An Efficient Meta-Heuristic for Multi-objective Flexible Job Shop Inverse Scheduling Problem

Rui Wu¹, Yibing Li¹, Shunsheng Guo¹ and Xixing Li²
¹School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China
²School of Mechanical Engineering, Hubei University of Technology, Wuhan, 430068, China
Corresponding author: Yibing Li (e-mail: ahlyb@whut.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grant 51805152, in part by the Science and Technology Support Program of Hubei Province in China under Grant 2015BAA063, and in part by the Fundamental Research Funds for the Central Universities under Grant 2018HI31GX.

ABSTRACT In reality, uncertainties may still encounter after a scheduling scheme is generated. These may make the original schedule non-optimal or even impossible. Traditional scheduling methods are not effective in dealing with these situations. In response to this phenomenon, by introducing the idea of inverse optimization into the scheduling field, a new scheduling strategy called “inverse scheduling” has been proposed. To the best of our knowledge, this is the first study to be conducted on flexible job shop inverse scheduling problem (FJISP). In this paper, first, a comprehensive mathematical model with adjustable processing time is established. Then, a hybrid multi-objective evolutionary algorithm based on decomposition and particle swarm optimization is adopted for solving FJISP. To make the proposed algorithm solving FJISP more efficiently, some new strategies are used. A three-dimension coding scheme is employed to represent the particles, multiple strategies are designed for generating a high-quality initial population, and effective discrete crossover and mutation operators are specially designed according to the FJISP’s characteristics. Finally, computational experiments are carried out using extended benchmarks, the results demonstrate the effectiveness of the proposed algorithm for solving the FJISP.

INDEX TERMS Flexible job shop inverse scheduling problem, Inverse optimization, Multi-objective optimization, Multi-objective evolutionary algorithm Based on decomposition, Particle Swarm Optimization.

I. INTRODUCTION

The flexible job shop scheduling problem (FJSP) [1] is a classic abstraction of an actual production environment, and this topic has been extensively studied for decades [2]–[6] and studies on the FJSP has great significance for the actual production [7]–[9]. But, as the manufacturing market is becoming more dynamic and changeable, manufacturing companies need more flexible and practical scheduling models and strategies to remain competitive. The FJSP is becoming more and more powerless in facing the increasingly harsh environment because it is essentially based on an ideal environment, all parameters are assumed and fixed before scheduling begins. A schedule obtained in the FJSP may be optimal before production begins, however, real manufacturing systems are not consistent with ideal conditions, and the assumed parameters may change, then the original schedule may become non-optimal, or even infeasible. To deal with this situation, the theory of inverse optimization has been introduced to the field of scheduling, making possible a novel scheduling method called inverse scheduling problem (ISP) [10]. In this paper, we focus on the inverse counterparts of the FJSP, that is the flexible job shop inverse scheduling problem (FJISP). The FJISP can be seen as a supplement to the FJSP. For example, the producer obtains an original schedule only by estimating the value of parameters at the beginning of production planning. When production begins, the real value of parameters may vary slightly from the estimated value, so the original schedule is no longer optimal. Usually, the reality is that original schedule cannot be changed randomly due to technological restrictions or cost limitation. Thus, the producers may consider adjusting job processing times, speeding up some of them by using additional resources or slowing down others, so that the original schedule become optimal for the adjusted processing parameters, the inverse
From the literature review above, we can see that only a few articles are concerned with the ISP, and these studies are mostly focused on simple scheduling environment, studies about solving methods are also deficient. Therefore, it is very necessary to carry out a more in-depth study of the ISP. Because the FISP is a widely used scheduling model, research on its inverse problem will further expand its scope of application and make it more adaptable, so, a study on the FJISP is valuable.

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) is a new multi-objective evolutionary algorithm proposed by Zhang and Li [18]. This approach explicitly decomposes a multi-objective optimization problem (MOP) into a number of scalar single-objective optimization sub-problems, and these sub-problems are solved by a population based evolutionary algorithm in a collaborative way. Due to its efficiency, various studies on MOEA/D has been published in recent years [19], [20], and has been dedicated to many fields, such as knapsack problem [21], [22], job shop scheduling [23], [24], [25], traveling salesman problem [26], [27], etc. The experimental and practical results all show excellent performance of the MOEA/D, therefore, MOEA/D is adopted as the framework of multi-objective evolution in this paper.

Particle swarm optimization (PSO) is a randomized population-based optimization method based on the simulation of social interactions by the flocking behavior of birds and human social interactions [28]. It was first proposed in [29] for continuous optimization problems, and has attracted increasing studies during recent years [30]–[36]. Thanks to its advantages of less parameters, simple programming and fast convergence, it has been widely used in many fields, such as Electrical and Electronic Engineering [37]–[39], Civil Engineering [40]–[42], Communication Theory [43]–[45], production scheduling [46]–[49], etc., and also has proven to be an effective method to solve the FJSP [50]–[53].

The aim of this work is to propose an efficient meta-heuristic to solve the FJISP. To our best knowledge, this may be the first study on the FJISP. Firstly, a mathematical model with three optimization objectives is established. Then, a hybrid algorithm (MOEA/D-PSO) based on MOEA/D and PSO is developed for solving FJISP. In order to make the proposed algorithm more efficient, some new strategies are used in the proposed algorithm: effective weight vector generation method and decomposition approach are employed, a three-layer encoding scheme is designed to represent the particles, several rules are proposed for generating a high-quality initial population, and advanced crossover and mutation operators are designed for a better balance between exploration and exploitation. Finally, computational experiments are carried out using extended benchmarks, computational results demonstrate the effectiveness of the proposed algorithm in solving the FJISP.

