as 1 and the other values are marked as 0, so \((0.26744, 0.33730)\) becomes \((1, 0)\).

According to the smallest position value rule, matrix \( M \) is discretized into matrix \( Z \). For \( 3-3-2 \) problem, the minimum position rule discretizes Table 3 into Table 4.

**C. SOLVING HFSP CONSIDERING MAKESPAN, NOISE AND DUST POLLUTION WITH DMOROA**

When MOROA is used to solve the HFSP, the initial population will be discretized to transformed MOROA into DMOROA. The DMOROA will be employed to solve HFSP considering the makespan, noise and dust pollution. The three objective functions are determined by equations (1), (2) and (3) respectively. The variables of equation (1) are determined by vector \( Q' \) and matrix \( Z \). The parameters of equation (1) are determined by matrix \( S \), \( C \) and \( T \). The variables of equation (2) are determined by matrix \( N \) and \( T \). The parameters of equation (2) are determined by matrix \( Z \). The variables of equation (3) are determined by matrix \( Z \).
TABLE 4. The discrete machine selection matrix \( Z \) of problem "3-3-2".

|     | n1 | n2 | n3 |
|-----|----|----|----|
| m1 | 1  | 1  | 0  |
| m2 | 0  | 0  | 1  |

TABLE 5. The initialization of time matrix \( T \) for '3-3-2' problem.

|     | n1 | n2 | n3 |
|-----|----|----|----|
| m1 | 1  | 1  | 1  |
| m2 | 0  | 0  | 0  |

TABLE 6. The initialization of noise matrix \( N \) for '3-3-2' problem.

|     | n1 | n2 | n3 |
|-----|----|----|----|
| m1 | 1  | 1  | 1  |
| m2 | 0  | 0  | 0  |

TABLE 7. The initialization of dust matrix \( D \) for '3-3-2' problem.

|     | n1 | n2 | n3 |
|-----|----|----|----|
| m1 | 1  | 1  | 1  |
| m2 | 0  | 0  | 0  |

TABLE 8. The result for '3-3-2' problem.

|               | Makespan | Noise production | Dust production |
|---------------|----------|-----------------|-----------------|
| result        | 140      | 756             | 812             |

The parameters of equation (3) are determined by matrix \( D \) and \( T \). For problem “3-3-2”, the matrix \( T, N \) and \( D \) are Table 5, 6 and 7. DMOROA solves the “3-3-2” problem and obtain the optimal solutions that are listed in Table 8.

The principle of DMOROA for HFSP considering makespan, noise and dust pollution is described in algorithm 3.

VI. SIMULATION EXPERIMENT

The experiment results of HFSP considering makespan, noise and dust pollution are compared between DMOROA, non-dominated sorting genetic algorithm 2 (NSGA2) and multi-objective particle swarm optimization algorithm (MOPSO) to evaluate the performance of DMOROA.

A. EXPERIMENTAL PLATFORM

The operating environment of the simulation experiment will be illustrated. The machine model is Dawning 5000A. The server is configured with Xeon x5620 CPU (4 cores) 2, 24GB memory and 1 * 300 GB SAS hard disk. The operating system is Linux rhel5.6. The programming tool is Matlab 2012a (for Linux).

B. EXPERIMENTAL METHOD

In order to evaluate the performance of DMOROA, the different configuration data from the size and structure of the HFSP considering makespan, noise and dust pollution is needed. In addition, relevant performance evaluation metrics are needed and the level and scope of some evaluation metrics are shown in Table 9.

In the experiment, each algorithm uses the same coding strategy that is explained in Part V, and the parameter settings of each algorithm are shown in Table 10. Each algorithm will test 10 times for each problem and run 500 iterations each time. The results are compared with the evaluation metrics proposed in the next section.

C. EVALUATION CRITERIONS

For multi-objective HFSP, it is difficult to choose a single performance metric to compare the effectiveness and efficiency of different algorithms. Therefore, four performance criterions are proposed for convergence, diversity, distribution uniformity and stability. The four performance criterions are coverage metric, number of non-dominated solutions, spread and mean relative deviation index.
TABLE 9. Summary of test data.

| Factors                                      | Levels | Number of levels |
|----------------------------------------------|--------|------------------|
| Number of jobs                               | 10, 30, and 50 | 3                |
| Number of stages                             | 3, 5 | 2                |
| Number of machines at each stage             | 3, 6, and 9 | 3                |
| Processing time of each operation            | Discrete uniform [1,50] | 1                |
| Dust per unit of time produced by each machine | Discrete uniform [1,5] | 1                |
| Noise per unit of time produced by each machine | Discrete uniform [1,5] | 1                |

