Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6672753, 11 pages
https://doi.org/10.1155/2021/6672753

Research Article

Production Scheduling Optimization of Prefabricated Building Components Based on DDE Algorithm

Chunguang Chang and Mengyao Han

School of Management, Shenyang Jianzhu University, Shenyang 110168, China

Correspondence should be addressed to Mengyao Han; 1071963233@qq.com

Received 20 October 2020; Revised 22 January 2021; Accepted 25 January 2021; Published 8 February 2021

Academic Editor: David Bigaud

At present, the prefabricated construction industry is in a situation of increasing types of prefabricated components and generally high production costs. A hybrid optimization model considering continuity and discreteness for the production of fabricated concrete members is established to minimize production costs through the analysis of the production characteristics of precast concrete members. Under the premise of fully considering the staffing constraints, process constraints, construction period constraints, process constraints, and special process time limits for component production, the production arrangement and staffing of the components are rationalized and optimized. A discrete differential evolution (DDE) algorithm is introduced for such NP-hard problems. The double genetic chromosome coding mode and the active scheduling decoding method are adopted. Based on the improved POX (Precedence Operation Crossover) cross-evolution method, the global evolution operation is carried out, and an interchange-based local search method and continuous work penalty mechanism are designed to find the global optimal solution. The experimental results verify the practicality and effectiveness of the optimization model and algorithm.

1. Introduction

A prefabricated building is a new building form that selects large-scale factories to produce components and assemble them on-site [1]. The degree of development of prefabricated buildings represents the degree of integration and development of the upstream and downstream industrial chains of the construction industry [2]. In recent years, the prefabricated construction industry in China has entered a new stage of development under the support of national policies and has been highly valued by more and more enterprises and scholars [3]. Now, the construction industry is actively exploring the formation of a mature prefabricated construction industry system [4]. However, as far as the situation in recent years is concerned, because domestic prefabricated buildings are still difficult to form industrial-scale benefits, the construction assembly rate in China is only 5% [5], which is far lower than the level of western developed countries. The high construction cost is the root cause that affects the enthusiasm of developers and hinders the development of construction industrialization [5]. The study found that the main reason for the substantial increase in the cost of prefabricated construction is the increase in the cost of production and transportation of prefabricated components. Among them, the key to the control of the production cost of prefabricated components lies in large-scale and standardized production, thereby reducing the cost amortization of the production line and taking the advantages of prefabricated components in labor and materials saving [6].

At present, many scholars have proposed improvements and optimization methods for the production stage of prefabricated building components. Ko and Wang [7] proposed a production planning system for prefabricated components based on genetic algorithms; Jeong et al. [8] constructed a production combination optimization model for house construction to achieve the purpose of shortening the construction period, reducing inventory, and saving costs. From the perspective of production quality optimization, Rausch and Nahangi [9] optimized key interfaces and reduced errors to form an optimal assembly plan for prefabricated components, thereby reducing rework and saving costs. To solve the NP-hard problem of production scheduling, metaheuristic algorithms have been widely used in recent years, such as gaining-sharing knowledge-based optimization algorithm [10, 11], ant colony algorithm [12], particle swarm algorithm [13], differential evolution algorithm [14, 15], and various hybrid algorithms. Among them, the DE is widely used in various fields due to its good self-learning and robustness [16–20] and is continuously optimized. Wagdy et al. [21] proposed two new mutation strategies, which, respectively, enhanced the algorithm's exploration and convergence capabilities and designed a hybrid framework to improve the algorithm's ability to solve complex problems. Cheng et al. [22] designed a new DE variant (named ADEwSE) to reduce the randomness of the search direction and improve the search efficiency of the algorithm.

Based on the above research results, this paper studies the prefabricated component production scheduling optimization model that is restricted by various factors such as staffing, technology, construction period, process, and special process requirements, and there are multiple optional resources for different component processes. Based on the basic differential evolution algorithm, the discrete differential evolution algorithm (DDE) with double coding is adopted to assign and schedule the processing time and work team of each component process. The algorithm aims to reduce the waiting time of labor and machinery and the overtime of workers, to improve the production efficiency of workers and machinery, and reduce the production cost of components under the premise of ensuring timely delivery.

2. Modeling of the Component Production Scheduling

2.1. Problem Description. The main feature of prefabricated buildings is the use of mobile assembly lines in the prefabricated yard to carry out the batch and standardized production of concrete components. But there is a big difference between the production of building components and the production of mechanical parts. The first is the coexistence of continuity and discreteness of production. The production process of concrete components generally includes five processes: mold placement, finished steel reinforcement placement, concrete pouring and vibrating, concrete component maintenance, and form removal and storage. Among them, concrete pouring and curing must be carried out continuously. The second is a strong artificial dependence. Due to the relatively low level of automated production technology for building components and the high cost of fully automatic component production equipment, many processes still require manual participation, so labor costs and working time arrangements must be considered.

Based on the above characteristics, the production scheduling optimization problem of precast concrete components can be summarized as follows. Component production tasks include several components to be processed, and each component production includes mold placement, finished steel reinforcement placement, concrete pouring and vibrating, concrete component maintenance, and form removal and storage. A group of nonunique professional work teams and unique processing machinery are allocated for each process and there is a certain order constraint relationship between different processes. Thus, resources involved in each process include processing equipment and processing team. Different component processes can be processed by any work team, which takes a different time. Since concrete pouring and curing can be performed over time, the operation is continuous and the processing process cannot be interrupted. If it cannot be completed within working hours, it will be carried out the next day. The rest of the process is regarded as an interruptible process.

Also, the production scheduling model of precast concrete components has the following assumptions:

(1) In the beginning, all work teams and processing equipment can arrange tasks, and all components can be processed
(2) Processing preparation time is included in the processing time of each process
(3) Different components have the same processing priority, and there is no component processing order constraint
(4) Before the current process is completed, the team cannot participate in other processes
(5) The transmission time of components between processing sites is not considered
(6) Machinery can meet the requirements
(7) The processing time of all components is less than the sum of normal working hours and overtime hours

The scheduling optimization goal is as follows: under the premise of satisfying the constraints, each process is assigned to the most suitable work team, and the best sequence and starting time are determined so that the production cost of the entire concrete component production is minimized.