The remainder of this paper is organized as follows: The proposed mathematical formulations for the FJISP are presented in Section II. MOEA/D-PSO is developed to solve the FJISP in Section III. Computational experiments and a comparative summary are presented in Section IV. Finally, conclusions and lines for further research are drawn in Section V.
II. PROBLEM DESCRIPTION AND FORMULATION

The FJISP can be defined as a set of \( n \) independent jobs \( J = \{ J_1, J_2, \ldots, J_n \} \) processed on a set of \( m \) machines \( M = \{ M_1, M_2, \ldots, M_m \} \), each job \( J_i \) is composed by \( \theta_i \) operations which must be processed in given sequence. Each operation \( O_{ij} \), i.e. the \( j \)th operation of job \( J_i \), can be processed by one or more machines in the machine set \( M \). \( p_{ijk} \) is denoted to be the processing time of \( O_{ij} \) processed on machine \( M_k \). The scheduling consists of two sub-problems: the routing sub-problem that assigns each operation to an appropriate machine and the sequencing sub-problem that determines a sequence of operations on all the machines. The objective is to find a schedule that makes the corresponding indicators (e.g. makespan, earliness/tardiness) optimal. The main difference between the FJISP and FJSP is that the exact values of parameters (e.g. processing times, due dates) in FJISP are controllable in a feasible range and an original schedule is given. The definition of the FJISP is about converting the original schedule obtained by FJSP optimal again or generating a optimal updated schedule by adjusting the controllable parameter.

A. NOTATION

The indices, parameters, and decision variables used in the mathematical model are defined as follows:

Indices

\( i, j \) Index of jobs, \( i, j \in \{ 1, 2, \ldots, N \} \).

\( k \) Index of machines, \( k \in \{ 1, 2, \ldots, M \} \).

\( h, l \) Index of operations, \( h, l \in \{ 1, 2, \ldots, \theta_j \} \).

Parameters

\( N \) Number of jobs.

\( M \) Number of machines.

\( \theta_j \) Number of operations in job \( j \), \( j \in \{ 1, 2, \ldots, N \} \).

\( S_{jl} \) Starting time of \( l \)th operation in job \( j \) in original scheduling.

\( P_{ijkl} \) Processing time of \( l \)th operation in job \( j \) processing on machine \( k \) in initial scheduling.

\( [p_{ijkl}, p'_{ijkl}] \) Feasible range of original processing time \( p_{ijkl} \).

\( x_{ijkl} \) 1 if \( l \)th operation in job \( j \) processing on machine \( k \) in initial scheduling, 0 otherwise.

Decision variables

\( C_{max} \) Makespan in inverse scheduling.

\( C_j \) Completion time of job \( j \) in inverse scheduling.

\( S_{jl} \) Starting time of \( l \)th operation in job \( j \) in inverse scheduling.

\( p_{ijkl} \) Processing time of \( l \)th operation in job \( j \) processing on machine \( k \) in inverse scheduling.

\( x_{ijkl} \) 1 if \( l \)th operation in job \( j \) processing on machine \( k \) in inverse scheduling, 0 otherwise.

\( y_{ijlk} \) 1 if \( l \)th operation in job \( i \) arranged just before \( l \)th operation in job \( j \) on machine \( k \), 0 otherwise.

B. MATHEMATICAL MODEL

Three optimization objectives are considered simultaneously in the FJISP. The first one is the original objective which has been considered in the forward scheduling, in this paper, we choose makespan as the first optimization objective:

\[
\min Z_1 = C_{max}
\]  

In the FJISP, we may need to adjust the controllable parameters to obtain the optimal solution. However, the adjustment of parameters should be as small as possible, since greater amounts of adjustments lead to a higher cost. Such as, reducing processing time usually means greater consumption of human resources or power. Therefore, the goal of minimizing the adjustment of parameters should also be considered. In this paper, we consider the processing time as the only adjustable parameter, so the second optimization objective can be defined as follows:

\[
\min Z_2 = \sum_{j=1}^{N} \sum_{l=1}^{M} \sum_{k=1}^{M} x_{ijkl} \times |p_{ijkl} - p'_{ijkl}|
\]

In order to ensure the continuity and stability of the entire scheduling process, the deviation between the original and updated schedules must be considered. This objective is necessary because when an operation is assigned to a machine, then the necessary tools, fixtures, raw materials, and operators should be ready. An adjustment of the processing sequence or the allocation of operations on machines will lead to a re-allocation of plant resources, resulting in additional costs. Therefore, the degree of deviation between the original schedule and updated schedule should be taken into consideration. In this paper, a deviation index is designed according to the characteristics of this problem. The index consists of two parts: the operation-sequence-deviation index and the machine-load-deviation index.

The operation-sequence-deviation index is the sum of deviations in the starting time of all operations between the original schedule and the updated schedule. The formula is as follows:

\[
I_{osd} = \sum_{j=1}^{N} \sum_{l=1}^{M} \eta_{jl} \times |S_{jl} - S'_{jl}|
\]

In the above formula, \( \eta_{jl} \) represents the penalty factor for the \( l \)th operation in job \( j \), whose value depends on the importance of the operation; for the sake of simplicity, we set \( \eta_{jl} = 1, \forall j, l, S_{jl} \) and \( S'_{jl} \) are described in Subsection II-A.