Algorithm 3 DMOROA for HFSP Considering Makespan, Noise and Dust Pollution

1. Input: parameter matrixes S, C, N, D and T
2. Output: Archive
3. Begin
4. Initialize the population X
5. X is divided into job processing sequence vector Q and machine selection matrix M
6. vector Q is discretized to vector Q'
7. Matrix M is discretized to matrix Z according to the smallest position value rule
8. Initialize the rider parameters: Steering angle T, Gear E, Accelerator e, and Break K
9. Initialize the Archive, find the leader
10. Find the success rate $r_i$
11. while $t < T_{OFF}$
12. for $i = 1$ to $R$
13. Update the position of the bypass rider using Eq. (21)
14. Update the position of the follower using Eq. (22)
15. Update the position of the overtaker using Eq. (27)
16. Update the position of the attacker using Eq. (32)
17. Solve the fitness values of Eqs. (1), (2) and (3) to obtain a solution set of three-dimension (makespan, noise, dust pollution).
18. Update the Archive using Eq. (42) and roulette mechanism
19. Find the leader using Eq. (43) and roulette mechanism
20. Calculate the $r_i$ of riders
21. Update T, E, e, and K
22. Return Archive
23. $t = t + 1$
24. end for
25. end while
26. Terminate

1) COVERAGE METRIC
The $C(A, B)$ represents the coverage metric of set $A$ to $B$. $A$ and $B$ are two sets of non-dominated solution. The value of $C(A, B)$ represents the percentage of solutions in $B$ dominated by at least one solution in $A$. It can be calculated by the equation (44).

$$C(A, B) = \frac{|\{b \in B : \exists a \in A : a \succ b \text{ or } a = b\}|}{|B|}$$

(44)

If the value of $C(A, B)$ is closer to 1, it means that almost all the solutions in $B$ are dominated by a certain solution in $A$. It is worth noting that $C(A, B) \neq 1 - C(B, A)$. This means that it is necessary to compare the values of $C(A, B)$ and $C(B, A)$ to determine which non-dominated solution set has higher precision.

2) NUMBER OF NON-DOMINATED SOLUTIONS
Number of non-dominated solutions represents the number of non-dominated solutions obtained by the algorithm that is applied to test a HFSP. In multi-objective HFSP, producers often need to choose specific scheduling strategies according to the results of scheduling evaluation. Getting more Pareto optimal solutions can make the Pareto front more uniform and smooth, which is beneficial for producer to choose the most suitable scheduling strategy for the current situation.

3) SPREAD
The spread metric measures the diversity and the distribution uniformity of solution set. The spread is a more comprehensive metric to show the quality of solution set. The calculation equation of spread is equation (45).

$$Spread = \frac{\sum_{j=1}^{obj\_num} d^e_j + \sum_{j=1}^{R} |d_i - \bar{d}|}{\sum_{j=1}^{obj\_num} d^e_j + R \cdot \bar{d}}$$

(45)

where $d_i$ is the Euclidean distance between the $i$-th solution in one solution set and its closest solution in its neighbor, $\bar{d}$ is the average value of all $d_i$. $d^e_j$ is the Euclidean distance between the extreme solutions in the $j$-th objective and the boundary solutions of the obtained solution set. The $obj\_num$ is the number of objective.

4) MEAN RELATIVE DEVIATION INDEX
Mean relative deviation index is applied to measure the stability of the algorithm. The mean relative deviation index of each objective is calculated by equation (46).

$$MRDI_j = \sum_{i=1}^{R} \left| \frac{obj_{ij} - obj^*_j}{obj^*_j} \right| \times \frac{100}{R}$$

(46)

($j = 1, \ldots, \text{objective number}$).

In the equation (45), $MRDI_j$ represents the mean relative deviation index of the $j$-th target. $obj^*_j$ represents the optimal
TABLE 10. Parameters setting of DMOROA, NSGA2, MOPSO.

| Algorithms   | Parameters setting                                      |
|--------------|---------------------------------------------------------|
| DMOROA       | gear numbers: 5  crossover probability: 1.0  mutation probability: 0.1 |
| NSGA2        | inertia coefficient: 0.4  Individual cognitive coefficient: 2 |
| MOPSO        | social cognition coefficient: 2  grid numbers: 20 |

FIGURE 5. Non-dominates solutions for 10-job problems.

The results of experiments will be analyzed to demonstrate the performance of DMOROA. Figure 5 to Figure 7 show the Pareto front end obtained by each algorithm on each problem. Figure 8 to Figure 10 are the comparison of the target values of each problem on the two-dimensional coordinate figure. Figure 5 and Figure 8 show the 10 workpiece problems, Figure 6 and Figure 9 show the 30 workpiece problems, Figure 7 and Figure 10 show the 50 workpiece problems.

In Figure 5, the solutions of DMOROA obviously dominate the solutions of the comparison algorithms in “10-3-3”, “10-3-6”, “10-5-6”. In the “10-5-3” problem, the solutions of DMOROA are slightly better than NSGA2, but the coverage of Pareto front end is wider and the diversity of solutions is better. In the 10-5-9 problem, a part of DMOROA solutions are excellent, which dominate the solutions of all comparison algorithms, but the other solutions are mixed with the solutions of MOPSO and dominated by the solutions of NSGA2.