2.2. Variable and Parameter Symbol Definition

(1) Label

i, l is the task force number; $i, l = 1, \ldots, A$. A is the highest number of the task forces. j, r is the component number; $j, r = 1, \ldots, N$. N is the largest number of the components. k, w is the process number; $k, w = 1, \ldots, K$. K is the largest number of the processes.

(2) Parameter

E_{ijk}: the time required for task force i to process the kth process of j component.

CG_i: the average wage of the work team i during the working day, yuan/hour.

CC: the unit salary for overtime work of each team, yuan/hour.

CM_k: depreciation cost per unit working time of equipment k.

TW: normal working hours.

TN: nonworking time, $TN = 24 - TW$.

e: daily start time.

OT: maximum overtime hours per day.

D: latest delivery date.

M: a large positive number.

k_1: the number of the concrete pouring processes.

k_2: the number of the concrete curing processes.

h_i: the last component processed by task force i.

s_i: the last process of task force i processing.

(3) Decision variables

$$X_{ijk} = \begin{cases} 1, & \text{if the task force } i \text{ processes the } k \text{ process of } j \text{ component,} \\ 0, & \text{others.} \end{cases}$$

$$U_{ijr} = \begin{cases} 1, & \text{if the task force } i \text{ processes component } j \text{ and then processes component } r, \\ 0, & \text{others.} \end{cases}$$

$$V_{jrk} = \begin{cases} 1, & \text{if machine } k \text{ first processes component } j \text{ and then processes component r,} \\ 0, & \text{others.} \end{cases}$$

t_{ijk}: the start time of process k for task i to process component j.

f_{ijk}: the end time of process k for task i to process component j.

2.3. Mathematical Model

$$\begin{aligned} \min z = {} & \sum_{i=1}^{A}\sum_{j=1}^{N}\sum_{k=k_1}^{k_2}\left(f_{ijk}\bmod 24 - TW - e\right)\cdot CC_i \cdot \operatorname{sgn}\left\{\max\left\{\left(f_{ijk}\bmod 24 - TW - e\right), 0\right\}\right\} \\ & + \sum_{i=1}^{A}\left\{\operatorname{Int}\left(\frac{f_{ih_is_i}}{24}\right)\cdot TW + \min\left\{\left(\frac{f_{ih_is_i}}{24}\right) - e, TW\right\}\right\}\cdot CG_i + \sum_{i=1}^{A}\sum_{j=1}^{N}\sum_{k=1}^{K} X_{ijk}\cdot E_{ijk}\cdot CM_k, \end{aligned} \tag{1}$$

$$\text{s.t.} \quad f_{ijk} \le t_{irw} + M\left(1 - U_{ijr}X_{ijk}X_{irw}\right), \; i = 1, \ldots, A, j, r = 1, \ldots, N, j \ne r; k, w = 1, \ldots, K, k \ne w, \tag{2}$$

$$\sum_{i=1}^{A} X_{ijk} = 1, \quad j = 1, \ldots, N, k = 1, \ldots, K, \tag{3}$$

$$\max_{j=1}^{N}\max_{k=1}^{K}\left\{f_{ijk}\right\} = f_{ih_is_i} \le D, \quad i = 1, \ldots, A, \tag{4}$$

$$e \le t_{ijk}\bmod 24 \le e + TW, \quad i = 1, \ldots, A, j = 1, \ldots, N, k = 1, \ldots, K, \tag{5}$$

$$t_{ijk} \ge f_{l,j,k-1}, \quad i, l = 1, \ldots, A, j = 1, \ldots, N, k = 2, \ldots, K, \tag{6}$$

$$t_{ijk} \ge M\left(1 - X_{ijk}\right), \quad i = 1, \ldots, A, j = 1, \ldots, N, k = 1, \ldots, K, \tag{7}$$

$$\begin{aligned} t_{ijk} = {} & t_{ijk}\cdot\operatorname{sgn}\left\{\max\left\{\left[\operatorname{Int}\left(\frac{t_{ijk}}{24}\right)\cdot 24 + TW + e - \left(t_{ijk} + E_{ijk}\right) + OT\right], 0\right\}\right\} + \left\{\left[\operatorname{Int}\left(\frac{t_{ijk}}{24}\right) + 1\right]\cdot 24 + e\right\} \\ & \cdot\left\{1 - \operatorname{sgn}\left\{\max\left\{\left[\operatorname{Int}\left(\frac{t_{ijk}}{24}\right)\cdot 24 + TW + e - \left(t_{ijk} + E_{ijk}\right) + OT\right], 0\right\}\right\}\right\}, \quad i = 1, \ldots, A, j = 1, \ldots, N, k = k_1, k_2, \end{aligned} \tag{8}$$

$$f_{ijk} = t_{ijk} + E_{ijk}, \quad i = 1, \ldots, A, j = 1, \ldots, N, k = k_1, k_2, \tag{9}$$

$$\begin{aligned} f_{ijk} = {} & f_{ijk}\cdot\operatorname{sgn}\left\{\max\left\{\left[\operatorname{Int}\left(\frac{t_{ijk}}{24}\right)\cdot 24 + TW + e - \left(t_{ijk} + E_{ijk}\right) + OT\right], 0\right\}\right\} + \left(f_{ijk} + TN\right) \\ & \cdot\left\{1 - \operatorname{sgn}\left\{\max\left\{\left[\operatorname{Int}\left(\frac{t_{ijk}}{24}\right)\cdot 24 + TW + e - \left(t_{ijk} + E_{ijk}\right) + OT\right], 0\right\}\right\}\right\}, \\ & i = 1, \ldots, A, j = 1, \ldots, N, k = 1, \ldots, K, \text{且} k \ne k_1, k_2. \end{aligned} \tag{10}$$

Formula (1) indicates that the optimization of production scheduling of concrete components is aimed at minimizing costs. The first item is the overtime expenses involved in the concrete pouring and curing process. The second item represents the normal working wages of each team. And the third item represents equipment depreciation expenses. In the equation, mod is the modulus operation symbol, sgn is the sign function, and Int is the integer function. Formula (2) represents the exclusive constraint of processing tasks in the same work team; that is, the same work team can only process one process at the same time. Formula (3) is the process constraint, which means that each process of each component must be assigned to a task force for processing. Formula (4) is the time limit. Formula (5) is the constraint of the processing start time, which means that the start time of work can only be within the normal working period. Formula (6) is the time constraint of the adjacent process; that is, only after the processing task of the previous process is completed, the processing resources can be allocated for this process. Formula (7) is the task assignment constraint; that is, only if the process j of component i is assigned to the task force k for processing, this start time will exist. Formula (8) represents the time limit for the start of the work when the components are placed in the concrete pouring and curing process; that is, if the processing cannot be completed within the normal work and overtime hours, the processing will not be performed on the same day, and the work will be performed on the next day. Formula (9) expresses the restriction on the relationship between the processing end time of the component concrete pouring process (the third process) and the concrete curing process (the fourth process). Formula (10) expresses the work end time limit when the processing can be interrupted; that is, if the processing operation cannot be completed within the normal working time of the day, the same work team will continue processing the next day.