The machine-load-deviation index is the sum of deviations in the number of operations performed on all machines between the original schedule and the updated schedule. The formula is as follows:

\[
I_{mile} = \sum_{k=1}^{M} \sigma_k \times \sum_{j=1}^{N} \sum_{l=1}^{M} |x_{ijkl} - x'_{ijkl}|
\]

In the above formula, \( \sigma_k \) is the penalty factor for the \( k \)th machine, which depends on the importance of the machine. Again, to simplify the problem, we set \( \sigma_k = 1, \forall k \). The total deviation between the original schedule and the updated
schedule is the sum of the operation-sequence-deviation index and machine-load-deviation index, so the objective function can be formulated as:

\[ \min Z_3 = I_{osd} + I_{mld} \]  

The overall combined objective functions considered in this paper are to minimize the makespan, adjustment of processing times, and total deviation:

\[ \min F = \min \{Z_1, Z_2, Z_3\} \]  

To compute the objectives, we develop the following formulas and restrictions:

\[ C_{max} = \max \{C_j\} \]  
\[ C_j = S_{j+1} + \sum_{k=1}^{M} p_{jk} x_{jk} \]  
\[ S_{j+1} \geq S_{j} + \sum_{k=1}^{M} p_{jk} x_{jk} \]  
\[ p_{jk} \leq p_{j} \leq p_{j} \]  
\[ S_{j} \geq S_{i} + p_{i} \]  
\[ \sum_{k=1}^{M} x_{jk} = 1 \]  

Equation (7) states that the makespan \( C_{max} \) equals the maximum completion time of all jobs. Equation (8) states that \( C_j \) equals the sum of the starting time and processing time of the last operation in job \( j \). Constraint (9) ensures that an operation cannot start until its pre-operation in the same job is finished. Constraint (10) says that the processing time must be within the feasible range. Constraint (11) states that an operation must start after its pre-operation on the same machine is finished. Constraint (12) indicates that an operation must be performed on one and only one machine.

### III. PROPOSED MOEA/D-PSO FOR FJISP

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) is an efficient algorithm for solving multi-objective problems (MOP), it uses a decomposition method to decompose the multi-objective problem into a number of scalar single objective sub-problems, then the single objective algorithms and fitness calculation method can be easily used in the MOEA/D [18]. Therefore, MOEA/D is adopted for solving FJISP in this paper, and PSO is employed for optimizing these sub-problems simultaneously. The details of MOEA/D-PSO are described in the following subsections.

To describe the proposed algorithm clearly, an example with three jobs, three machines is presented in Table 1, \( p \) represents the processing time and \([ p, \bar{p}]\) is the feasible range of \( p \). The details of MOEA/D-PSO are described in the following subsections.

<table>
<thead>
<tr>
<th>Job</th>
<th>Operation</th>
<th>( M_1 )</th>
<th>( M_2 )</th>
<th>( M_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( o_{11} )</td>
<td>3</td>
<td>([2.5, 3.5])</td>
<td>7</td>
<td>([6.0, 8.0])</td>
</tr>
<tr>
<td>( o_{12} )</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>([4.5, 5.5])</td>
</tr>
<tr>
<td>( o_{13} )</td>
<td>4</td>
<td>([3.5, 4.5])</td>
<td>3</td>
<td>([2.8, 3.7])</td>
</tr>
<tr>
<td>( o_{21} )</td>
<td>9</td>
<td>([7.5, 9.0])</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( o_{31} )</td>
<td>7</td>
<td>([6.5, 7.9])</td>
<td>7</td>
<td>([6.8, 7.5])</td>
</tr>
<tr>
<td>( o_{32} )</td>
<td>9</td>
<td>([7.9, 9.1])</td>
<td>6</td>
<td>([5.1, 6.3])</td>
</tr>
<tr>
<td>( o_{33} )</td>
<td>7</td>
<td>([6.5, 7.2])</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### A. ALGORITHMIC FRAMEWORK

The main characteristic of MOEA/D is that it decomposes the MOP into a number of single-objective optimization sub-problems. By choosing \( N \) weight vectors \( \lambda_1, \lambda_2, \ldots, \lambda_N \), every single objective sub-problem has a different weight vector, and each weight vector applies to only one individual in the current population. MOEA/D solves these sub-problems simultaneously by evolving a sub-population of each solution, at each generation, the sub-population is composed of the best solution found so far for each sub-problem [23]. By using a specified polymerization method, such as a weighted-sum or Tchebycheff approach [18], the fitness calculation method can be used in the evolution of these sub-problems, which greatly reduces the computational complexity. The overall procedure of MOEA/D-PSO is as follows:

**Algorithm 1 MOEA/D-PSO**

**input:**
1. MOP;
2. A termination criterion;
3. H: User-defined positive integer;
4. T: The number of weight vectors in the neighborhood of each weight vector.

**output:** EP: An external population used to store non-dominated solutions.
5. Set parameters, set \( EP = \emptyset \).
6. Generate weight vectors \( \omega = \{\lambda_1, \lambda_2, \ldots, \lambda_N\} \) using the method described in Subsection III-B, and get the size \( N \) of \( \omega \).
7. Compute the Euclidean distances between any two weight vectors, and then determine the closest weight vectors to each weight vector. For each \( i = 1, 2, \ldots, N \), set \( B(i) = \{i_1, i_2, \ldots, i_T\} \), where \( \lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_T} \) are the \( T \) closest weight vectors to \( \lambda_i \).
8. Generate an initial population \( \{x_1, x_2, \ldots, x_N\} \) by the problem-specific method presented in Subsection III-E. Set \( FV_i = F(x_i) \), \( FV_i \) is the objective function values.
9. Initialize the reference point \( z^* = (z_1, z_2, \ldots, z_m)^T \); the elements in vector \( z^* \) can be calculated by \( z_i = \min \{f_1(x_1), f_2(x_2), \ldots, f_m(x_N)\} \). continued on next page
10: repeat
11:  for $i = 1 \rightarrow N$ do
12:   Selection: Treat all individuals $x_k$, where $k \in B(i)$, as a subpopulation, and calculate the number of dominating individuals of each $x_k$. Order the subpopulation by the obtained number and take the first individual as the global best individual $gb$, and the second individual as the personal best individual $pb$. Finally, randomly select an individual as the current individual $c$, which is not $pb$ or $gb$.
13:   Reproduction: Generate a new individual $y$ from $c$, $pb$, and $gb$ by PSO, as described in Subsection III-G.
14:   Update of $z^*$: For each $j = 1, 2, \ldots, m$, if $z_j < f^*_j(y)$, then set $z_j = f^*_j(y)$.
15:   Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{bp}(y|x_j, z^*) \leq g^{bp}(x_j|x_j, z^*)$, then set $x_j = y$ and $FV_j = F(y)$.
16:   Update of $EP$: Remove all the vectors dominated by $F(y)$ from $EP$; add $F(y)$ to $EP$ if no vectors in $EP$ dominate $F(y)$.
17:  end for
18: until Termination criteria are not satisfied