In Figure 6, the solutions of DMOROA are obviously better than the comparison algorithms in “30-3-3”, “30-3-9” and “30-5-6” problems. In “30-3-6” and “30-5-9”, most of the solutions of DMOROA and NSGA2 dominate each other, but DMOROA still has a small part of solutions that dominate all the solutions of the comparison algorithms. In the “30-5-3” problem, the effect of NSGA2 is the best. The solutions of DMOROA and MOPSO are mixed together and dominated by the solutions of NSGA2.

In Figure 7, the effect of DMOROA is better than the two comparison algorithms in “50-3-6”, “50-3-9”, “50-5-3”, “50-5-6” and “50-5-9”. In these problems, the solutions of each algorithm are mixed in the center region of the Figure 7, but a part of DMOROA solutions with excellent performance break through the center region and dominate other solutions. DMOROA performs best in “50-5-3” and “50-5-9”. On the contrary, in the “50-3-3” problem, DMOROA has only a few solutions that dominate all other solutions.

In conclusion, it is obvious that with the increase of the number of jobs, DMOROA performs better in dealing with problems of more stages and more parallel machines.
Table 11 shows the test results of coverage metric. Each HFSP is evaluated by three statistical indexes: maximum, average and minimum. The data of three statistical indexes comes from ten independent tests of each HFSP. There are 6 columns of data in Table 11. Pairwise contrast to evaluate which algorithm is better. In the comparison between DMOROA and NSGA2, DMOROA is superior to NSGA2 in “10-3-6”, “10-3-9”, “10-5-6”, “10-5-9”, “30-3-3”, “30-3-9”, “30-5-6”, “50-3-9”, “50-5-3” and “50-5-9”. DMOROA completely dominates NSGA2 except for “10-3-6”. In the “50-3-6” and “50-3-9” problems, DMOROA is only slightly better than NSGA2 with a
FIGURE 8. Two dimensional comparison of objective function values for 10-job problems.
FIGURE 9. Two dimensional comparison of objective function values for 30-job problems.
FIGURE 10. Two dimensional comparison of objective function values for 50-job problems.
TABLE 11. Coverage metric value of non-domination sets.