3. Discrete Differential Evolution Algorithm Based on Double Coding

The basic DE is based on real number coding, which is suitable for solving continuous optimization problems. Therefore, for the optimization problem of precast component production scheduling, this paper adopts DDE based on double coding. Through task code and team code, the optimal order and optimal processing resource allocation strategy for each concrete component processing can be determined.

3.1. Encoding and Decoding Scheme. To express the processing sequence of each component process and the corresponding work team at the same time, the two-level coding form is adopted. The first layer is the task code based on components and processes [23], denoted as chromosome 1. The length of the chromosome is the sum of all the processes of all the components, which is defined as Z, and each locus is directly represented by the component number. The second layer is the team code based on the task force distribution sequence, denoted as chromosome 2. Chromosome length is Z. Each gene locus is directly represented by the task force number. After the two codes are determined, the gene combination corresponding to the same position on each pair of chromosomes indicates the processing order and work team allocation plan of a process of a component.

Set chromosome $1 = \{a_1, a_2, \ldots, a_Z\}$; $a_n (n \in 1, \ldots, N)$ corresponds to the currently processed component. In the task chromosome, components of the same species are given the same number, and different positions of the same number represent different processes of the component. The cumulative number of occurrences of the same number is the number of processes required to process the component. Set chromosome $2 = \{b_1, b_2, \ldots, b_Z\}.b_n (n \in 1. \ldots, N)$ corresponds to the available task force number, used to determine the processing work team of each process of each component. The double-layer chromosome encoding is shown in Figure 1:

Define O_{jk} as the k th process of j component. In the case shown in Figure 1, there are two components to be processed, and each component contains three processes, which can be processed by three teams. The task code in the figure is $(1, 2, 1, 2, 2, 1)$, which determines the processing sequence of the two components for a total of 6 processes. The sequence of processes can be expressed as $O_{11} - O_{21} - O_{12} - O_{22} - O_{23} - O_{13}$. The team code is $(1, 3, 2, 3, 1, 2)$, where the first "1" indicates that the process is performed by the task force 1, and similarly, the processing teams of the remaining processes can be obtained.

The chromosome decoding process is the process of converting the code into a feasible scheduling solution. This paper will adopt the activity scheduling method [24]. The decoding steps for chromosome 1 and chromosome 2 are as follows:

Step 1. Decode the team code. Read the code value from left to right and form the work queue sequence matrix W and the operation time sequence matrix T according to the known operation time. $W(j, 1), \ldots, W(j, K)$ means the processing team number of all processes of the j th component, and there is a processing order constraint among them. $T(j, k)$ represents the operating time of the k th process of the j th component. Matrix W corresponds to matrix T one-to-one.

Step 2. Read the code value on chromosome1 from left to right and convert them to the corresponding process O_{jk}. And the processing team $N_i = W(j, k)$ and operation time $E_{ijk} = T(j, k)$ are obtained through the work queue sequence matrix W and the time sequence matrix T.

Step 3. Define each work team's idle start time $TS_i = (e, e, \ldots, e)$ and work idle end time $TE_i = (D, D, \ldots, D)$.

Step 3.1. Determine whether process k is the first process of component j. If it is the first process, get the idle time of the work team that can be inserted according to formula (11), then judge whether this process can be completed within the working hours, and get the earliest start time Ta. If not, perform Step 3.2.

$$TE_i - TS_i \geq E_{ijk}, \tag{11}$$

$$Ta = TS_i. \tag{12}$$

Step 3.2. Determine whether process k is the third or fourth process of component j. If it is, it is necessary to consider the completion time $C_{j,k-1}$ of the previous

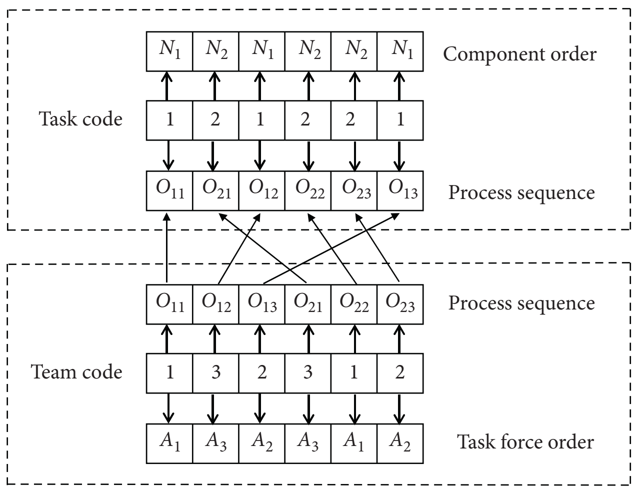
Component order
Task code
Process sequence
Process sequence
Team code
Task force order
N_1 N_2 N_1 N_2 N_2 N_1
1 2 1 2 2 1
O_{11} O_{21} O_{12} O_{22} O_{23} O_{13}
O_{11} O_{12} O_{13} O_{21} O_{22} O_{23}
1 3 2 3 1 2
A_1 A_3 A_2 A_3 A_1 A_2

Figure 1: Schematic diagram of double chromosome coding.

process and the idle start time of the work team at the same time, as shown in formula (13). It is also necessary to judge whether the process can be completed before the end of overtime. If the conditions are met, the process can be inserted into the current free period. If not, Ta starts from the next day and judges whether the interval meets the conditions at this time.

$$TE_i - \max\left\{TS_i, C_{j,k-1}\right\} \geq E_{ijk}. \tag{13}$$

Step 3.3. Determine the difference between the completion time of the previous process and the start time and end time of the work team interval, and then determine whether it is completed within the working time. If it can, the process can be inserted into the time interval that meets the conditions; if not, it is necessary to judge whether the sum of the operation time and the rest time meets the current insertion conditions.