### B. GENERATE WEIGHT VECTORS

Weight-vector generation is an important step in the realization of MOEA/D, and it is better to use uniformly distributed weight vectors rather than non-uniformly distributed weight vectors, especially in high-dimension problems [54]. To obtain uniformly distributed weight vectors, a method of taking points on plane $f_1 + f_2 + \ldots + f_m = 1$ or surface $f_1^2 + f_2^2 + \ldots + f_m^2 = 1$ is generally adopted. The details are described below:

**Algorithm 2 Weight vector generation**

**input:**
1: $H$: User-defined positive integer;
2: $m$: Dimension of the MOP

**output:** $\omega$: A set of weight vectors.

3: Generate a set of weight vector element values
4: $\delta = \{0, \frac{1}{H}, \frac{2}{H}, \ldots, \frac{H-1}{H}\}$

5: for $i_1 = 1 \rightarrow H + 1$ do
6:  for $i_2 = i_1 + 1 \rightarrow H + 1$ do
7:   \ldots
8:   for $i_m = i_{m-1} + 1 \rightarrow H + 1$ do
9:    if $\delta_{i_1} + \delta_{i_2} + \ldots + \delta_{i_m} = 1$ then
10:     Put $\{\delta_{i_1}, \delta_{i_2}, \ldots, \delta_{i_m}\}$ into $\omega$.
11:  end if
12: end for
13: \ldots
14: end for
15: end for
16: Output $\omega$.

In the process outlined above, the total number $N$ of weight vectors is controlled by $H$, or more precisely, the number $N$, the user-defined positive integer $H$, and the problem dimension $m$ satisfy the formula $N = C_{H+m-1}^m$.

### C. DECOMPOSITION APPROACH

There are several approaches to convert the MOP into a number of scalar optimization problems, the penalty-based boundary intersection approach proposed by [18] is employed in this article. Compared to the commonly used Tchebycheff approach, the resultant optimal solutions in the penalty-based boundary intersection approach can be much more uniformly distributed, particularly when the number of weight vectors is not large. It can be calculated using the following formula:

$$\min g^{bp}(x|\bar{\lambda}, z^*) = d_1 + \theta d_2$$

where

$$d_1 = \frac{\| (z - F(x))^T \bar{\lambda} \|}{\| \bar{\lambda} \|}$$

$$d_2 = \| F(x) - (z^* - d_1 \bar{\lambda}) \|$$

In the equations above, $\bar{\lambda}$ is a weight vector generated in Subsection III-B, and $\theta$ is a preset penalty value.

### D. NORMALIZATION MECHANISM

When MOEA/D is directly applied to the engineering optimization problems with different dimensions and orders of magnitude, it is easier to evolve towards the objective with a higher order of magnitude. In this way, we can only get the pareto frontier with poor dispersion, therefore, normalization is needed in the MOEA/D. In this paper, we use a simple normalization mechanism based on non-dominated solutions in the current population. Let $z_{iL}$ and $z_{iU}$ be the minimum and maximum values of the $i$-th objective $f_i(x)$, $i \in \{1, 2, \ldots, m\}$ among non-dominated solutions in the current population, respectively. In this setting, $z_{iL} = (z_{1L}, \ldots, z_{mL})$ and $z_{iU} = (z_{1U}, \ldots, z_{mU})$ can be viewed as estimated ideal and nadir points from the nondominated solutions in the current population, respectively. Using $z_{iL}$ and $z_{iU}$, the objective value $z_i$ of the $i$-th objective $f_i(x)$ is normalized as

$$z_i' = \frac{z_i - z_{iL}}{z_{iU} - z_{iL} + \epsilon} \quad \text{for } i = 1, 2, \ldots, m$$

where $\epsilon$ is a very small positive value for preventing the denominator from becoming zero in the case of $z_{iL} = z_{iU}$.

### E. ENCODING

One of the key problems in applying PSO is encoding, which is the conversion between solutions and vectors. To obtain a solution for the FIISP requires sorting all operations, selecting a machine for each operation, and determining the adjustment of processing time. Therefore, we propose a three-layer encoding scheme. The first layer is an operation-sequence vector to determine the processing sequence of
operations; The second layer is a machine-assignment vector based on the allocation of machines, this vector determines the selection of a machine for each operation; The third layer is a parameter-adjustment vector, which determines the adjustment of processing time for each operation. A particle is composed of this three vectors, and it is a feasible solution for the FJISP.