| Problem | Statistic | C(DMOROA, NSGA2) | C(NSGA2, DMOROA) | C(DMOROA, MOPSO) | C(MOPSO, DMOROA) | C(NSGA2, MOPSO) | C(MOPSO, NSGA2) |
|---------|-----------|------------------|------------------|------------------|------------------|----------------|-----------------|
| 10-3-3  | Max       | 0                | 0                | 0.9              | 0                | 1              | 0               |
|         | Mean      | 0                | 0                | 0.825            | 0                | 0.65           | 0               |
|         | Min       | 0                | 0                | 0.75             | 0                | 0.3            | 0               |
| 10-3-6  | Max       | 0.75             | 0                | 1                | 0                | 0.5833333333   | 0               |
|         | Mean      | 0.445            | 0                | 0.79166667       | 0                | 0.4166666667   | 0               |
|         | Min       | 0.14             | 0                | 0.5833333333     | 0                | 0.25           | 0               |
| 10-3-9  | Max       | 1                | 0                | 1                | 0                | 0.0714285714    | 0.625           |
|         | Mean      | 0.5416666667     | 0                | 0.774345         | 0                | 0.035714285714  | 0.625           |
|         | Min       | 0.0833333333     | 0                | 0.54869          | 0                | 0              | 0.625           |
| 10-5-3  | Max       | 0                | 0.0833333333     | 1                | 0                | 0.5714285714    | 0               |
|         | Mean      | 0.0416666667     | 0                | 0.73156          | 0                | 0.3965714285714 | 0               |
|         | Min       | 0                | 0.46312          | 0                | 0                | 0.221714285714  | 0               |
| 10-5-6  | Max       | 1                | 0                | 1                | 0                | 0.55           | 0               |
|         | Mean      | 0.72             | 0                | 0.7639           | 0                | 0.349265       | 0               |
|         | Min       | 0.44             | 0                | 0.5278           | 0                | 0.14853        | 0               |
| 10-5-9  | Max       | 1                | 0                | 1                | 0                | 0.25           | 0.285714285714  |
|         | Mean      | 1                | 0                | 0.777777         | 0                | 0.218171       | 0.205285143     |
|         | Min       | 1                | 0                | 0.55555          | 0                | 0.18634        | 0.124856       |
| 30-3-3  | Max       | 0.592973         | 0                | 0.92306          | 0                | 0              | 0.7             |
|         | Mean      | 0.05946          | 0                | 0.84612          | 0                | 0              | 0.4             |
| 30-3-6  | Max       | 0.3333333333     | 0.3333333333     | 0.5714285714     | 0                | 0.35714285714  | 0               |
|         | Mean      | 0.1666666667     | 0.1666666667     | 0.3070642857     | 0                | 0.30289142857  | 0               |
|         | Min       | 0                | 0                | 0.04258          | 0                | 0              | 0.24864        |
| 30-3-9  | Max       | 1                | 0                | 1                | 0                | 0.5666666667   | 0.142857143     |
|         | Mean      | 1                | 0                | 0.78164          | 0                | 0.6666666667   | 0.135714285714  |
|         | Min       | 1                | 0                | 0.56328          | 0                | 0.6666666667   | 0.12857143     |
| 30-5-3  | Max       | 0                | 0.739130435      | 0.9166666667     | 0                | 0              | 0               |
|         | Mean      | 0.499610217      | 0.7916666667     | 0.96293          | 0                | 0              | 0               |
|         | Min       | 0                | 0.2609          | 0.66666667       | 0                | 0.92586        | 0               |
| 30-5-6  | Max       | 1                | 0                | 1                | 0                | 0.4            | 0               |
|         | Mean      | 0.964            | 0                | 1                | 0                | 0.32259        | 0               |
|         | Min       | 0.928            | 0                | 1                | 0                | 0.24518        | 0               |
| 30-5-9  | Max       | 0.1666666667     | 0.2666666667     | 1                | 0                | 0.875          | 0               |
|         | Mean      | 0.1166666667     | 0.1333333333     | 0.76393          | 0                | 0.650055       | 0               |
|         | Min       | 0.0666666667     | 0                | 0.52786          | 0                | 0.42511        | 0               |
| 50-3-3  | Max       | 0                | 0.5625          | 0                | 0.25             | 0.9375         | 0               |
|         | Mean      | 0                | 0.412           | 0                | 0.225            | 0.46875        | 0               |
|         | Min       | 0                | 0.2615          | 0                | 0.2              | 0              | 0               |
| 50-3-6  | Max       | 0.0555555556     | 0.1666666667     | 0.52             | 0.1666666667     | 0.36           | 0               |
|         | Mean      | 0.0277777778     | 0.0955883333     | 0.26             | 0.0833333333     | 0.274995       | 0               |
|         | Min       | 0                | 0.02451         | 0                | 0                | 0.18999        | 0               |
| 50-3-9  | Max       | 0.05             | 0                | 1                | 0                | 0.8666666667   | 0               |
|         | Mean      | 0.025            | 0                | 0.7              | 0                | 0.8666666667   | 0               |
|         | Min       | 0                | 0.4             | 0                | 0                | 0.8666666667   | 0               |
| 50-5-3  | Max       | 1                | 0                | 0.9166666667     | 0                | 0.0833333333   | 0.318181818     |
|         | Mean      | 1                | 0                | 0.8147983333     | 0                | 0.0416666667   | 0.251690909    |
|         | Min       | 1                | 0                | 0.71293          | 0                | 0              | 0.1852         |
| 50-5-6  | Max       | 0.1666666667     | 0.1666666667     | 0.875            | 0.1666666667     | 0.0625         | 0.0416666667   |
|         | Mean      | 0.1433333333     | 0.0833333333     | 0.6375           | 0.0833333333     | 0.03125        | 0.0208333333   |
|         | Min       | 0.12             | 0                | 0.4              | 0                | 0              | 0               |
| 50-5-9  | Max       | 1                | 0                | 1                | 0                | 0.85714285714  | 0               |
|         | Mean      | 0.816665         | 0                | 1                | 0                | 0.814841429    | 0               |
|         | Min       | 0.63333          | 0                | 1                | 0                | 0.77254        | 0               |
Wilcoxon rank sum test is carried out for coverage metrics to verify the significance of DMOROA and the test results are shown in Table 12. DMOROA is significantly better than NSGA2 in most problems except for “10-3-3”, “10-5-3”, “30-5-3” and “30-5-9”. The Wilcoxon rank sum test of DMOROA is more significant than MOPSO, so the coverage metric of DMOROA is better than MOPSO. NSGA2 is significantly better than MOPSO in most problems, so NSGA2 has better performance than MOPSO.

In conclusion, the search accuracy of DMOROA is better than NSGA2 and MOPSO in almost every aspect. NSGA2 dominates MOPSO in 12 problems, so NSGA2 is better than MOPSO.