Step 3.4. Determine whether the current chromosome has been read. If it is satisfied, the loop ends; otherwise, proceed to step 3.1.

After decoding, the activity scheduling plan can be obtained; that is, the operation team of each process of each component can be designated, and the operation start time and end time can be obtained.

3.2. Fitness Function. Formula (14) expresses the goal of minimizing production cost; therefore, the fitness function is

$$\begin{aligned}\min z = {} & \sum_{i=1}^{A}\sum_{j=1}^{N}\sum_{k=k_1}^{k_2}\left(f_{ijk}\mathrm{mod}24 - TW - e\right)\cdot CC_i \cdot \mathrm{sgn}\left\{\max\left\{\left(f_{ijk}\mathrm{mod}24 - TW - e\right), 0\right\}\right\} \\ & + \sum_{i=1}^{A}\left\{\mathrm{Int}\left(\frac{f_{ih_is_i}}{24}\right)\cdot TW + \min\left\{\left(\left(\frac{f_{ih_is_i}}{24}\right) - e, TW\right\}\right\}\right\}\cdot CG_i + \sum_{i=1}^{A}\sum_{j=1}^{N}\sum_{k=1}^{K} X_{ijk}\cdot E_{ijk}\cdot CM_k.\end{aligned} \tag{14}$$

3.3. Global Search Operation. An individual in the population is composed of task chromosomes and team chromosomes. Therefore, in the iteration of the population, it is necessary to use the DDE evolution operation on the two chromosomes, respectively.

3.3.1. Improve POX Crossover Method. Traditional DE is based on real number coding in continuous space. For the discrete problem of production scheduling optimization, traditional evolutionary operations are no longer applicable. Therefore, this paper adopts POX (Precedence Order-based Crossover) [25] to recombine each gene on the chromosome and improve the crossover method to adapt to the differential evolution operation of the discrete coding of double coding. The operation process is

Step 1. Randomly divide the component set $N = (N_1, N_2, \ldots, N_n)$ into two complementary nonempty sets, Set 1 and Set 2.

Step 2. Respectively, copy the parental chromosomes $F1$ and $F2$ containing the numbers contained in Set 1 and retain their order and position, thereby generating offspring chromosomes C1 and C2.

Step 3. Copy the component numbers contained in Set 2 and the parent chromosomes $F1$ and $F2$ simultaneously to the offspring chromosomes c2 and c1 and retain their order until all the missing chromosomes are filled in. If there are still vacancies after the complete copy, 0 will be used to occupy the position.

Step 4. Determine in turn whether the code on each gene position of the offspring chromosomes is 0; if it is zero, randomly generate any value between $[0.A]$ for replacement.

Examples of POX are shown in Figures 2 and 3.

The example in Figure 2 contains 4 components, and each component requires two processes to process. Copy the parts containing 2 and 3 in $F1$ and $F2$ to $C1$ and $C2$, respectively, then copy the remaining parts in $F1$ to $C2$ in order, and finally copy the remaining parts in $F2$ to $C1$ in order. Figure 3 is the same, vacant positions are occupied by 0, and redundant genes are discarded.

3.3.2. Mutation Crossover Strategy Design. To adapt to the discretization operation, based on the improved POX crossover method, this paper will adopt the following crossover mutation strategy:

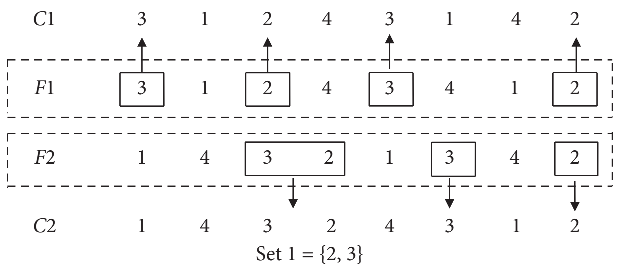

Figure 2: POX crossover example 1.

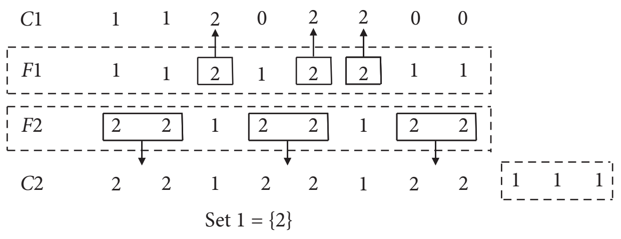

Figure 3: POX crossover example 2.

(1) Mutation operation

$$Y_i(t+1)=\begin{cases} P\left(X_{r1}(t),X_{r2}(t)\right), & \text{rand}<F,\\ X_{r1}(t), & \text{others},\end{cases} \tag{15}$$

$$V_i(t+1)=\begin{cases} P\left(Y_i(t+1),X_{r3}(t)\right), & \text{rand}<F,\\ Y_i(t+1), & \text{others}.\end{cases} \tag{16}$$

In the formula, $r1, r2, r3$ are any mutually different integers from 1 to n, n is the number of populations, $t$is the evolutionary algebra, F is the scaling factor, P represents the POX operation, and rand is the random number of [0, 1]. Equation (15) represents the difference vector formed by two different parent individuals, and equation (16) represents the difference vector formed by the difference vector and the other parent individual.

(2) Cross operation

$$U_i(t+1)=\begin{cases} P\left(V_i(t+1),X_i(t)\right), & \text{rand}<Cr,\\ V_i(t+1), & \text{others}.\end{cases} \tag{17}$$

In the formula, Cr is the crossover probability. Formula (17) represents the temporary experimental individual obtained after the crossover between the mutant individual and the parent individual.

(3) Continuous work penalty mechanism

To make the process of generating feasible solutions more effective, a continuous work penalty mechanism is adopted for the chromosome 2 population in response to a large amount of work undertaken by the same task force. By introducing penalty coefficients α and penalty factors ε, increase the variation and crossover probability of unreasonable feasible solutions. The specific steps are as follows:

Step 1. Judge whether the workload percentage of each work team in the current chromosome is greater than the penalty coefficient; if yes, execute Step 2; otherwise end the operation.

