Hindawi
Scientific Programming
Volume 2022, Article ID 5622466, 16 pages
https://doi.org/10.1155/2022/5622466

Research Article

A Two-Population Cooperative Multiobjective Differential Evolution Algorithm for Batching Scheduling Problem

Cunli Song [1,2]

[1]*College of Software, Dalian Jiaotong University, Liaoning, Dalian 116052, China*
[2]*Artificial Intelligence Key Laboratory of Sichuan Province, Sichuan, Zigong 643000, China*

Correspondence should be addressed to Cunli Song; scunli@163.com

Received 22 November 2021; Revised 19 January 2022; Accepted 7 February 2022; Published 7 March 2022

Academic Editor: Cristian Mateos

Batch processing machine (BPM) scheduling problem is a NP hard problem for it includes machine allocation, job grouping, and batch scheduling. In this paper, to address the BPM scheduling problem with unrelated parallel machine, a multiobjective algorithm based on multipopulation coevolution is proposed to minimize the total energy consumption and the completion time simultaneously. Firstly, the mixed integer programming model of the problem is established, and four heuristic decoding rules are proposed. Secondly, to improve the diversity and convergence of the algorithm, the population is divided into two populations: each of the populations evolves independently by using different decoding rules, and the two populations will communicate through a common external archive set every certain number of generations. Thirdly, an initialization strategy and a variable neighborhood search algorithm (VNS) are proposed to improve the overall performance of the algorithm. Finally, in order to evaluate the proposed algorithm, a large number of comparative experiments with the state-of-the-art multiobjective algorithms are carried out, and the experimental results proved the effectiveness of the proposed algorithm.

1. Introduction

Manufacturing industry reflects a country's productivity level, and production scheduling is a key link to determine enterprises to achieve their production objectives. Therefore, the research on production scheduling has always been a hot issue for enterprises and scholars. With the intensification of competition, process renewal, resource shortage, and environmental pollution, new research hotspots have emerged in production scheduling. (1) Research on energy efficiency scheduling problem, such as Zhao et al. [1], addressed the energy-efficient scheduling problem of the no-wait flow-shop; Song [2] addressed the energy-efficient scheduling problem of the hybrid flow-shop (HFS) with unrelated parallel machine, etc. (2) Combined with the production practice, the research on the complex multiconstraint production scheduling problem, such as Zhao et al. [3], addressed the scheduling problem of distributed no-idle flow-shop in heterogeneous factory system; Zhao et al. [4] addressed the scheduling problem of the distributed blocking flow shop; Liu et al. [5] addressed the charging scheduling of electric vehicles with more requirements, etc. (3) Research on multiobjective production scheduling problem, such as Zhou et al. [6], addressed a BPM scheduling problem with the aim to minimize the makespan and the total electricity cost; Zhu-Min [7] addressed the scheduling problem of HFS with the aim to minimize delay penalties, proceeding time, setup cost, and holding cost; Lu et al. [8] addressed the scheduling problem of HFS to minimize the noise pollution, makespan and energy consumption simultaneously, etc. Therefore, it is of great significance to study the multiobjective energy-efficient scheduling problem under complex production conditions.

In manufacturing shops, batch processing machine scheduling problem (BPMSP) is a NP hard problem; it has a wide range of applications, such as foundry industry [9], logistics freight [10], and electronics manufacturing facilities [11]. The character of BPMSP is that it takes a batch as a unit to process multiple jobs at the same time, and the processing time of a batch is determined by the job with the longest

processing time, while the earliest start processing time is determined by the job that arrives the latest. The unrelated parallel BPM scheduling problem (UPBPMSP) is an extension of the traditional BPMSP and exists widely in reality. Owing to the fact that there is more than one BPM with different capacity, processing efficiency, and processing speed, UPBPMSP is more complex than BPMSP. In UPBPMSP, for each job, we should decide the processing machine, with which jobs to form a batch and the process sequence of the batches. All these affect the production efficiency and the total processing energy consumption of enterprises at the same time. Therefore, the research on this problem is of great significance. In the past decades, the BPMSP has been studied extensively. To minimize the makespan, Zarook [12], zhou et al. [13], and Xin et al. [14] addressed a single machine BPMSP, Muter et al. [15] addressed an identical parallel BMPSP, and Li et al. [16] addressed the UPBMPSP. To minimize the total weighted tardiness, Chou et al. [17] addressed a parallel BMPSP, etc.

Obviously, the studies for energy efficiency UPBPMSP are less. In view of this, we studied the UPBPMSP with the aim to minimize the energy consumption and the maximum completion time (makespan) simultaneously, as they are the main goals an enterprise pursues. At the same time, more constraints are considered here, such as the different machine capacity and power, the different job arrival time, size, processing time, and the special process requirements for special job family cluster. Compared with the existing research, the main contributions of this paper can be summarized as follows:

(1) Aiming at minimizing the total energy consumption (TEC) and the makespan, a multiobjective differential evolution algorithm based on multipopulation coevolution (MODE/MPC) is proposed

(2) To ensure the distribution and diversity of solutions, the algorithm adopts two-population coevolution: each population adopts different coding and decoding methods to search for excellent solutions from different perspectives. Every several generations, the frontier solutions of the two populations fuse to form the frontier solutions of the whole society and are replaced by the frontier solution set of the whole society to guide the population's evolution.

(3) To improve the initial population quality, three initialization strategies are proposed to initialize the population during the initialization stage.

(4) To speed up the convergence and improve the local search ability of the algorithm, a greedy variable neighborhood search algorithm (VNS) is proposed and performed on the frontier solutions archive set of each population.

The rest of this paper is organized as follows: Section 2 introduces the related work of the batching scheduling problems and differential evolution (DE) optimization algorithms. Section 3 presents the studied problem. Section 4 describes the proposed MODE/MPC algorithm in detail. Experimental results and comparison are provided in Section 5. The conclusions of this study and future research directions are presented in Section 6.

2. Literature Review

2.1. BPM Scheduling Problem. In 1980, Lerner et al. [18] firstly studied the single machine BPMSP for semiconductor production. Since then, many scholars have studied BPMSP. To minimize the makespan of a single machine BPMSP, Zarook et al. [12] and Huang et al. [19] proposed a heuristic algorithm, respectively, with the consideration of machine maintenance; Kashan et al. [20] proposed an effective hybrid genetic algorithm (HGA) with consideration of nonidentical job sizes; Guo [21] proposed a variable neighborhood based memetic algorithm (VNMA). To minimize the maximum lateness of the single machine BPMSP, Zhou et al. [13] presented a modified particle swarm optimization (MPSO) algorithm with the consideration of nonidentical job sizes and release dates. To minimize flow time of the single machine BPMSP, Parsa et al. [22] proposed a hybrid max–min ant system with consideration nonidentical job size; Ghazvini and Dupont [23] proposed various new heuristics. For the identical parallel BPMSP, Jia et al. [24] proposed an ant colony optimization algorithm with the criterion of minimizing the total weighted completion time; Jia et al. [25] Integrated production and transportation and proposed a deterministic heuristic and two hybrid metaheuristic algorithms based on ant colony optimization with the criterion of minimizing the total weighted delivery time of jobs. To minimize makespan of the UPBPMSP with different machine capacity, Jia et al. [26] proposed a fuzzy ant colony optimization (FACO) algorithm with the consideration of the jobs' nonidentical sizes and fuzzy processing times.