1) Encoding of operation sequence
The number of elements in the operation-sequence vector equals the total number of operations. All operations are represented by their index of the job, and the repetition number of the index is the total number of operations in this job. Scan the operation-sequence vector from left to right, the first character ‘j’ in the vector represents the first operation in job j, the second character ‘j’ represents the second operation in job j, and so on for the remaining operations in job j. Fig. 1a shows one possible operation-sequence vector for the example in Table 1, in this vector, characters ‘1’ represent operations in job 1, and the three characters ‘1’ represent the 1st, 2nd, and 3rd operations in job 1 in order, the same as 2 and 3.

<table>
<thead>
<tr>
<th>0₀₁₁</th>
<th>0₀₁₂</th>
<th>0₀₁₃</th>
<th>0₁₀₂</th>
<th>0₁₀₃</th>
<th>0₀₁₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) Operation sequence vector

<table>
<thead>
<tr>
<th>w₁</th>
<th>w₂</th>
<th>w₃</th>
<th>w₄</th>
<th>w₅</th>
<th>w₆</th>
<th>w₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Machine assignment vector

| -0.1 | -0.2 | +0.4 | -0.3 | +0.2 | -0.7 | -0.2 | +0.3 |

(c) Parameter adjustment vector

FIGURE 1: Coding example.

2) Encoding of machine allocation
The machine-assignment vector is as long as the operation-sequence vector. Each element in the machine-assignment vector represents the selected machine number of the operation at the corresponding place in the operation-sequence vector, and it determines the processing machine for the operation. It should be pointed out that the machine number must be a legal number in the corresponding machine set. An example is shown in Fig. 1b.

3) Encoding of parameter adjustment
Each element in this vector represents the processing-time adjustment of the corresponding operation and machine. An element’s value represents the variable relative to the original processing time, and the actual processing time is increased or reduced depending on its sign. It should be noted that an element’s value must be in the feasible range. An example is presented in Fig. 1c.

F. INITIALIZATION
Population initialization has a great impact on the quality of the solution. The commonly used method is to generate the population randomly, but this does not ensure stable population quality. We will adopt a combination of random generation and strategic choice to generate the initial population. Inverse scheduling utilizes a original schedule, and since this original schedule was once optimal, it is highly likely that its neighbor individuals will be optimal now. What’s more, one of the objectives is minimizing the deviation between the original schedule and updated schedule, hence, if we generate individuals based on the original schedule, then the deviation is naturally very small. Therefore, the operation-sequence vector and machine-assignment vector in 20% individuals are generated by copy from original schedule; next, we will discuss the method of generating the remaining 80% individuals. First, the operation-sequence vector is generated using the mixed strategy proposed by [2], which is composed of the following three rules: a)

1) Choose a job randomly;
2) Give preference to the job with the longest processing time of the remaining operations;
3) Give preference to the job with the largest number of rest operations.

Using the above three strategies to generate operation-sequence vectors respectively accounted for 10%, 35%, and 35% of the initial population. Second, for the machine-assignment vector, 30% are randomly selected in the constraint set; the other 50% are generated by the method proposed in [55], the implementation process is as follows: scan the operations in the operation-sequence vector in order, select the machine with the shortest processing time for the current operation, then add the current processing time to all the rest operations which can be processed on this machine, the selection of machine for next operation is based on the updated processing-time matrix. The method of random generation within the feasible range is adopted for all parameter-adjustment vectors.

G. EVOLUTIONARY OPERATORS IN PSO
1) Discrete location update strategy
The traditional particle swarm algorithm is suitable for optimizing continuity problems, and its original location-update strategy cannot be applied to solve the FJISP because the FJISP is a discrete problem. Therefore, this paper applies a new discrete location-update strategy to the FJISP, based on the proposed encoding rules in Subsection III-E. The particle-position update formula can be defined as follows:

\[
y = c₂ \odot f₃ \{c₁ \odot f₂ \{w \odot f₁(c), pb\}, gb\}
\]  

(17)

In the above formula, w, c₁, and c₂ are given coefficients such that w, c₁, c₂ \( \in \{0, 1\}\), and c, pb, and gb are different particles introduced in Algorithm 1. f₁, f₂, and f₃ are update operators that are specially designed for the FJISP, details
of them will be described in later subsections. Formula (17) consists of the following three parts:

\[ E = w \odot f_1(c) = \begin{cases} f_1(c) & \text{rand} < w \\ c & \text{others} \end{cases} \]  

(18)

\[ F = c_1 \otimes f_2\{E, pb\} = \begin{cases} f_2\{E, pb\} & \text{rand} < c_1 \\ E & \text{others} \end{cases} \]  

(19)

\[ y = c_2 \otimes f_3\{F, gb\} = \begin{cases} f_3\{F, gb\} & \text{rand} < c_2 \\ F & \text{others} \end{cases} \]  

(20)

Equation (18) represents the reflections on the previous state of the current particle, \( rand \) is a random number on the interval \((0, 1)\), \( f_1 \) is equivalent to the mutation operator. Equation (19) indicates that the particle is adjusted according to the personal best \( pb \), where \( f_2 \) is a discrete crossover operator. Equation (20) indicates that the particle is adjusted according to the global best \( gb \).

2) Mutation operator \( f_1(c) \)

The main operations of \( f_1(c) \) is to fine tune the three vectors in the current particle, which is equivalent to the acceleration operation in the original PSO. It is implemented as follows:

1) Randomly select two different positions in the operation-sequence vector and exchange elements at these two positions. Adjust the machine-assignment and parameter-adjustment vector according to the new operation sequence to ensure the original machine assignment and parameter adjustment remain unchanged, as shown in Fig. 2(a) and Fig. 2(b).

2) Randomly select an operation in the operation-sequence vector, then select a new machine randomly from the available machine set to replace it’s machine, as shown in Fig. 2(c).