Table 13 shows the number of non-dominated solutions that each algorithm obtains in each HFSP. Table 13 also includes three statistical indexes: maximum, average, and minimum. The statistical data comes from ten independent tests of each HFSP. The number of non-dominated solutions obtained by DMOROA in 10 problems is better than the comparison algorithms, and 10 problems are more than half of the total number of problems, so solution set of DMOROA has more diversity. The ten problems are: “10-3-3”, “10-3-6”, “10-5-6”, “30-5-3”, “30-5-6”, “50-3-9”, “50-5-3”, “50-5-6”, “50-5-9” and “30-3-9”. In the “10-3-9” problem, statistical indexes of DMOROA are slightly worse than NSGA2, but better than MOPSO. Similarly, in the “50-3-3” problem, the statistical indexes of DMOROA are slightly worse than MOPSO, but better than NSGA2. In “10-5-3”, “10-5-9”, “30-3-3”, “30-3-6”, “30-5-9”, “50-3-6”, the number of non-dominated solutions of DMOROA is not ideal, but the minimum value is better than MOPSO. Generally, the solutions of DMOROA in optimization of multi-objective HFSP have better diversity than the classical multi-objective heuristic algorithm. DMOROA performs well in the problem with 3,6 parallel machines, but the number of non-dominated solutions in 9 parallel machine problem is significantly reduced.

Table 14 lists the values of spread. The smaller the spread value is, the more uniform and diverse the solution set is. DMOROA has the best spread value to the comparison algorithms in 10 problems, such as “10-3-3”, “10-3-9”, “10-5-3”, “10-5-9”, “10-5-6”, “10-5-9”, “30-3-6”, “30-3-9”, “30-3-9”.

### Table 12. Wilcoxon rank sum for the coverage metric value of the test problems.

| Problem | C(DMOROA,NSGA2)/C(NSGA2, DMOROA) | C(DMOROA,MOPSO)/C(MOPSO,DMOROA) | C(NSGA2,MOPSO)/C(MOPSO,NSGA2) |
|---------|-----------------------------------|-----------------------------------|----------------------------------|
| H       | P                                | Sig                              | H                                | P                                | Sig                              | H                                | P                                | Sig                              |
| 10-3-3  | 0.0522637                        | ≈ 1                              | 0.04534                          | 1                                | 4.02E-07                         | 7                                |
| 10-3-6  | 1.535E-06                        | + 1                             | 0.019634                         | + 1                              | 0.0351                           | + 1                              |
| 10-3-9  | 8.83E-08                         | + 1                             | 0.005254                         | + 0                             | 0.321245                         | ≈ 1                              |
| 10-5-3  | 0.133539                         | ≈ 1                             | 0.035691                         | + 1                              | 0.007772                         | + 1                              |
| 10-5-6  | 0.035621                         | + 1                             | 0.042589                         | + 1                              | 6.58E-15                         | + 1                              |
| 10-5-9  | 0.028769                         | ≈ 1                             | 0.036522                         | + 0                             | 0.236412                         | ≈ 1                              |
| 30-3-3  | 0.056523                         | ≈ 1                             | 0.001124                         | + 1                              | 8.88E-06                         | + 1                              |
| 30-3-6  | 6.61E-07                         | + 1                             | 6.32E-05                         | + 1                              | 3.16E-05                         | + 1                              |
| 30-5-9  | 0.00605                          | + 1                             | 9.36E-06                         | 0                                | 0.052814                         | + 1                              |
| 30-5-6  | 0.0521                           | ≈ 1                             | 6.4E-08                          | + 1                              | 2.1E-05                          | + 1                              |
| 30-5-9  | 0.0096                           | + 1                             | 0.015298                         | + 1                              | 8.21E-06                         | + 1                              |
| 30-5-9  | 0.4619                           | ≈ 1                             | 0.024488                         | + 1                              | 0.01523                          | + 1                              |

### Table 13. Number of non-dominated solutions found by each algorithm.

| Problem | DMOROA | NSGA2 | MOPSO |
|---------|--------|-------|-------|
|         | Max    | Mean  | Min   | Max    | Mean  | Min   | Max    | Mean  | Min   |
| 10-3-3  | 17     | 13.5  | 10    | 13     | 12    | 11    | 6      | 4.5   | 3     |
| 10-3-6  | 10     | 7     | 4     | 8      | 7     | 6     | 5      | 3.5   | 2     |
| 10-3-9  | 10     | 8     | 6     | 12     | 10.5  | 9     | 8      | 7.5   | 7     |
| 10-5-3  | 5      | 4     | 3     | 15     | 13    | 11    | 12     | 9.5   | 7     |
| 10-5-6  | 7      | 6     | 5     | 5      | 4     | 3     | 9      | 6     | 3     |
| 10-5-9  | 8      | 5     | 2     | 14     | 10    | 6     | 10     | 8.5   | 7     |
| 30-3-3  | 16     | 13    | 10    | 24     | 22    | 20    | 21     | 13    | 5     |
| 30-3-6  | 9      | 8     | 7     | 15     | 13.5  | 12    | 11     | 9.5   | 8     |
| 30-3-9  | 19     | 15    | 11    | 8      | 6.5   | 5     | 12     | 8.5   | 5     |
| 30-5-3  | 17     | 13    | 9     | 17     | 14    | 11    | 15     | 4     | 3     |
| 30-5-6  | 13     | 9     | 5     | 11     | 9     | 7     | 12     | 13    | 11    |
| 30-5-9  | 11     | 10.5  | 10    | 25     | 20.5  | 16    | 15     | 10.5  | 6     |
| 50-3-3  | 17     | 15    | 13    | 6      | 5.5   | 5     | 19     | 17    | 14    |
| 50-3-6  | 12     | 8.5   | 5     | 27     | 22.5  | 18    | 13     | 11.5  | 10    |
| 50-3-9  | 23     | 18    | 13    | 8      | 7.5   | 7     | 19     | 16    | 13    |
| 50-5-3  | 18     | 13.5  | 9     | 16     | 15    | 14    | 12     | 10    | 8     |
| 50-5-6  | 17     | 12.5  | 8     | 6      | 4     | 4     | 15     | 12    | 9     |
| 50-5-9  | 20     | 17.5  | 15    | 17     | 13    | 9     | 10     | 9.5   | 9     |
TABLE 14. Spread comparison of different algorithms for the test problems.