For multiobjective BPMSP, to address a single machine BPMSP, Kashan et al. [20] proposed two different multiobjective genetic algorithms considering the constraints of nonidentical job sizes to achieve the goals of minimum makespan and maximum tardiness; Zhou [27] proposed a hybrid multiobjective metaheuristic algorithm to minimize the makespan and total electricity cost under the time-of-use (TOU) pricing policy. For the identical parallel machine BPMSP with the constraints of dynamic job arrivals and a time-of-use pricing scheme, Zhou et al. [5] proposed a DE algorithm to minimize makespan and total electricity cost. To minimize the tardiness cost and maximize the utilization rate of dyeing vats, Zhang et al. [28] proposed a multiobjective artificial bee colony (ABC) algorithm in fabric dyeing processes. For the UPBPMSP, Shahvari and Logendran [29] developed an efficient metaheuristic algorithm based on tabu search with multilevel diversification to minimize production cost and maximize customer satisfaction; Qian et al. [30] proposed a multiobjective evolutionary algorithm based on adaptive clustering with the criterion of minimizing the makespan and total electricity cost.

It can be seen from the above review that, for BPMSP, more research focuses on the single machine BPMSP with the criterion of makespan. For parallel BPMSP, most

research just considered the identical parallel or parallel machine with different capacity. As for energy efficient BPMSP, the research is little, and most of the them focus on minimizing total electricity cost under the time-of-use (TOU) pricing policy. However, the UPBPMSP with more constraints is studied less, and it exits widely in reality especially for steel industry. Therefore, the research on the UPBPMSP with more constraints is of great significance.

2.2. DE Algorithm about Production Scheduling. In 1997, DE algorithm was proposed by Storn and Price [31]. Like GA and PSO algorithms, DE has strong optimization ability for combinational optimization problems; therefore, many scholars use it to solve production scheduling problems. In [5], Zhou et al. proposed a discrete DE algorithm to address the multiobjective parallel BPMSP. In [4], Zhao et al. proposed a discrete algorithm to address the blocking flow shop scheduling problem. In [32], Shengyao et al. proposed a self-adaptive DE algorithm for a single machine BPMSP, in which, based on the historical performances, mutation operators are adaptively chosen, and control parameter values are adaptively determined. In [33], Liang et al. proposed a self-adaptive DE algorithm to solve multi-objective flow shop scheduling problems with limited buffers, in which the parameters are adjusted adaptively, and various local searches are used to improve the convergence. In [34], Zhang et al. proposed a hybrid multiobjective DE (HMOEA/DE) to solve the flow shop scheduling problems (FSPs); this algorithm designed a global search strategy and two different mutation strategies for elite individuals to improve the convergence and distribution of solutions, etc.

In summary, there are many applications of DE algorithm in production scheduling problem, and the design of DE is mainly from the following aspects: (1) in view of the discretization of production scheduling problem, many scholars have designed and studied the discrete DE algorithm. (2) From the perspective of parameter control, adaptive DE algorithm is studied. (3) In order to improve the performance of the algorithm, the local search strategy is mixed with the DE algorithm or design new mutation operator and crossover operator. The above improvement measures do not make full use of the problem characteristics to design DE algorithm, and there are almost no multi-population coevolution algorithms with different coding mechanisms. Therefore, it is of great significance to study the multiobjective differential evolution algorithm with two-population coevolution.

3. Problem Formulation

3.1. Problem Description. There is a set of n jobs, $J = \{J_1, J_2, \ldots, J_n\}$, to be processed on m UPBPM, $M = \{M_1, M_1, \ldots, M_m\}$. For each job $J_j \in J$, the size is s_j, the arrive time is r_j, the processing times on m different machines are $p_{j,k}$ ($k = 1, \ldots, m$), and there are a few jobs that can only be processed on a specific machine with a specific power. Each machine $M_k \in M$ has a different capacity c_k and different power l_k (note, where machines are sorted nondecreasing by capacity, and the last machine can process the special job family cluster with a special power). The machines can process the jobs in batches as long as the total size of all the jobs in the batch do not exceed its capacity c_k. Our scheduling tasks include the following: (1) assign jobs to different batch processing machines. (2) Group the jobs on the machine under the constraints of machine capacity. (3) Determine the processing sequence and start time of each batch on each machine. The objective is to minimize TEC and makespan ($C_{\max}$) simultaneously. In order to better describe the problem, we make the following assumptions:

(1) One machine can only process one batch at the same time. Once a batch starts processing, it is not allowed to be interrupted.

(2) Once a batch is processed, you cannot add a job to the batch or remove a job from the batch until the batch is finished.

(3) The total sizes of all jobs in a batch are not allowed to exceed the capacity of the corresponding machine.

(4) The start processing time of a batch should be greater than or equal to the last arrival job in the batch, and the processing time of a batch is determined by the job with the longest processing time in the batch.

3.2. Notations. In order to analyze the problem easily, some notations are defined as follows:

Indexes:

j index of jobs and $j = 1, 2, \ldots, n$.
k index of machines and $k = 1, 2, \ldots, m$.
i index of batches on one machine, and $0 \le i \le n$.

Parameters:

n total number of jobs.
m total number of the machines
s_j The size of job J_j
r_j The arrive time of job J_j
$p_{j,k}$ The processing time of job J_j on machine M_k
c_k The capacity of machine M_k
l_k The power of machine M_k

Decision variables:

$B_{k,i}$ The ith batch of machine M_k
N_k The total number of batches of machine M_k
$x_{j,k,i}$ Equal to 1 if job J_j is grouped to batch $B_{k,i}$, otherwise, $x_{j,k,i} = 0$
$BN_{k,i}$ The total number of jobs in the ith batch of machine M_k
$ST_{k,i}$ The start processing time of batch $B_{k,i}$
$CT_{k,i}$ The complete time of batch $B_{k,i}$
$PT_{k,i}$ The processing time of batch $B_{k,i}$
$WE_{k,i}$ The waste energy of batch $B_{k,i}$
TEC Total energy consumption
$C_{\max}$ The maximum complete time
π A scheduling

3.3. Mixed-Integer Linear Programming Model (MILP).

$$C\max = \min\{C\max(\pi^*)|\pi^* \in \pi\}, \tag{1}$$

$$\text{Energy} = \min\{\text{energy}(\pi^*)|\pi^* \in \pi\}. \tag{2}$$

Subject to:

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

$$\sum_{j=1}^{n}\left(x_{j,k,i}\cdot s_j\right) \le c_k, \quad i = 1,2,\ldots,N_k, \tag{4}$$

$$PT_{k,i} = \max\left\{x_{j,k,i}\cdot p_{j,k}|i = 1,2,\ldots,N_k\right\}, \tag{5}$$

$$ST_{k,i} \ge x_{j,k,i}\cdot r_j, \quad i = 1,2,\ldots,N_k, \tag{6}$$

$$ST_{k,i+1} \ge CT_{k,i}, \quad i = 1,2,\ldots,N_k - 1, \tag{7}$$

$$CT_{k,i} = ST_{k,i} + PT_{k,i}, i = 1,2,\ldots,N_k, \tag{8}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{N_k} BN_{k,i} = n, \tag{9}$$

$$C\max = \max\left\{CT_{k,i}|k = 1,2,\ldots,m, i = 1,2,\ldots,N_k\right\}, \tag{10}$$

$$\text{TEC} = \sum_{k=1}^{m}\sum_{i=1}^{N_k} PT_{k,i}\cdot l_k, \tag{11}$$

$$WE_{k,i} = l_k\cdot\left(\sum_{J_j\in B_{k,i}}\left(P_{k,i} - p_j\right)\cdot s_j + P_{k,i}\cdot\left(c_k - \sum_{J_j\in B_{k,i}} s_j\right)\right). \tag{12}$$