3) Replace the processing time adjustment of operation and machine in the previous step by a random value from the feasible range, as shown in Fig. 2(c).

3) Crossover operator \( f_2\{E, pb\} \)

Compared with other evolutionary algorithms, PSO has the best characteristics of learning from the global best position and the personal best position. To maintain this good characteristic, the particle-position adjustment method designed here retains the information in the personal best as much as possible. Furthermore, taking the characteristics of the FJISP into account, multiple machine-assignment operations are performed on a single operation-sequence vector to obtain more combinations of the operation-sequence and machine-assignment vectors, expanding the search range and helping to prevent the algorithm from falling into a local optimum.

\( f_2\{E, pb\} \) represents the learning process of the current particle from the personal best \( pb \). We use the improved precedence-preserving-order-based crossover (POX) [56] to adjust the current particle. We describe the details of the improved POX crossover below and provide a simple example in Fig. 3.

1) Divide the job set \( 1, 2, \ldots, N \) into two non-empty complementary subsets \( J_1 \) and \( J_2 \). Copy all operations in \( J_1 \) of particle \( E \) into \( \varphi_1 \) in order, and copy all corresponding machine-assignment and parameter-adjustment information at the same time. Obtain \( \varphi_2 \) in the same way for personal best \( pb \).

2) Fill \( F \) by randomly selecting elements from \( \varphi_1 \) and \( \varphi_2 \), as shown in Fig. 3. The sequence of elements in \( \varphi_1 \) and \( \varphi_2 \) must be preserved, but the elements between the sets have no order-relation constraints.

4) Crossover operator \( f_3\{F, gb\} \)

The details of \( f_3\{F, gb\} \) are described below:

1) Randomly generate a vector \( \vec{r} \) consisting of a decimal number in \((0,1)\), whose length equals the particle dimension.

2) Find all positions in \( \vec{r} \) with values less than \( pt \), where \( pt \) is a dynamic variable calculated as follows:

\[ pt = pt_{\text{max}} + (pt_{\text{max}} - pt_{\text{min}}) \times \frac{\text{gen}}{\text{iter}} \]  

(21)

Identify all operations in the selected positions and copy the corresponding machine indices and processing-time adjustments from \( gb \) into \( y \). Fig. 4 shows an example with \( pt = 0.5 \). In equation (21),
$p_{t_{\text{max}}}$ and $p_{t_{\text{min}}}$ are the maximum and minimum adaptive adjustment probabilities respectively, $\text{gen}$ is the current iteration number, and $\text{iter}$ is the maximum number of iterations of Algorithm 1. From the above equation, we can see that $p_t$ will dynamically adjust itself with the number of current iterations, $p_t$ is larger in the initial search, and more elements will be involved in the crossover operation, which is more suitable for a global search. At the end of the search process, $p_t$ is closer to $p_{t_{\text{min}}}$. The small value means there are fewer elements to be crossed, and it is better for local search.

**IV. EXPERIMENTAL STUDIES AND DISCUSSION**

In this section, experiments are designed and developed to verify the effectiveness of the proposed algorithm. Since no algorithms have been used to solve the multi-objective FJISP, we employed the NSGA-II and SPEA-II, which are commonly used in multi-objective FJISP, as contrast algorithms. The encoding and population-initialization method in NSGA-II and SPEA-II are the same as MOEA/D-PSO. The crossover and mutation operators in [57] are used for the machine-assignment operators in [58] are adopted for the machine-assignment vector, and a random multi-point crossover, like $f_3$ in Subsection III-G4, a random-variation operator is designed for the parameter-adjustment vector. The repair strategy in [57] is used in NSGA-II and SPEA-II to ensure the feasibility of solutions. In addition, because the influence of normalization is strongly problem dependent [59], the algorithm (MOEA/D-PSO-R) which removes the normalization mechanism from MOEA/D-PSO is used as a comparison algorithm to verify the effectiveness of normalization. To verify the effectiveness of the improvement strategies in the proposed algorithm, IMOEa/D in [23] is modified to be suitable for solving this problem and employed as a comparison algorithm. Details of the experiment are described in the following subsections.

**A. TESTING INSTANCES**

We are aware of no common benchmarks for testing FJISP. Therefore, benchmarks for FJSP are extended and adapted for this purpose. We choose 36 commonly used benchmarks from [60], [61], and [62] for expansion. The original schedule and feasible range of processing time need to be added on the basic benchmarks, the extending procedures are described below: Step 1:

1) For each instance in basic benchmarks aiming to minimize the makespan, use any method (genetic algorithm proposed in [63] is used in this paper) to obtain an original schedule. Then we can obtain the parameters $x_{jlk}$ and $S_{jl}$ from the original schedule. The quality of this original schedule is not important, it just needs to be legal.

2) Generate the feasible range of the processing time by floating up and down in the original processing time with a certain probability. For a given original processing time $p_{jlk}$ in the basic benchmarks, generate a random number $r_{\text{nd}}$, $r_{\text{nd}} \in (0, 1)$. If $r_{\text{nd}}$ is less than a given probability (0.5 in this paper), for simplicity, let $p'_{jlk} = p_{jlk} + 1.2$ and $p''_{jlk} = p_{jlk} \times 0.8$. Otherwise, $p'_{jlk} = p_{jlk} = p_{jlk}$, which means the processing time is not changeable. Thus, we obtain an feasible range of the processing time.

The testing instances can be obtained using the method described above. Due to space limitations, we do not list the generated data in this paper, but upload it as supplementary material and available at http://ieeexplore.org/1127.