|          | 10-3-3 | 10-3-6 | 10-3-9 | 10-5-3 | 10-5-6 | 10-5-9 |
|----------|--------|--------|--------|--------|--------|--------|
| DMOROA   | 1.18783| 1.12775| 1.10042| 1.175018| 0.886092| 1.091696|
| NSGA2    | 1.299686| 1.123828| 1.464741| 1.178525| 1.115982| 1.17906|
| MOPSO    | 1.901721| 1.478311| 1.373494| 1.436785| 1.240157| 1.284914|
|          | 30-3-3 | 30-3-6 | 30-3-9 | 30-5-3 | 30-5-6 | 30-5-9 |
| DMOROA   | 1.050609| 0.999888| 0.974051| 1.069807| 1.001474| 1.050349|
| NSGA2    | 1.000228| 1.034463| 1.080022| 0.929587| 1.022572| 1.020506|
| MOPSO    | 1.060562| 1.161585| 1.128098| 1.049627| 1.007832| 1.057827|
|          | 50-3-3 | 50-3-6 | 50-3-9 | 50-5-3 | 50-5-6 | 50-5-9 |
| DMOROA   | 0.98249| 0.938681| 0.984045| 1.006331| 0.991832| 1.015697|
| NSGA2    | 0.996429| 0.972555| 0.965033| 0.952736| 0.973649| 1.019022|
| MOPSO    | 0.872572| 1.004191| 0.998847| 0.969474| 0.979632| 1.040155|

TABLE 15. Mean relative deviation index comparison of different algorithms for the test problems.

| Problem | DMOROA | NSGA2 | MOPSO |
|---------|--------|-------|-------|
|         | Makespan | Noise | Makespan | Noise | Makespan | Noise |
| 10-3-3  | 7.45628 | 2.42589 | 10.05247 | 8.18182 | 14.98865 | 20.19355 |
| 10-3-6  | 17.38905 | 18.36707 | 31.78171 | 28.57143 | 16.15902 | 19.81982 |
| 10-3-9  | 12.92735 | 8.05556 | 34.54740 | 13.41991 | 16.82616 | 14.56137 |
| 10-5-3  | 6.35524 | 6.22348 | 3.42869 | 7.80744 | 7.34448 | 16.94639 |
| 10-5-6  | 6.25000 | 9.90492 | 11.30314 | 8.18345 | 10.92687 | 13.90086 |
| 10-5-9  | 20.21721 | 5.92486 | 9.25229 | 10.71429 | 14.20135 | 28.86543 |
| 30-3-3  | 7.73457 | 5.33982 | 8.15900 | 21.17063 | 9.13815 | 11.10847 |
| 30-3-6  | 5.43478 | 6.57964 | 10.67180 | 14.61426 | 10.64193 | 13.23588 |
| 30-3-9  | 11.51386 | 3.96102 | 8.61514 | 18.28402 | 13.84196 | 11.98037 |
| 30-5-3  | 2.44683 | 1.29192 | 5.48306 | 16.75655 | 17.01518 | 13.43987 |
| 30-5-6  | 17.17734 | 4.80249 | 5.75363 | 16.95502 | 6.37529 | 7.99950 |
| 30-5-9  | 11.06512 | 6.19935 | 9.76131 | 15.50000 | 12.95220 | 6.35154 |
| 50-3-3  | 12.64066 | 7.97941 | 10.19423 | 24.14402 | 20.10954 | 12.17448 |
| 50-3-6  | 12.90123 | 5.98185 | 5.99308 | 23.88546 | 9.27790 | 8.64479 |
| 50-3-9  | 12.04545 | 5.33256 | 6.90413 | 13.54000 | 5.76706 | 8.42444 |
| 50-5-3  | 14.80958 | 4.22165 | 6.57529 | 5.19863 | 5.30127 | 6.65645 |
| 50-5-6  | 18.53903 | 2.19823 | 4.65497 | 16.47462 | 13.99746 | 11.81543 |
| 50-5-9  | 1.00000 | 3.83770 | 6.21705 | 9.25092 | 4.23267 | 4.56954 |