Equation (1) is the objective function of makespan. Equation (2) is the objective function of TEC. Equation (3) ensures that each job is assigned to exactly one batch. Equation (4) guarantees that the total size of jobs in a batch does not exceed the machine capacity. Equation (5) determines that the processing time of a batch is determined by the job with the largest processing time in the batch. Equations (6) and (7) limit that the start processing time of a batch is greater than or equal to the arrival time of jobs in a batch and the completion time of the previous batch on that machine, respectively. Equation (8) shows that the completion time of a batch is equal to the start processing time of the batch plus its processing time; that is, the processing of a batch is not allowed to be interrupted. Equation (9) ensures that the number of jobs of all batches is equal to n. Equation (10) specifies that the makespan (Cmax) is the largest complete time of all batches. Equation (11) specifies that TEC is calculated by summing up all batches' energy consumption, and the energy consumption of a batch is calculated by multiplying the processing time and the power the machine of that batch. Equation (12) is the waste energy of batch $B_{k,i}$.

4. Description of MODE/MPC Algorithm

The traditional DE adopts a float-point encoding scheme and utilizes the differentiation information among the population to find the global optimal solution in a continuous space. DE starts with a population of randomly generated individuals. At each generation, mutation and crossover operators are employed to produce offspring, and then a selection operator is used to determine whether the offspring or the parent can survive at the next generation. In this section, the details of the MODE/MPC algorithm are described in the following sections.

4.1. Representation of Solution. It is an important issue to decide how to represent a solution when designing an effective algorithm for UPBPMSP. As described earlier in this article, the solution of the problem should include two parts, i.e., the jobs' grouping order and the processing machine assigned to the jobs. Therefore, the solution of UPBPMSP is expressed by an integer vector with $2 \times n$ elements. The first n integers in vector correspond to the index of the jobs, and they represent the order of the batches the jobs are grouped into. The last n integers in vector correspond to the first n integers one by one and represent the processing machine of each job. Compared with the most job permutation code [21, 23], this method provides a more comprehensive search space and can support a variety of coding strategies proposed here. At the same time, the expression of the solution improves the probability of finding Pareto frontier solution. To illustrate the solution representation, consider an instance with 10 jobs and 2 BPMs, and Figure 1 shows an example.

4.2. Decoding. For production scheduling problem, coding often cannot fully express the scheduling results; it needs decoding algorithm to further determine the scheduling results. Better decoding methods can easily get better scheduling results and accelerate algorithm convergence. Based on this, we proposed four objective-oriented decoding rules.

Rule-1: Principle of minimizing energy waste: firstly, allocate the jobs to the machines according to the solution as shown in Figure 1; then, group the jobs of each machine in turn according to the principle of minimizing energy waste. That is to say, if there are multiple batches on the current machine that meet the insertion requirements of a job, then insert the job into the batch with the minimal energy waste. Equation (12) can be used to calculate the energy waste for a specific batch. If the job cannot be inserted into the existing batch of the machine, create a new batch on the machine and insert the job into the new batch.

Rule-2: Principle of earliest processing: firstly, allocate the jobs to the machines according to the solution, and group the jobs of each machine in turn according to the

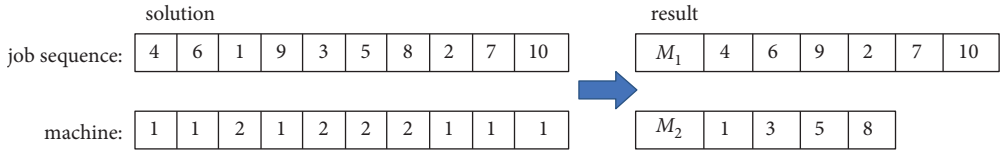

Figure 1: A solution for 10 jobs processed on 2 BPMs.

earliest processing principle. That is to say, the jobs of a machine will be inserted into the existing batch of that machine in turn according to the principle of earliest processing. If there are no such batches on the machine fit for inserting, create a new batch on the machine and insert the job into the new batch.

Rule-3: Principle of minimizing processing time increment (note, this decoding method just uses the first n element of the solution to decode), which is a variant of rule H3 [35], that is, according to the group order of the jobs in the solution, the batch of each machine is operated as follows: (1) assign the job to the batch with the smallest production time increment and calculate its production time increment. (2) A new batch is generated, and its production time increment is calculated. Repeat m times, record the machine with the smallest increment of production time and its batch, and insert the job to the batch of that machine.

Rule-4: Principle of minimizing energy consumption priority (note, this decoding method also just uses the first n elements of the solution to decode), that is, according to the group order of the jobs in the solution, the batch of each machine is operated as follows: (1) assign the job to the batch with the smallest energy consumption increment. (2) Create a new batch on the machine, insert the job to this batch and calculate the energy consumption increment. Repeat m times, record the machine with the smallest energy consumption increment and its batch, and insert the job to the batch of that machine.

Obviously, among the above decoding rules, Rule-1 and Rule-4 consider energy saving, while Rule-2 and Rule-3 consider the completion time. Rule-3 and Rule-4 ignore the last n components of the solution. However, in order to ensure the normal communication of the two population in this paper, during the decoding process, it is necessary to record the machine allocated for the job when using the decoding Rule-3 and Rule-4.

4.3. Scheduling Batches on Machine. Decoding determines the batches of each machine and the process sequence of each bath. The processing time of each batch is determined by the job with the longest processing time in that batch, and the power per unit time of the machine is fixed, so the TEC is completely determined after decoding. However, on the same machine, there exist batches that arrive later but are processed earlier, which will delay the completion time of that machine. Therefore, from the perspective of minimizing makespan, the batches on the same machine are nondecreasing, sorted according to the arrival time and rescheduled according to this order. The pseudocode is shown in Algorithm 1.

4.4. Initializing Population. A better initial population will speed up the algorithm convergence. If there is an initial solution that minimizes one of the objectives in the population, the solution will tend to the optimal solution at a faster speed during evolution. Based on this idea, we propose three methods to initialize the solution according to the characteristics of the problem in this paper.

(1) Arrive first, group first (AFGF): firstly, sort the jobs in nondescending order according to their arrival time, and we get the jobs permutation. Secondly, decode the job permutation by using decoding Rule-3 and Rule-4, respectively, and record the machine assigned to each job. Then, we get two solutions. This method considers the principle of first come, first group batch and process, so it can start processing as soon as possible.

(2) Large size priority batch (LSPB): firstly, sort the jobs in nonascending order according to their size, and we get the job permutation. Secondly, decode the job permutation using decoding Rule-3 and Rule-4, respectively, and record the machine assigned to each job. Then, we get two solutions. This method gives priority to the large-size jobs into batches, and the subsequent small-size jobs can be inserted into the remaining space of the existing batch, so as to maximize the space utility of each batch and reduce waste.

(3) Random generate (RG): first, randomly sort the n jobs to get the job permutation. Second, for each job, randomly select a machine that can process the job; then, we get a solution.