**B. PARAMETER SETTING**

The selection of parameters has a significant impact on the performance of algorithms. To obtain parameters that provide the best performance within a reasonable computation time, we performed computational tests using uniform design method [64] on instances generated from the above subsection. The final parameters in MOEA/D-PSO, NSGA-II, and SPEA-II are shown in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>13</td>
</tr>
<tr>
<td>$T$</td>
<td>15</td>
</tr>
<tr>
<td>Maximum number of iterations $\text{iter}$</td>
<td>200</td>
</tr>
<tr>
<td>Penalty value $\theta$</td>
<td>0.5</td>
</tr>
<tr>
<td>$u_1$</td>
<td>0.15</td>
</tr>
<tr>
<td>$c_1$</td>
<td>0.8</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0.5</td>
</tr>
<tr>
<td>$p_{t_{\text{max}}}$</td>
<td>0.8</td>
</tr>
<tr>
<td>$p_{t_{\text{min}}}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Population size in NSGA-II</td>
<td>120</td>
</tr>
<tr>
<td>Population size in SPEA-II</td>
<td>140</td>
</tr>
<tr>
<td>External population size of SPEA-II</td>
<td>100</td>
</tr>
</tbody>
</table>

**C. PERFORMANCE EVALUATION INDICATORS**

MOP is different from the single objective optimization problem, as the quality of solutions in a multi-objective optimization problem must focus on two requirements: 1) the obtained
non-dominated solution set should be as close as possible to the true Pareto front; 2) the non dominated solution-set should be distributed as evenly as possible. To compare the quality of the solutions obtained by three algorithms, we adopt the following three kinds of performance-evaluation indicators to compare the convergence and distribution of solution sets.

1) Inverse generational distance
The inverse generational distance (IGD) [65] is a variant of the generational distance (GD), according to [66]. IGD is a measure of the distance between the approximate Pareto front, obtained by the proposed algorithm, and the real Pareto front. A lower IGD indicates better convergence and diversity of the approximate Pareto front, meaning it is closer to the real Pareto front. Define $P_f$ as the true Pareto front, and let $\Omega$ represent the solutions obtained by the algorithms. A known $P_f$ is needed to calculate IGD. A lack of research on the inverse scheduling problem means there is no available $P_f$ for the FJISP, so we combine the solutions obtained by the three algorithms, perform a non-dominate sort, and select all non-dominated solutions as $P_f$. IGD denotes the distance between $P_f$ and solution set $\Omega$, and it can be calculated as follows:

$$IGD = \frac{1}{|P_f|} \sum_{i=1}^{|P_f|} Dist_i \quad (22)$$

$$Dist_i = \min_{j=1}^{|\Omega|} \left( \sum_{m=1}^M \left( \frac{f_m(p_i) - f_m(a_j)}{f_{m_{\max}} - f_{m_{\min}}} \right)^2 \right)^{1/2} \quad (23)$$

In equation (23), $Dist_i$ represents the minimum normalized Euclidean distance, $f_{m_{\max}}$ is the maximum value of the $m$th objective in all solutions of $P_f$, and $f_{m_{\min}}$ is the respective minimum value. $f_m(p_i)$ and $f_m(a_j)$ represent the respective values of the $m$th objective for solutions $p_i$ and $a_j$, $m = 1, 2, \ldots, M$, where $M$ is the number of objectives, $p_i \in P_f$, $i = 1, 2, \ldots, |P_f|$ and $a_j \in \Omega, j = 1, 2, \ldots, |\Omega|$.

2) Number of non-dominated solutions
Define $N_{nds}$ as the number of non-dominated solutions, which is the number of solutions in $\Omega$ that are not dominated by a solution in $P_f$. Obviously, a larger value of $N_{nds}$ indicates better performance of the algorithm.

3) Error ratio
The error ratio can be defined as follows:

$$ER = \frac{\sum_{i=1}^Q e_i}{Q} \quad (24)$$

In equation (24), $Q$ is the total number of solution sets of $\Omega$ obtained by the testing algorithm. $e_i$ is a binary value such that $e_i=0$ if solution $i$ in $\Omega$ is not dominated by any solution in $P_f$, and $e_i=1$ otherwise. For the indicator $ER$, the smaller the better.

### D. MAIN RESULTS AND DISCUSSION

In order to reduce the impact of randomness on experimental results, we run 30 replications of each algorithm for each instance independently, the final result is the average of the 30 replications.

1) Comparison with NSGA-II and SPEA-II
The final summary results are shown in Table 3. As shown in Table 3, the average of all $IGD$ results obtained by three algorithms are 0.09, 0.30, and 0.35, respectively, it shows that MOEA/D-PSO achieves the best results in general. More specifically, the MOEA/D-PSO achieves 34 optimal values of 36 instances on the $IGD$ indicator, this illustrates that solutions obtained by MOEA/D-PSO are closest to the real solutions and have better quality than solutions obtained by the other two algorithms. What's more, it is obvious that MOEA/D-PSO also performs best on the other two indicators in most instances, and it is most time-saving. This can be sufficient to show that MOEA/D-PSO is the most efficient of the three algorithms.