“30-5-6”, “50-3-6”, “50-5-9”. So DMOROA obtains the solution set with optimal homogeneity and diversity in more than half of the problems. In the two problems of “10-3-9” and “10-5-6”, the spread value of DMOROA is superior. DMOROA can get a good solution set of uniform spread when the complexity of the HFSP is low. As the HFSP increases complexity, the solution set of DMOROA decreases the uniformity slightly.

Table 15 shows the mean relative deviation index of each HFSP to each algorithm. Each algorithm is evaluated by the three statistical indexes: makespan, noise and dust. DMOROA is better than the comparison algorithms in 7 problems, such as “10-3-9”, “10-5-3”, “10-5-6”, “30-3-9”, “30-5-3”, “50-3-6”, “50-5-9”. For other problems, DMOROA is excellent in overall performance, but slightly inferior to the comparison algorithms in specific indexes of specific problems. In the makespan index, DMOROA is slightly inferior to NSGA2 in “10-5-9” and “50-5-3”, and slightly worse than both of the two comparison algorithms in “30-5-6” and “50-5-6”. In dust index, DMOROA is worse than NSGA2 in “10-5-9” and “50-5-9”, and slightly worse than MOPSO in “30-3-9”, “30-3-6” and “50-3-3”. In particular, stability of DMOROA in “10-3-6” and “30-5-9” is not satisfactory. In conclusion, the stability of DMOROA is slightly better than the classical multi-objective heuristic algorithm.

The test results have been compared and analyzed, indicating that DMOROA has the following characteristics.

1. For the multi-objective HFSP, in general, the higher complexity of the problem has, the higher search precision of DMOROA has.
2. Compared with classical multi-objective heuristic algorithm, DMOROA can show excellent performance in solving the HFSP that have the same complexity.
3. If DMOROA can obtain more non-dominated solutions, Pareto front of DMOROA has more diversity. But with the increase of the complexity of the HFSP, the number of non-dominated solutions of DMOROA will decrease.
4. When HFSP has low complexity, DMOROA can obtained the uniform and diverse solution set easily. With the increase of the HFSP complexity, the solution set of DMOROA decreases the uniformity slightly.
5. DMOROA has excellent stability in solving HFSP.

VII. CONCLUSION

This paper studies green HFSP from the perspective of labor right, and proposes a HFSP considering makespan, noise and dust pollution. The MOROA is presented based on ROA, and MOROA not only keeps the advantages of ROA, but also improves the ability of handling complex problem. DMOROA is introduced to solve HFSP considering makespan, noise and dust pollution.
The simulation experiment is carried out with 10, 30 and 50 jobs HFSP considering the makespan, noise and dust pollution between DMOROA, NSGA2 and MOPSO. The experimental results show that DMOROA is superior to NSGA2 and MOPSO in search accuracy, number of non-dominated solution and stability. Therefore DMOROA is suitable to solve HFSP considering makespan, noise and dust pollution. When the HFSP considering makespan, noise and dust pollution increases complexity, the number of non-dominated solutions and diversity of solution set are slightly reduced. In the future research, this problem is worth studying and solving. The green scheduling factors considered in this paper are limited. The factors such as energy efficiency and radioactive substance can be introduced as the optimization objectives in the further work [47].

REFERENCES

[1] Y. Han, “Multi-objective batch flowshop scheduling based on evolutionary optimization,” Ph.D. dissertation, Dept. Inf. Commun., Xuzhou Univ., CUMT, Beijing, China, 2016.

[2] A. Jaszkiewicz and T. Lust, “ND-Tree-Based update: A fast algorithm for the dynamic nondominance problem,” IEEE Trans. Evol. Comput., vol. 22, no. 5, pp. 778–791, Oct. 2018.

[3] Y. Gao, L. Peng, F. Li, MiaoLiu, and W. Li, “A multi-objective PSO with Pareto archive for personalized E-Course composition in moodle learning system,” in Proc. 8th Int. Symp. Comput. Intell. Design (ISCID), vol. 2, Dec. 2015, pp. 21–24.

[4] A. Jaszkiewicz, “Many-objective Pareto local search,” Eur. J. Oper. Res., vol. 271, no. 3, pp. 1001–1013, Dec. 2018.

[5] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, “A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems,” IEEE/CIAA J. Automatica Sinica, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[6] J. N. D. Gupta, “Two-stage, hybrid flow shop scheduling problem,” J. Oper. Res. Soc., vol. 39, no. 4, pp. 359–364, Apr. 1988.