In addition, for each population, there is a Pareto frontier solution archives (PA) to record the nondominated solutions of that population; with the evolutionary iteration of the population, PA is updated constantly.

4.5. Mutation Operation. The standard scheme of DE denoted as $DE|rand|1|bin$ is considered in the MODE/MPC algorithm. At generation g, randomly select three different individuals $X_{p_1}(g)$, $X_{p_2}(g)$ and $X_{p_3}(g)$ from PA of the current population; if the solutions in PA are less than 3, then randomly select a solution from that population. Then, variant individual $V_i(g)$ can be generated with the following equation:

$$V_i(g) = X_{p_1}(g) + F \cdot \left(X_{p_2}(g) - X_{p_3}(g)\right), \tag{13}$$

where the scaling factor F is a float point data with a value between [0, 2]. Obviously, the variant individual $V_i(g)$

```
Input: machines and their batches
Output: machines and each batch's start time and complete time of that machine
for k = 1 to m //for each machine
  { for l = 1 to N_k//for each batch on machine M_k
    {computer ST_{k,l} ; }//where ST_{k,l} represents the arrival time of each batch
  sort all batches B_{k,l} of machine M_k nondecreasing according to ST_{k,l};
  for l = 1 to N_k
    {computer ST_{k,l} and CT_{k,l} of the batch B_{k,l} with equations (7) and (8)}
  }
```

Algorithm 1: Scheduling batches

```
Input: mutation individuals V_i(g)
Output:legal mutation individuals V_i(g)
//Step 1:the first n components of variant V were transformed into job permutation
  Define an integer array a (n);
/* Sort the first n elements of variant V nondecreasing, record the sorting position of each component, and then, replace the first n
components of V with their sorting position. */
  for i = 1 to n
    {for j = i to n
      if V (i) > V (j) then
        a (i) = a (i) + 1;
      else
        a (j) = a (j) + 1;
      endif
  V (i) = a (i) + 1;
  }
//Step 2: transform the last n components of variant V into legal machines.
/* Step 2.1: Sort the last n components of variant V nondecreasing, record the sorting position of each component, and then, replace
the last n components of variant V with their sorting position. */
    for i = n + 1 to 2 · n
      { for j = i to 2 · n
          if V (i) > V (j) then
        a (i) = a (i) + 1;
      else
          a (j) = a (j) + 1;
      endif
    V (i) = a(i) + 1; //replace
  }
/* Step 2.2: Replace the last n components of variant V with legal processing machine corresponding to the job */
    for i = n + 1 to 2 * n
      V (i) = m (V (i − n)) + V(i)% M (V (n − i));
```

Algorithm 2: Pseudocode for transforming the mutation individuals

generated by equation (14) does not guarantee that every component obtained is an integer. Therefore, we will use Algorithm 2 to transform the variant to a legitimate solution. Note, the BPM in this paper is numbered according to its capacity from small to large.

4.6. Crossover Operator. In DE algorithm, individuals generate children through crossover operation with variants to explore the solution space. In this paper, we design a two-point crossover operator. Take 10 jobs to be processed on 3 BPMs as an example. It is known that jobs numbered 1–5 can be processed on all 3 machines, jobs numbered 6–8 can be processed on the last two machine, and jobs numbered 9–10 can only be processed on the third machine due to the capacity limitation of batch processing machines. Assume that the variant is (6, 3, 7, 2, 5, 9, 4, 8, 10, 1|3, 2, 2, 1, 1, 3, 1, 2, 3, 1), and the current father individual is (3, 5, 9, 1, 7, 8, 4, 10, 2, 6|1, 2, 3, 2, 2, 3, 1, 3, 1, 2). Then, Figure 2 shows an instance of this two individuals' crossover process, and Algorithm 3 is the pseudocode of crossover operator.

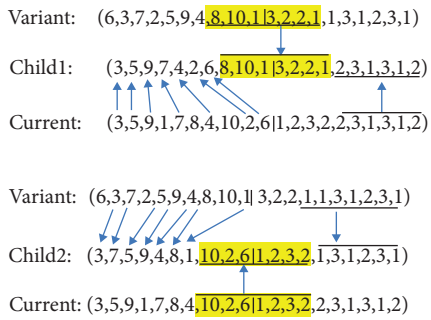

Figure 2: Crossover operator.

4.7. Variable Neighborhood Search (VNS). In order to enhance the local exploration ability and speed up the convergence of the algorithm, a greedy variable neighborhood search algorithm (VNS) is proposed. Each time when a new frontier solution archive set of the population is generated, we will perform a variable neighborhood search for each solution in the set and update the set by using the greedy update mechanism of literature [6]. Three neighborhood structures in [2], swap, insert, and reverse, are used here owning to their effectiveness. Algorithm 4 shows the pseudocode of VNS.

4.8. Framework of MODE/MPC. Considering that decoding methods Rule-1 and Rule-2 can evolve in a larger solution space, they have strong local search ability but slower evolution speed. Decoding Rule-3 and Rule-4 evolve in a relatively small space and have strong exploitation ability but poorer local search ability. In order to make full use of them, we designed the multiobjective DE algorithm with multipopulation coevolution (MODE/MPC). Then, the algorithm includes two populations with the same size. One population decodes with Rule-1 and Rule-2 and 50% probability for each, and the other population decodes with Rule-3 and Rule-4 and 50% probability, respectively. For each population, the frontier solutions will be recorded in external archive set SA. Every several generations, the SA of the two populations will be merged to get the whole society's Pareto frontier solution archive set PA. In turn, we will use PA to reset SA of each population to promote the communication. For the convenience of describing the algorithm, the variables used in the algorithm are listed in Table 1, Algorithm 5 shows the pseudocode, and Figure 3 shows the flow chart.

In Figure 3, considering that the computing time is mainly affected by the decoding algorithm ($O(n \cdot m)$) and the fast nondominated sorting algorithm ($O(2N^2)$), therefore, the computing time for one evolutionary is $O(2 \cdot N \cdot (n \cdot m + N))$, and the total computing time of the algorithm is $O(2 \cdot iter \cdot N \cdot (n \cdot m + N))$. Compared with the scale of jobs n, the number of iterations $iter$ and the population size N, the number of machines m is generally small. Therefore, the overall computing time of the algorithm is $O(iter \cdot N \cdot (n + N))$.

5. Experiment Analysis

In this section, we carry out the computational experiments to evaluate the performance of the proposed algorithm. First, we verify the effectiveness of the strategies proposed in MODE/MPC. Then, we compare the algorithm with other best known multiobjective algorithms to verify its effectiveness. All the algorithms are implemented in C++ language on an intel core i7-8550U, 1.88 GHz PC with 16 GB memory.