Step 2. Penalize the scaling factor of the chromosome as shown in formula (18).

$$F=F+\varepsilon. \tag{18}$$

3.3.3. Population Retention Strategy. In one iteration, after mutation and crossover operations, under the premise of ensuring the validity of the search information, a one-to-one greedy strategy mechanism is adopted to select the next-generation population individuals to prevent the genetic characteristics of the offspring from becoming uniform. That is, for each parent individual and temporary experimental individual, by comparing the fitness function value $f(x)$, select the better individual and random individual to enter the next-generation population. The selection method is shown in the formula (19).

$$X_i(t+1)=\begin{cases} U_i(t+1), & f\left(U_i(t+1)\right)>f\left(X_i(t)\right),\\ X_i(t), & \text{others}.\end{cases} \tag{19}$$

3.4. Local Search Operation. The DDE algorithm has the characteristics of fast convergence, but it is easy to fall into the local optimum. To solve this problem and further improve the optimization efficiency of the algorithm, a local search operation based on the interchange neighborhood structure [26] is introduced. Perform a local search operation for the best individual and 20% random chromosomes in the chromosome 2 population. The selected individual is defined as chromosome 2∗. The operation is as follows:

Step 0. Generate random integers between [0, 1], denoted as q_1. Then generate random integers between [1, chromosome 2.length], denoted as q_2. Let chromosome = chromosome 2∗; $m = q_2$. If $q_1 = 0$, execute Step 1.1–Step 1.3; otherwise, execute Step 2.1–Step 2.3.

Step 1.1. If $q_2 > 1$, generate a random integer between [1, $q_2 - 1$], which is recorded as q_3. It means searching for q_3 bits from the q_2 bit of the chromosome forward.

Step 1.2. Exchange q_3 position and $(m-1)$ position gene of the chromosome; if f(chromosome 1^*, chromosome) < f(chromosome 1^*, chromosome 2^*), then chromosome 2^* = chromosome.

Step 1.3. $m = m - 1$; judge whether m is greater than $(q_2 - q_3)$; if yes, execute Step 1.2; otherwise end the operation.

Step 2.1. If q_2 < chromosome 2.length, generate random integers between [1, chromosome 2.length-q_2], denoted as q_3. It means searching for q_3 bits from chromosome q_2 backward.

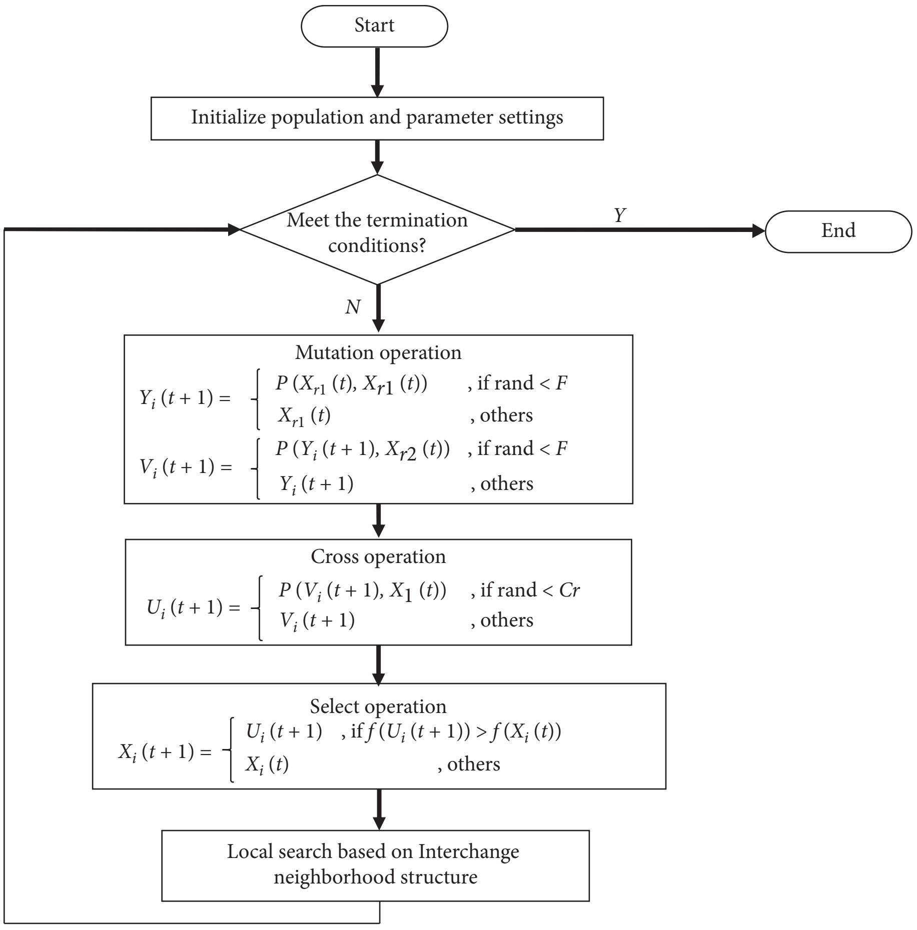
Start
Initialize population and parameter settings
Meet the termination conditions?
Y
End
N
Mutation operation
$Y_i(t+1) = \begin{cases} P(X_{r1}(t), X_{r1}(t)) & \text{, if rand} < F \\ X_{r1}(t) & \text{, others} \end{cases}$
$V_i(t+1) = \begin{cases} P(Y_i(t+1), X_{r2}(t)) & \text{, if rand} < F \\ Y_i(t+1) & \text{, others} \end{cases}$
Cross operation
$U_i(t+1) = \begin{cases} P(V_i(t+1), X_1(t)) & \text{, if rand} < Cr \\ V_i(t+1) & \text{, others} \end{cases}$
Select operation
$X_i(t+1) = \begin{cases} U_i(t+1) & \text{, if } f(U_i(t+1)) > f(X_i(t)) \\ X_i(t) & \text{, others} \end{cases}$
Local search based on Interchange neighborhood structure

Figure 4: Flowchart of discrete differential evolution algorithm.

Step 2.2. Exchange q_3 position and $(m+1)$ position gene of the chromosome; if f(chromosome 1^*, chromosome) < f(chromosome 1^*, chromosome 2^*), then chromosome 2^* = chromosome.