[7] R. Linn and W. Zhang, “Hybrid flow shop scheduling: A survey,” Comput. Ind. Eng., vol. 37, nos. 1–2, pp. 57–61, Oct. 1999.

[8] R. Ruiz and C. Maroto, “A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility,” Eur. J. Oper. Res., vol. 169, no. 3, pp. 781–800, Mar. 2006.

[9] O. Engin and A. Döyen, “A new approach to solve hybrid flow shop scheduling problems by artificial immune system,” Future Gener. Comput. Syst., vol. 20, no. 6, pp. 1083–1095, Aug. 2004.

[10] O. Moursli and Y. Pochet, “A branch-and-bound algorithm for the hybrid flowshop,” Int. J. Prod. Econ., vol. 64, nos. 1–3, pp. 113–125, Mar. 2000.

[11] Z. Jin, Z. Yang, and T. Ito, “Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem,” Int. J. Prod. Econ., vol. 100, no. 2, pp. 322–334, Apr. 2006.

[12] M. K. Marichelvam, M. Geetha, and T. Ömür, “An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problem with the effect of human factors–A case study,” Comput. Oper. Res., vol. 104, Feb. 2020, Art. no. 104812.

[13] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, “A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems,” IEEE Trans. Evol. Comput., vol. 18, no. 2, pp. 301–305, Apr. 2014.

[14] B. Zhang, Q.-K. Pan, L. Gao, X.-Y. Li, L.-M. Meng, and K.-K. Peng, “A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem,” Comput. Ind. Eng., vol. 156, pp. 325–344, Oct. 2019.

[15] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, “A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution,” Appl. Soft Comput., vol. 75, pp. 728–749, Feb. 2019.

[16] J.-Q. Li, X.-R. Tao, B.-X. Jia, Y.-Y. Han, C. Liu, P. Duan, Z.-X. Zheng, and H.-Y. Sang, “Efficient multi-objective algorithm for the lot-streaming hybrid flowshop with variable sub-lots,” Swarm Evol. Comput., vol. 52, Feb. 2020, Art. no. 100600.
[39] Y. Li, S. Liao, and G. Liu, “Thermo-economic multi-objective optimization for a solar-dish brayton system using NSGA-II and decision making,” *Int. J. Electr. Power Energy Syst.*, vol. 64, pp. 167–175, Jan. 2015.

[40] O. Sener and V. Koltun, “Multi-task learning as multi-objective optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, Dec. 2018, pp. 527–538.

[41] V. Indragandhi, R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, P. Siarry, and L. Uden, “Multi-objective optimization and energy management in renewable based AC/DC microgrid,” *Comput. Electr. Eng.*, vol. 70, pp. 179–198, Aug. 2018.

[42] Y. Hou, N. Wu, M. Zhou, and Z. Li, “Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[43] V. Indragandhi, R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, P. Siarry, and L. Uden, “Multi-objective optimization and energy management in renewable based AC/DC microgrid,” *Comput. Electr. Eng.*, vol. 70, pp. 179–198, Aug. 2018.

[44] Y. Hou, N. Wu, M. Zhou, and Z. Li, “Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[45] V. Indragandhi, R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, P. Siarry, and L. Uden, “Multi-objective optimization and energy management in renewable based AC/DC microgrid,” *Comput. Electr. Eng.*, vol. 70, pp. 179–198, Aug. 2018.

[46] G. G. Tejani, S. Kumar, and A. H. Gandomi, “Multi-objective modified heat transfer search for truss optimization,” *Eng. Comput.*, early access, Mar. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s00366-020-01010-1

[47] K. Gao, Y. Huang, A. Sadollah, and L. Wang, “A review of energy-efficient scheduling in intelligent production systems,” *Complex Intell. Syst.*, Sep. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s40747-019-00122-6

**YANMING FU** received the Ph.D. degree from Sichuan University, in 2011. He is currently an Associate Professor with Guangxi University. He has published a number of journal articles and conference papers. His research interests include data mining, computation intelligence, and network security.

**ZHUOHANG LI** is currently a Graduate Student with Guangxi University. His research interests are the optimization and application of swarm intelligence algorithm.

**NINGJIANG CHEN** (Member, IEEE) received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2006. He is currently a Professor with Guangxi University. His research interests include intelligent software engineering, big data, and cloud computing. He is a member of ACM and the Chinese Computer Federation.

**CHIWEN QU** received the M.S. degree in computer software and theory from Southwest Jiaotong University, in 2008. He is currently an Associate Professor with Baise University. He has published over 20 journal articles and conference papers. He also hosted or participated in many National Natural Science Fund Projects and Provincial Nature Fund Projects. His research interests include computation intelligence, neural networks, and intelligence optimization. He is a member of the Chinese Computer Society.

---

* * *