5.1. Test Instance. Here, 10 groups of test instances are set. The scale of jobs is 100, 150, 200, 250, and 300, respectively. 3 and 5 UPBPMs are considered. Each test instance is named as $j * m * a *$, where $*$ is an integer number, letter j represents job, and letter m represents machine. For example, j100m3a1 represents the first test instance that 3 BPMs process 100 jobs. The processing time of jobs is evenly distributed in [8, 48]. The detailed data of the test instance are shown in Table 2. The size of jobs is a key parameter affecting batch processing, too many large-size jobs simplify the allocation of BPM; therefore, Table 2 shows the proportion of jobs with different sizes, and the job size is generated by uniform distribution in each interval. In Table 2, the arrival time of jobs is divided into three intervals, namely, $[0, R_1]$, $[R_1, R_2]$ and $[R_2, R_3]$. The proportion of jobs arriving in each time interval is 50%, 25%, and 25%, respectively, and is generated in a uniform manner, where R1, R2, and R3 are calculated by equations (14)~(27). The jobs of special family cluster account for 10% of the total jobs and can only be processed on the machine with the maximum capacity under a special power. In this paper, to facilitate machine scheduling, all machines are indexed from small to large according to capacity.

$$R = 0.05 \cdot \sum_{j=1}^{n} \left(\frac{\sum_{k=1}^{m} p_{j,k}}{m} \right), \tag{14}$$

$$R_1 = rnd \cdot R, \tag{15}$$

$$R_2 = R + rnd \cdot R, \tag{16}$$

$$R_3 = 2 \cdot R * rnd \cdot 2 \cdot R. \tag{17}$$

5.2. Performance Metrics. It is difficult to evaluate the performance of a multiobjective optimization algorithm by a single metric. In order to evaluate the algorithm objectively, the following metrics are selected:

(1) Number of nondominated solutions (NNS). More nondominated solutions mean more choices the decision-maker has. Here, NNS refers to the number of frontier solutions of the algorithm in the set of ideal frontier solutions.

(2) C metric [3]. Let A and B denote the Pareto frontier solution sets obtained by the two algorithms. The

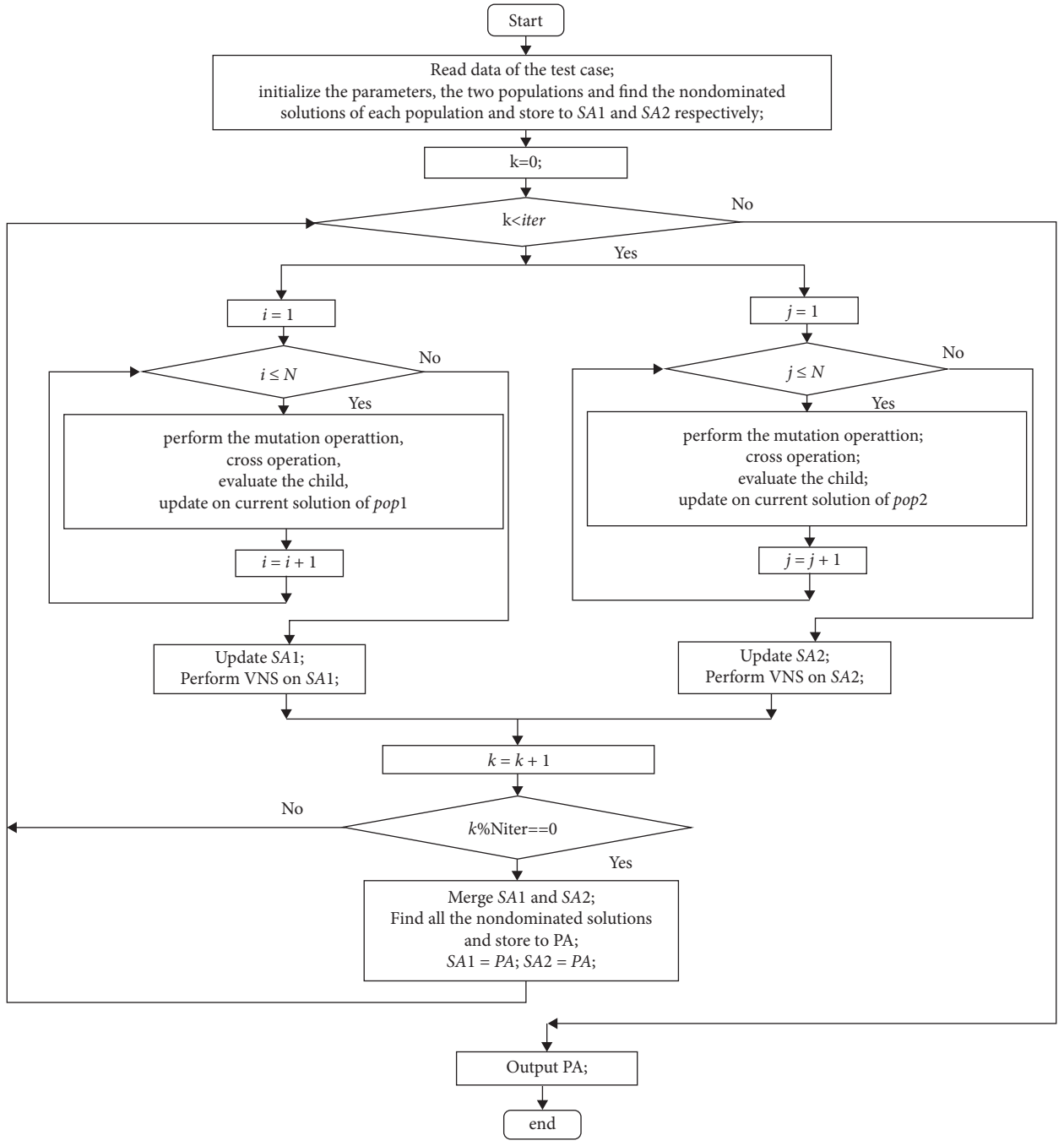

Figure 3: The framework of algorithm MODE/MPC.

value of $C(A, B)$ represents the proportion of solutions in set B dominated by at least one solution in set A, which can be calculated by the following equation:

$$C(A, B) = \frac{|\{f \in B | \exists e \in A: e \prec f\}|}{|f|}, \tag{18}$$

where $C(A, B) \in [0, 1]$, and the closer the value of $C(A, B)$ is to 1, the more solutions in B are dominated by the solutions in A, which indicates that the quality of solutions in set A is better than that in set B. It should be noted that the value of this quota is asymmetric; that is, $C(A, B)$ and $C(B, A)$ need to be calculated separately.

(3) Hypervolume (HV). This metric is used to evaluate the approximation degree between one nondominated solution set obtained by the algorithm and the true Pareto frontier solution. The larger the value of HV, the closer the nondominated solution set to the true Pareto frontier solution, indicating better convergence and diversity of the algorithm. In this paper, the normalized target value is used to calculate HV.

Input: variant solution and current solution
Output: two children solution
Step 1: Randomly generate two random integers between 1 and $2n$, **assign** the smaller value to h1 and the bigger value to h2;
Step 2: Do crossover operator as show in Figure 3.
Step 3: //**Adjust** the machine assignment code for two children
for $i = 1$ to n
{ **if** (child1 $[n + i]$ is not a legal machine for job child1 $[i]$) **then**
update child1 $[n + i]$ using equation (13);
if (child2 $[n + i]$ is not a legal machine for job child2 $[i]$) **then**
update child2 $[n + i]$ using equation (13);
}

Algorithm 3: Pseudocode of crossover operator

Input: a solution π
Output: updated FA
Step 1: Initialize the maximum iteration number iternum of per neighborhood;
Step 2: For each neighborhood, performs the follow operations
{ $k = 0$;
while ($k <$ **iternum**)
{ **Perform** the neighborhood operations;
if ($\mathbf{f}_{\pi_{new}} \prec \mathbf{f}_{\pi}$) **then**// π_{new} dominate π
{$\pi = \pi_{\text{new}}$;
delete the solutions in FA that dominated by π_{new};
$k =$ iternum; }
elseif ($\mathbf{f}_{\pi_{new}} \not\prec \mathbf{f}_{\pi}$ **an d** $\mathbf{f}_{\pi} \not\prec \mathbf{f}_{\pi_{new}}$) then
if ($\mathbf{f}_{\pi_{new}} \not\prec \mathbf{f}_{\text{solution in SA}}$) then
{FA = FA $\cup$ $\{\pi_{\text{new}}\}$;
delete solutions in FA that dominated by π_{new};
$k =$ iternum; }
else
$k = k + 1$;
endif
else
$k = k + 1$;
endif
}
}

Algorithm 4: Pseudocode of VNS

Table 1: Variables and its descriptions in algorithm.