#### Table 3: Comparison results with NSGA-II and SPEA-II

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-PSO</th>
<th>NSGA-II</th>
<th>SPEA-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>$N_{nds}$</td>
<td>$ER$</td>
<td>$N_{nds}$</td>
</tr>
<tr>
<td>1</td>
<td>10a</td>
<td>0.16</td>
<td>31.97</td>
</tr>
<tr>
<td>2</td>
<td>11a</td>
<td>0.16</td>
<td>42.47</td>
</tr>
<tr>
<td>3</td>
<td>12a</td>
<td>0.17</td>
<td>53.84</td>
</tr>
<tr>
<td>4</td>
<td>13a</td>
<td>0.17</td>
<td>65.91</td>
</tr>
<tr>
<td>5</td>
<td>14a</td>
<td>0.17</td>
<td>77.60</td>
</tr>
<tr>
<td>6</td>
<td>15a</td>
<td>0.17</td>
<td>89.28</td>
</tr>
<tr>
<td>7</td>
<td>16a</td>
<td>0.18</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>17a</td>
<td>0.18</td>
<td>111.11</td>
</tr>
<tr>
<td>9</td>
<td>18a</td>
<td>0.18</td>
<td>122.22</td>
</tr>
<tr>
<td>10</td>
<td>19a</td>
<td>0.19</td>
<td>133.33</td>
</tr>
<tr>
<td>11</td>
<td>20a</td>
<td>0.19</td>
<td>144.44</td>
</tr>
<tr>
<td>12</td>
<td>21a</td>
<td>0.20</td>
<td>155.56</td>
</tr>
</tbody>
</table>

In order to check whether the observed differences from Table 3 are indeed statistically significant, one-way analysis of variance (ANOVA) is adopted in this paper. The statistical results are given in Table 4, and these indicate statistically significant differences in the indicators $IGD$, $N_{nds}$, and $ER$ at a 95% confidence level.

To compare the algorithms more intuitively, the corresponding box plots are shown in Fig. 5. From Fig. 5a we can see that MOEA/D-PSO has obtained the lowest average $IGD$ value, and no abnormal values exist, which indicates that solutions obtained by MOEA/D-PSO is the best and the quality of the solutions is most stable. Fig. 5b and Fig. 5c show that MOEA/D-PSO also achieves the optimal mean value on indicator $N_{nds}$ and $ER$, but abnormal values exist, which means the performance of MOEA/D-PSO is slightly
The running time of MOEA/D-PSO is slightly longer than MOEA/D-PSO-R, which is reasonable because normalization adds a small amount of computation. A slightly longer running time is acceptable because MOEA/D-PSO achieves better results. As a whole, the MOEA/D-PSO significantly performs better than the IMOEA/D on most instances since the MOEA/D-PSO has an overwhelming advantage over IMOEA/D in terms of the three metrics. This can be sufficient to show that MOEA/D-PSO is more effective than IMOEA/D in solving FJISP.

The results of statistical analysis are also given, as shown in Table 6. The analysis results show that MOEA/D-PSO is significantly better than MOEA/D-PSO-R and IMOEA/D.

The box plots are shown in Fig. 6. From Fig. 6 we can see that MOEA/D-PSO has obtained the lowest average $IGD$, $N_{nds}$, and $ER$, and no abnormal values exist, which indicates that MOEA/D-PSO performs the best and is the most stable one.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a novel inverse scheduling problem called FJISP is studied for the first time, and an effective hybrid multi-objective evolutionary algorithm based on decomposition and particle swarm optimization is developed to solve the problem. First, a comprehensive mathematical model for
the FJISP is established while considering three objectives simultaneously, which are minimizing the makespan, minimizing the adjustment of processing time, and minimizing the adjustment of schedule. Then, a hybrid algorithm named MOEA/D-PSO which hybridizes MOEA/D and PSO is proposed to solve this problem. In the proposed algorithm, MOEA/D is used to decompose this MOP into single objectives sub-problems, so that the solving methods which are dealing with single objective problems can be applied. PSO is adopted as an evolutionary operator under the framework of MOEA/D. To make the PSO suitable for solving the Sub-problems after decomposition, some new strategies are used: an effective three-dimension coding scheme is employed to generate a high-quality initial population, and advanced discrete crossover and mutation operators are well designed to perform efficient search operations. Finally, benchmarks for FJISP are extended for testing the performance of the proposed algorithm. The well known NSGA-II, SPEA-II are selected as contrast algorithms, and the experimental results indicate that MOEA/D-PSO has significantly better performance when solving the FJISP.

As a direction for future study, the proposed algorithm can be further improved, and other, more efficient meta-heuristic algorithms can be proposed to solve the FJISP. Other objective functions can be considered, such as cost, quality, machine load, and earliness/tardiness. Investigating the studied problem with different criteria such as an adjustable due date could be an interesting research topic. Finally, other types of production environments, like flow shop scheduling, can be considered.

REFERENCES


RUI WU received the B.Sc. degree from Wuhan University of Technology, Wuhan, China, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include manufacturing scheduling and intelligent optimization algorithms.

YIBING LI received the M.S. degree (2003) and Ph.D. degree (2008) in Mechanical Manufacturing and Automation from Wuhan University of Technology, Wuhan, China.

He is currently a vice-Professor with the School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, China. He has published about 15 journal papers. His current research interests include fault diagnosis of mechanical equipment, and intelligent manufacturing and shop scheduling.

SHUNSHENG GUO received the B.Sc. degree in Mechanical manufacturing and automation from Huazhong University of Science and Technology, Wuhan, China, in 1984 and the Ph.D. degree in Mechanical Design and Theory from Wuhan University of Technology, Wuhan, China, in 2001.

He is currently a Professor with the School of Mechanical and Electronic Engineering, Wuhan, China. He has published over 70 journal papers. His current research interests include manufacturing informatization and intelligent manufacturing.

XIXING LI received the M.S. degree (2014) and Ph.D. degree (2017) in Mechanical Engineering from Wuhan University of Technology, Wuhan, China.

He is currently a lecturer with the School of Mechanical Engineering, Hubei University of Technology, Wuhan, China. He has published about 10 journal papers. His current research interests include production planning & scheduling and optimization modeling.

***