Step 2.3. $m = m + 1$; judge whether m is less than $(q_2 + q_3)$; if yes, execute Step 2.2; otherwise, end the operation.

3.5. Algorithm Flow. The process of the improved DDE algorithm in this paper is shown in Figure 4.

4. Case Analysis

Use Visual Studio 2017 to program the discrete difference algorithm based on C# language to simulate the production scheduling optimization problem of precast concrete components. Its scale is 5 components and 5 work teams, each component contains 5 processes, and each work team can operate 5 processes. The operation time of each process is different due to the difference in the component type and the work team. The specific time is shown in Table 1. The wages of each task force are shown in Table 2. Depreciation expenses of equipment unit working hours are shown in Table 3. The normal working hours of the task force are 9 hours, and the maximum overtime hours are 4 hours. The daily starting time is 8 am.

After experiments, the optimization results are better when the parameters are set as evolutionary algebra $G = 1000$, population size $n = 50$, mutation factor $F = 0.9$, and cross factor $Cr = 0.35$. The results of the component process assignment are shown in Table 4, and the component production scheduling Gantt chart is shown in Figure 5. The minimum production cost is 2626.7 yuan.

To analyze the effectiveness of the improved discrete difference algorithm based on the local search strategy (LSS-DDE), compare it with the solution results of the traditional discrete difference algorithm (TRAD-DDE), ant colony algorithm (ACO), and particle swarm algorithm (PSO). Each algorithm performs 20 independent simulation experiments for the above component production scheduling examples, and the termination conditions are all 1000 iterations. The algorithm parameter setting table is shown in Table 5. The comparison of running results is shown in Table 6. The convergence situation comparison is shown in Figure 6:

Table 1: The operation time of different processes of processing each component of each work team.

Component	Process	Optional task force	Processing time/h
Column	*k*1	*A*1, *A*2, *A*3, *A*4, *A*5	0.6, 0.8, 0.9, 1, 0.7
	*k*2	*A*1, *A*2, *A*3, *A*4, *A*5	3.3, 2.7, 3.5, 4.5, 3.3
	*k*3	*A*1, *A*2, *A*3, *A*4, *A*5	1.8, 1.8, 1.6, 2.3, 1.8
	*k*4	*A*1, *A*2, *A*3, *A*4, *A*5	5.5, 5.5, 5.5, 5.5, 5.5
	*k*5	*A*1, *A*2, *A*3, *A*4, *A*5	0.5, 0.6, 0.6, 0.7, 0.3
Beam	*k*1	*A*1, *A*2, *A*3, *A*4, *A*5	0.5, 0.6, 0.8, 0.9, 0.6
	*k*2	*A*1, *A*2, *A*3, *A*4, *A*5	3.6, 3, 3.7, 4.8, 3.7
	*k*3	*A*1, *A*2, *A*3, *A*4, *A*5	1.6, 1.6, 1.4, 2.1, 1.6
	*k*4	*A*1, *A*2, *A*3, *A*4, *A*5	5.5, 5.5, 5.5, 5.5, 5.5
	*k*5	*A*1, *A*2, *A*3, *A*4, *A*5	0.4, 0.5, 0.5, 0.6, 0.3
Board	*k*1	*A*1, *A*2, *A*3, *A*4, *A*5	0.3, 0.5, 0.6, 0.6, 0.4
	*k*2	*A*1, *A*2, *A*3, *A*4, *A*5	3.3, 2.8, 3.6, 4.7, 3.5
	*k*3	*A*1, *A*2, *A*3, *A*4, *A*5	2.5, 2.5, 2, 3, 2.6
	*k*4	*A*1, *A*2, *A*3, *A*4, *A*5	5.5, 5.5, 5.5, 5.5, 5.5
	*k*5	*A*1, *A*2, *A*3, *A*4, *A*5	0.3, 0.4, 0.4, 0.6, 0.2
Stairs	*k*1	*A*1, *A*2, *A*3, *A*4, *A*5	0.9, 1.1, 1.2, 1.4, 1
	*k*2	*A*1, *A*2, *A*3, *A*4, *A*5	4, 3.5, 4.2, 6, 4.2
	*k*3	*A*1, *A*2, *A*3, *A*4, *A*5	1.4, 1.4, 1.3, 1.7, 1.5
	*k*4	*A*1, *A*2, *A*3, *A*4, *A*5	5.5, 5.5, 5.5, 5.5, 5.5
	*k*5	*A*1, *A*2, *A*3, *A*4, *A*5	0.7, 0.9, 0.9, 1, 0.5
Shear wall	*k*1	*A*1, *A*2, *A*3, *A*4, *A*5	0.3, 0.5, 0.6, 0.6, 0.4
	*k*2	*A*1, *A*2, *A*3, *A*4, *A*5	3.6, 3, 3.7, 4.8, 3.7
	*k*3	*A*1, *A*2, *A*3, *A*4, *A*5	2.5, 2.5, 2, 3, 2.6
	*k*4	*A*1, *A*2, *A*3, *A*4, *A*5	5.5, 5.5, 5.5, 5.5, 5.5
	*k*5	*A*1, *A*2, *A*3, *A*4, *A*5	0.3, 0.4, 0.4, 0.6, 0.2

Table 2: Wages of each task force.

Task force	*A*1	*A*2	*A*3	*A*4	*A*5
Average salary (h)	21	22.5	18.5	17.5	21
Overtime pay (h)	35	35	35	35	35

Table 3: Depreciation expenses of equipment unit working hours.

Processing machinery	*K*1	*K*2	*K*3	*K*4	*K*5
Depreciation expense (h)	20	55	30	50	15

Table 4: Component process assignment results.