Variable	Description
N	Size of the two populations
pop1, pop2	Arrays of population 1 and population 2 respectively
F	Influence factors of DE
iter	The maximum number of iterations of the algorithm MODE/MPC
W	Weight array of the algorithm MODE/MPC
NB	Neighborhood array of the algorithm MODE/MPC
Niter	The two populations will communicate once every Niter generations
$SA1, SA2, PA$	Frontier solution set of pop1, pop2 and the whole society
k, i	Cyclic control variable

5.3. Parameter Setting. There are some parameters that affect the performance of MODE/MPC; it is necessary to determine appropriate values for these parameters. The Taguchi method is one of the popular statistical analysis methods that can obtain better parameter settings through fewer experiments. Therefore, Taguchi is used to determine the

```
// Step 1: Initialize phase
    Initialize N, F, iter, W, NB, Citer and SA1 = ∅, SA2 = ∅, PA = ∅;
    Read processing data;
    Initialize pop1 and pop 2 with the initialization algorithm in subsection 4.3 and decode them;
    Find the nondominated solutions of pop1 and pop2 and stored them to SA1 and SA2 respectively.
//Step 2: Evolution phase
  k = 0;
  while k <iter
  {  //Step 2.1: Perform evolutionary operation
     {  for i = 1 to N//evolutionary operation on pop1
        {  Do mutation operator in Section 4.5;
           Do crossover operator proposed in Section 4.6 to generate children C1 and C2;
           Decode C1 and C2;
           Update the current solution and its neighborhood with C1 and C2 ;
        }
     Merge pop1 and SA1 into Q;
     Find all the frontier solutions in Q and store them in SA1;
     For each solution in SA1 do VNS and update SA1;
     }
//Step 2.2: Perform evolutionary operation on pop2
     {  Do operation as Step 2.1 on pop2 but decode with Rule-3 or Rule-4; }
     k = k + 1;
//Step 2.3: Communication between populations
  if (k%Niter = 0) then
     {Q = SA1 ∪ SA2;
        Find all the nondominated solutions in Q and store to PA;
        SA1 = PA;
        SA2 = PA;
     }
     endif
  }
//Step 3: output
  Output solution in PA;
```

Algorithm 5: Pseudocode for MODE/MPC

Table 2: Parameter of the test instance.

Factors	Note	Value
Job number	n	$n \in \{50, 100, 150, 200, 250, 300\}$
Machine number	m	$m = 3\vert 5$
Capacity of machines	c_k	$c_1 = 30, c_2 = 40, c_3 = 50,\ c_4 = 60, c_5 = 70$
Power of machines	l_k	$l_k = 5 \cdot (rnd \cdot (k+1) + k - 1)$
Processing time of jobs	$p_{j,k}$	$U[8, 48]$
Arrive time intervals of jobs	r_j	$[0, R_1], [R_1, R_2], [R_2, R_3]$
Proportion of jobs according to the arrival time		{50%, 25%, 25%}
Size of jobs	s_j	$U_1[1, 30], U_2[30, 40], U_3\ [40, 50], U_4\ [50, 60], U_5\ [60, 70]$
Proportion of jobs according to size from small to large	pp_j	$\begin{cases} \{70\%, 20\%, 10\%\}, & \text{where}, k = 3 \\ \{70\%, 10\%, 10\%, 5\%, 5\%\}, & \text{where}, k = 5 \end{cases}$
Proportion of special job clusters	ps	10%

parameter values in this paper. The main parameters considered in MODE/MPC include N (size of each population), NC (the two populations will communicate once every NC evolutions), NV (the maximum iterations times of VNS), and *iter* (the iteration times of the algorithm, where iter = $T \cdot NC$, T is the communication time between two population), which are summarized with three levels in Table 3. According to the Taguchi method, 9 parameter combinations are designed and listed in Table 4. In the experiment, 10 test instances are randomly selected from 10 instance groups and one for each group. The mean value of HV will be statistics to evaluate the performance of the algorithm MODE/MPC with different parameter combinations. The experimental results are shown in Figure 4 in the form of line chart. It can see that algorithm MODE/MPC with parameter combination No. 6 is better than the others.

Table 3: Parameters setting.

Levels	N	NV	NC	T
1	30	2	100	5
2	40	3	120	6
3	50	4	150	7

Table 4: Parameter combinations.

NO.	N	NV	NC	T
1	30	2	100	5
2	30	3	120	6
3	30	4	150	7
4	40	2	120	7
5	40	3	150	5
6	40	4	100	6
7	50	2	150	6
8	50	3	100	7
9	50	4	120	5

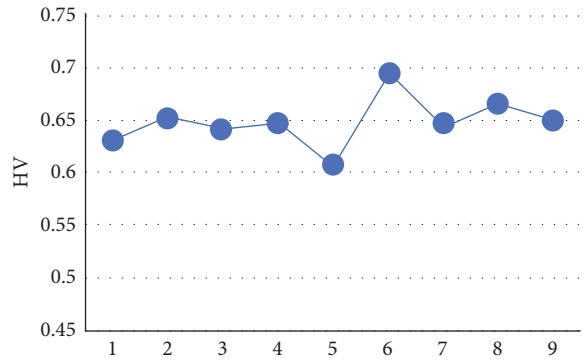

Figure 4: Line chart of mean value of HV on 10 test instances.

So, in the later experiments, the parameter N will be equal to 40, NV will be equal to 4, NC will be equal to 100, and *iter* will be equal to $NC \times T$.

5.4. Effectiveness of Two-Population Cooperative Evolution. To verify the effectiveness of the two populations coevolution strategy, algorithm MODE/MPC and two single population algorithms MODEI and MODEII are compared, where MODEI adopts Rule-1 and Rule-2 to decode the solution, and each rule is selected with 50% probability. Similar to algorithm MODEI, MODEII adopts Rule-3 and Rule-3 to decode the solution. Three algorithms use the same mutation operator and crossover operator and do not use VNS in MODE/MPC. The population size of MODEI and MODEII is 80, the size of each population of MODE/MPC is 40, and the other parameters are consistent with each other. Three algorithms will run 10 times on each test instance. Each time, a population will be randomly generated, and three algorithms will run once based on the population and calculate three evaluation metrics once. Table 5 shows the mean value of NNS and HV in each instance group, and better values are in bold. Figure 4 shows the of line chart of mean C value on 10 instance groups.

In Table 5, MODE/MPC got better value on evaluations metrics of NNS and HV on 9 instance groups except for j100c5a$*$; this shows the effectiveness of the two populations cooperative evolution strategy. At the same time, the mean values of NNS and HV obtained by MODEII are better than those obtained by MODEI in 10 instance groups, which shows that the decoding rules Rule-3 and Rule-4 have better exploitation than Rule-1 and Rule-2. In Figure 5, the mean value of C (MODE/MPC, MODEI) is equal to 1 in 9 instance groups and greater than 0.8 in group j150c5a$*$. This shows that the performance of MODE/MPC is better than that of MODEI. The mean value of C(MODE/MPC, MODEII) is greater than C (MODEII, MODE/MPC) on 9 instance groups except instance group j250c3a$*$; this also shows that the performance of MODE/MPC is better than that of MODEII. Therefore, three evaluation metrics show the effectiveness of the two populations cooperative evolution strategy. The mean values of C(MODEII, MODEI) on 10 instance groups are all greater than 0.7, while C (MODEII, MODEI) on 10 instance groups is equal to 0; these show that the decoding rules Rule-3 and Rule-4 have better exploitation than Rule-1 and Rule-2.

5.5. Effectiveness of VNS and Initialization Strategy. To verify the effectiveness of the VNS and initialization strategy proposed in this paper, we will do comparative experiments with three algorithms, namely, algorithm MODE/MPC with random initialization marked A-RI, algorithm MODE/MPC with random initialization adding VNS marked A-RI-VNS, and algorithm MODE/MPC with initialization strategy adding VNS marked A-IS-VNS. Except for the difference here, the other factors of the three algorithms adopt the same strategy. Three algorithms will run 10 times on each test instance. To make it fair, each time, an initial population will be randomly generated and algorithms A-RI and A-RI-VNS will run based on the same population, while algorithm A-IS-VNS will randomly replace some solution in the initial population with the solutions generated by the initialization strategy AFGA and LSPB and run on the changed population. Table 6 shows the mean value of NNS and HV on each instance group, and better values are in bold. Figures 6–8 show the line chart of the mean value C on each instance group.

In Table 6, compared the results of algorithm A-RI with A-RI-VNS, it is not difficult to find that A-RI-VNS has achieved better results in 10 instance groups in terms of NNS and HV, which shows the effectiveness of VNS. Secondly, comparing the results of algorithm A-RI-VNS with A-IS-VNS, it is can be found that A-IS-VNS has achieved better results in 9 instance groups in terms of NNS and HV, except the group j100c5a$*$. This shows that the initialization strategy is not effective for small-scale problems, but it has good effectiveness for medium-sized and large-scale problems, both from the metrics of NNS and HV. The line charts of C in Figures 6–8 further proved this conclusion. In

Table 5: Comparative experiment on 10 instance groups.

Instance group	MODEI		MODEII		MODE/MPC	
	NNS	HV	NNS	HV	NNS	HV
j100c3a*	0	0.5425	2.3	0.8442	**4.6**	**0.99**
j100c5a*	0	0.4316	**3.1**	**0.9703**	2.4	0.8952
j150c3a*	0	0.3411	7	0.7243	**9.6**	**0.7307**
j150c5a*	1.2	0.4216	4.7	0.7325	**7.2**	**0.7971**
j200c3a*	0	0.3317	5.6	0.6358	**10.8**	**0.6889**
j200c5a*	0	0.32711	5.3	0.6281	**8.9**	**0.7142**
j250c3a*	0	0.3019	7.1	0.5605	**11.4**	**0.6455**
j250c5a*	0	0.2889	6.9	0.8073	**11.7**	**0.8189**
j300c3a*	0	0.2683	9.1	0.5589	**11.4**	**0.5831**
J300c5a*	0	0.0869	3.6	0.4209	**5.7**	**0.4360**

The bold values are the better values got on the instance group.

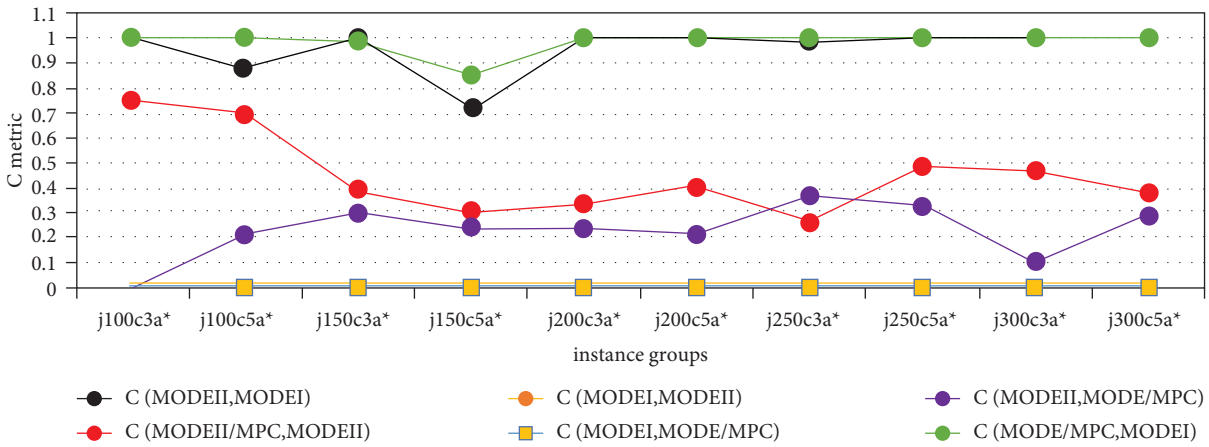

Figure 5: Line chart of MODEI, MODEII and MODE/MPC on 10 groups instance for C.

Table 6: Comparative experiment on 10 instance groups.

	A-RI		A-RI-VNS		A-IS-VNS	
	NNS	HV	NNS	NNS	HV	NNS
j100c3a ∗	0	0.4821	1.4	0.8637	**2.3**	**0.9091**
j100c5a ∗	0	0.5660	**4.2**	**0.8356**	1.6	0.8127
j150c3a ∗	0	0.4572	6.7	0.6607	**13.7**	**0.7387**
j150m5a ∗	0.7	0.4337	4.4	0.6416	**5.7**	**0.7675**
j200c3a ∗	0.1	0.4476	8.4	0.6289	**14.6**	**0.6830**
j200m5a ∗	0.1	0.5336	4.3	0.6474	**9.8**	**0.7943**
j250c3a ∗	0.3	0.4529	10.5	0.6018	**18.3**	**0.6582**
j250m5a ∗	0	0.4673	4.4	0.6559	**8.6**	**0.7701**
j300c3a ∗	1.6	0.3987	4.4	0.4738	**12.8**	**0.5765**
j300m5a ∗	0.2	0.1238	1.9	0.1493	**8.5**	**0.2512**

The bold values are the better values got on the instance group.

Figure 8, it can be seen that the larger the scale of the problem, the greater the impact of initialization strategy on the performance of the algorithm.