Process component	*k*1	*k*2	*k*3	*k*4	*k*5
Column	*A*3	*A*2	*A*3	*A*4	*A*5
Beam	*A*1	*A*2	*A*2	*A*4	*A*4
Board	*A*4	*A*2	*A*3	*A*4	*A*2
Stairs	*A*5	*A*2	*A*3	*A*5	*A*5
Shear wall	*A*1	*A*1	*A*3	*A*4	*A*3

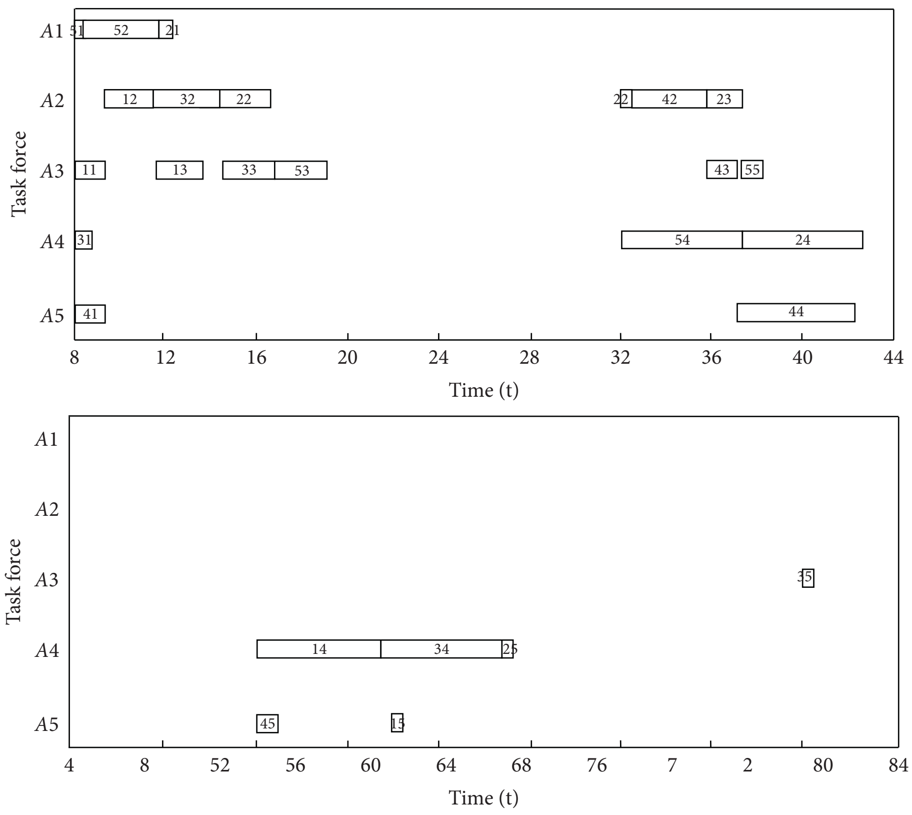

Figure 5: Gantt diagram of component production scheduling.

Table 5: Algorithm parameter setting table.

Algorithm	The main parameters
LSS-DDE	$G = 100,\ n = 50,\ F = 0.9,\ Cr = 0.35$
TRAD-DDE	$G = 100,\ n = 50,\ F = 0.9,\ Cr = 0.35$
ACO	$G = 100,\ n = 50,\ \alpha = 1.5,\ \beta = 4.2,\ \rho = 0.5,\ Q = 200$
PSO	$G = 100,\ n = 50,\ \omega_{\max} = 0.9,\ \omega_{\min} = 0.4,\ \phi_1 = \phi_2 = 1.49618$

Table 6: Comparison of running results.

Algorithm	Optimal minimum cost	Worst minimum cost	Average minimum cost	Standard deviation
LSS-DDE	2626.7	2636.2	2633.6	2.995
TRAD-DDE	2682.4	2714.9	2700.5	9.849
ACO	2869.5	2908.1	2892.8	13.208
PSO	2852.3	2881.5	2868.2	10.230

It can be seen from the comparison of the results that the application of a discrete differential evolution algorithm can provide better optimization results. Also, it can be seen from the convergence that LSS-DDE can get a better global optimal solution. Due to the local search operation, LSS-DDE jumps out of the local optimum, thus avoiding premature maturity.

To verify the reliability of the comparison test, Wilcoxon signed-rank test was used to test the nonrandomness of the difference in the results of the algorithm. Use SPSS to analyze experimental results, $\alpha = 0.01$, and the test results are shown in Figures 7–9.

It can be seen from the test results that the P-value is less than 0.001. That is, compared with other algorithms, LSS-DDE shows a significant improvement over TRAD-DDE, ACO, and PSO, with a level of significance $\alpha = 0.01$.

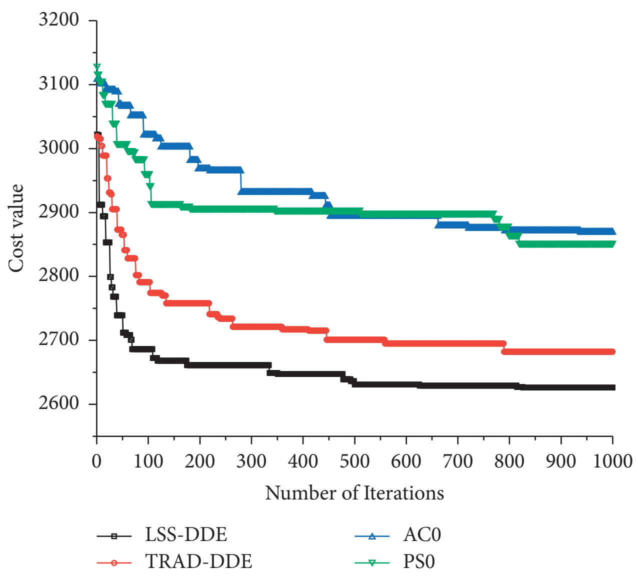

FIGURE 6: Comparison of simulation convergence.

Test statistics[b]

	TRADDDE-LSSDDE
Z	−3.922[a]
Asymp.Sig. (2-tailed)	.000

a. Based on negative rands.
b. Wilcoxon signed ranks test

FIGURE 7: Wilcoxon signed ranks test results 1.

Test statistics[b]

	ACO-LSSDDE
Z	−3.921[a]
Asymp.Sig. (2-tailed)	.000

a. Based on negative rands.
b. Wilcoxon signed ranks test

FIGURE 8: Wilcoxon signed ranks test results 2.

Test statistics[b]

	PSO-LSSDDE
Z	−3.921[a]
Asymp.Sig. (2-tailed)	.000

a. Based on negative rands.
b. Wilcoxon signed ranks test

FIGURE 9: Wilcoxon signed ranks test results 3.

5. Conclusion

As a new type of construction method, prefabricated buildings have distinctive features such as high production efficiency, high industrialization level, high standardization, strong environmental friendliness, and low external climate impact. Building industrialization is an important development direction in the future construction field. However, the high production cost of fabricated building components is the main reason that affects its promotion and application.

The author reasonably distinguished the costs involved in component production and established an optimization model for assembly-type component production scheduling to minimize component production costs. This model optimizes component production scheduling under the premise of fully considering the constraints of component production process requirements, process time constraints, and production equipment resource constraints. This paper uses a discrete differential evolution algorithm to solve the optimization model. Double-layer chromosome encoding and activity scheduling decoding are used for global search, and the local search process is based on the interchange neighborhood structure.

In the selection process, a continuous work penalty mechanism is designed to improve the performance of the algorithm. Through the optimization model and algorithm in the application simulation of a concrete component production plant, the validity and scientificity are verified. The author provides an efficient and accurate optimization method for reducing the production cost of prefabricated building components, which adapts to the standardized and streamlined production characteristics of prefabricated buildings. This method provides technical support for reducing the overall cost of prefabricated buildings. More detailed constraints are integrated into the optimization model of assembly-type component production scheduling, and the in-depth study of applying the multiangle improved DDE algorithm to the solution of the optimization model is one of the important future research directions in this field.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This project was funded by the National Natural Science Foundation of China (no. 51678375) and Liaoning Provincial Natural Fund Guidance Plan (no. 2019-ZD-0683).

References

[1] K. N. Liu, S. J. Zhang, and Y. K. Su, "Summary of research in the field of prefabricated building management," *Journal of Civil Engineering and Management*, vol. 35, no. 6, pp. 163–170, 2018.

[2] T. Yashiro, "Conceptual framework of the evolution and transformation of the idea of the industrialization of building in Japan," *Construction Management and Economics*, vol. 32, no. 12, pp. 16–39, 2018.

[3] Z. Q. Wang, Q. M. Zhang, and W. B. You, "Evolutionary game and simulation research on government incentive strategies for prefabricated buildings—based on the perspective of government subsidies," *System Engineering*, vol. 37, no. 3, pp. 151–158, 2019.

[4] G. F. Qiu, M. M. Zhao, and X. Hu, "Evaluation of prefabricated building component carrier selection based on rough set-grey situation decision," *Journal of Civil Engineering and Management*, vol. 36, no. 3, pp. 48–53, 2019.

[5] D. Z. Li and Y. Wang, "Driving and hindering factors for construction units to actively adopt prefabricated buildings," *Journal of Civil Engineering and Management*, vol. 36, no. 3, pp. 7–11, 2019.

[6] H. M. Gou, C. Mao, and Q. Y. Dong, "Comparative study on the construction cost of prefabricated and cast-in-place buildings," *Building Economy*, vol. 39, no. 3, pp. 71–74, 2018.

[7] C.-H. Ko and S.-F. Wang, "GA-based decision support systems for precast production planning," *Automation in Construction*, vol. 19, no. 7, pp. 907–916, 2010.

[8] J. G. Jeong, M. Hastak, M. Syal, and T. Hong, "Internal relationship modeling and production planning optimization for the manufactured housing," *Automation in Construction*, vol. 20, no. 7, pp. 864–873, 2011.

[9] C. Rausch and M. Nahangi, "Optimum assembly planning for modular construction components," *Journal of Computing in Civil Engineering*, vol. 31, no. 1, pp. 1–14, 2016.

[10] W. Ali, K. Ali, and H. Anas, "Gaining-sharing knowledge based algorithm for solving optimization problems algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 1501–1529, 2020.

[11] P. Agrawal, T. Mohamed, and A. Wagdy, "A novel binary gaining-sharing knowledge-based optimization algorithm for feature selection," *Neural Computing and Applications*, pp. 1–20, 2020.

[12] X. Q. Chen, D. W. Hu, Q. Q. Yang et al., "Improved ant colony algorithm for multi-objective delivery vehicle routing problem," *Control Theory and Applications*, vol. 35, no. 9, pp. 1347–1356, 2018.

[13] W. Zhang and W. M. Huang, "Multi-strategy adaptive multi-objective particle swarm algorithm based on population partition," *Acta Automatica Sinica*, pp. 1–14, 2020, http://kns.cnki.net/kcms/detail/11.2109.TP.20200915.0941.002. Html.

[14] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 842–844, IEEE, Nayoya, Japan, May 1996.

[15] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution-an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[16] W. Du, L. Tong, and Y. Tang, "Metaheuristic optimization-based identification of fractional-order systems under stable distribution noises," *Physics Letters A*, vol. 382, no. 34, pp. 2313–2320, 2018.

[17] X. Nian, Z. Wang, and F. Qian, "A hybrid algorithm based on differential evolution and group search optimization and its application on ethylene cracking furnace," *Chinese Journal of Chemical Engineering*, vol. 21, no. 5, pp. 537–543, 2013.

[18] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[19] K. Wang and W. Z. Gong, "Improved differential evolution algorithm for solving nonlinear equation systems," *Control and Decision*, vol. 35, no. 9, pp. 2121–2128, 2020.

[20] Q. Xiang, Y. Y. Zhou, and Z. J. Lu, "Improved differential evolution parameter control and two-way scheduling algorithm for resource constraint projects," *Acta Automatica Sinica*, vol. 46, no. 2, pp. 283–293, 2020.

[21] A. Wagdy, A. Hadi, and K. Jambi, "Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global Numerical Optimization," *Swarm and Evolutionary Computation*, vol. 50, Article ID 100455, 2019.

[22] L. Cheng, Y. Wang, C. Mohamed, and T. Xiao, "Adaptive differential evolution based on successful experience information," *IEEE Access*, vol. 8, pp. 164611–164636, 2020.

[23] M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan, and Y. C. Liang, "A discrete differential evolution algorithm for the no-wait flow shop scheduling problem with total flowtime criterion," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling*, pp. 251–258, IEEE, Honolulu, HI, USA, April 2007.

[24] M. Pinedo and K. Hadavi, "Scheduling theory, algorithms, and system," *Operations Research Proceedings*, vol. 1991, pp. 35–42, 2002.

[25] I. Kacem, "Genetic algorithm for the flexible job-shop scheduling problem," in *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA, 2003.

[26] H. Y. Wang, Y. W. Zhao, W. L. Wang, and X. L. Xu, "Two-stage differential evolution algorithm for multi-resource job shop batch scheduling problem," *Control and Decision*, vol. 25, no. 11, pp. 1635–1644, 2010.