5.6. Comparison with Other Algorithms. To evaluate the performance of the algorithm MODE/MPC, three multi-objective optimization algorithms, SPEA2 [36], NSGAII [37], and a better algorithm PDACO [38], proposed recently for the similar BPMSP, have been used for comparative experiments. The solutions representation of NSGAII and SPEA2 is the same as in this paper, and the decoding rule randomly selects Rule-3 and Rule-4 with 50% probability. The cross operation of NSGAII and SPEA2 is the same as Section 4.6; their mutation operator is reverse variation, and the population size is 80. Algorithm PDACO is completely implemented according to [38]; the two populations are all 40, and the other parameters are the same as [38]. Table 7 shows the mean values of C on the 10 instance groups, and Table 8 shows the mean values of HV and NNS. The box plot

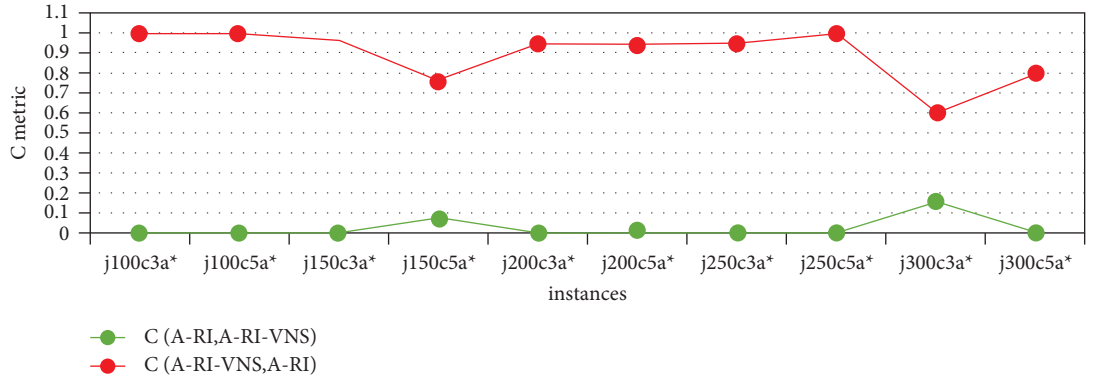

Figure 6: Line chart of algorithm A-RI and A-RI-VNS on 10 instance groups for C metric.

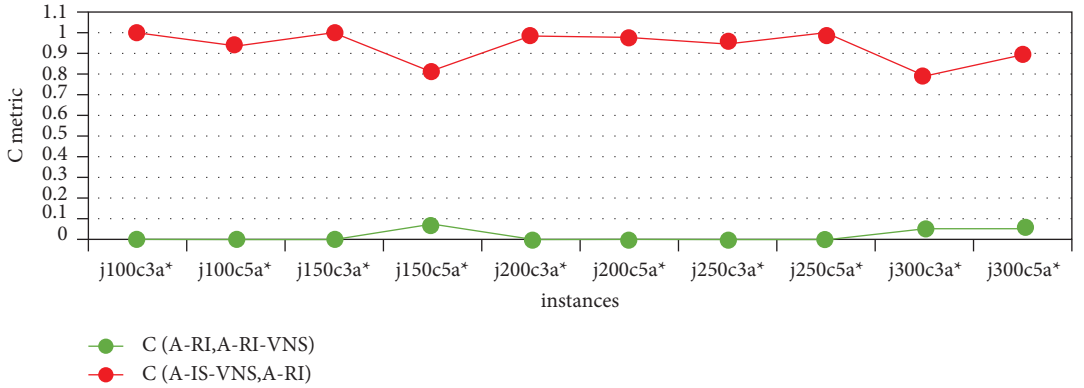

Figure 7: Line chart of algorithm A-RI and A-SI-VNS on 10 instance groups for C metric.

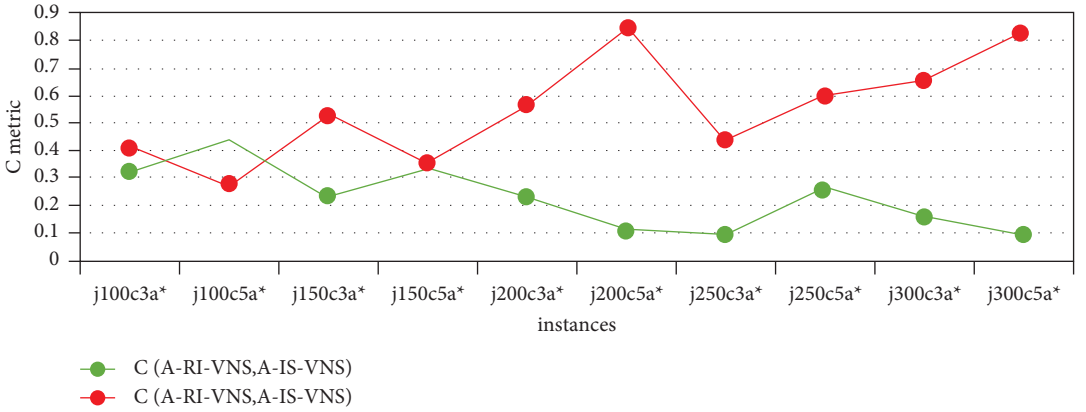

Figure 8: Line chart of algorithm A-RI-VNS and A-SI-VNS on 10 instance groups for C metric.

Table 7: Comparison of algorithms in terms of metric *C*.

Instance group	*C* (MODE/MPC, NSGAII)	*C* (NSGAII, MODE/MPC)	*C* (MODE/MPC, SPEA2)	*C* (SPEA2, MODE/MPC)	*C* (MODE/MPC, PDACO)	*C* (PDACO, MODE/MPC)
j100c3a ∗	**0.4744**	0.2135	**0.4482**	0.2316	0.3364	**0.3567**
j100c5a ∗	**0.3473**	0.3126	**0.3721**	0.3162	**0.3864**	0.3207
j150c3a ∗	**1**	0	**0.9667**	0	**0.4333**	0.1667
j150c5a ∗	**0.3109**	0.0167	**0.4314**	0.1133	**0.2731**	0.2483
j200c3a ∗	**1**	0	**1**	0	**0.4013**	0.1027
j200c5a ∗	**1**	0	**1**	0	**0.5892**	0.2217
j250c3a ∗	**1**	0	**1**	0	**0.6724**	0.2328
j250c5a ∗	**1**	0	**1**	0	**0.6743**	0.2526
j300c3a ∗	**1**	0	**1**	0	**0.5412**	0.1722
j300c5a ∗	**1**	0	**1**	0	**0.6438**	0.2851

The bold values are the better values got on the instance group.

Table 8: Comparison of four algorithms in terms of metric HV and NNS.

	MODE/MPC		NSGAII		SPEA2		PDACO	
	HV	NNS	HV	NNS	HV	NNS	HV	NNS
j100c3a ∗	**0.88099235**	2.51	0.80841435	1.36	0.79980652	0.94	0.8512306	**4.82**
j100c5a ∗	**0.92857121**	**4.43**	0.88150478	2.04	0.90274042	1.98	0.90633125	2.4
j150c3a ∗	**0.67946238**	**17.58**	0.36591312	4.67	0.3297343	5.23	0.60406404	7.6
j150c5a ∗	**0.69221336**	**11.4**	0.32439498	4.32	0.28474322	3.82	0.5826978	6.56
j200c3a ∗	**0.71812236**	**26.18**	0.31180754	0.86	0.30210736	1.45	0.5715248	13.32
j200c5a ∗	**0.7043381**	**13.46**	0.35194444	0	0.28043654	0	0.56445422	6.4
j250c3a ∗	**0.68820244**	**24.78**	0.43827707	0	0.31880564	0	0.60917126	11.38
j250c5a ∗	**0.73206882**	**19.42**	0.48377166	0	0.43654152	0	0.58227674	9.34
j300c3a ∗	**0.6021029**	**26.52**	0.32820564	0	0.28901574	0	0.63145038	10.04
j300c5a ∗	**0.2873152**	**15.15**	0.33692618	0	0.28901574	0	0.63837386	7.12

The bold values are the better values got on the instance group.

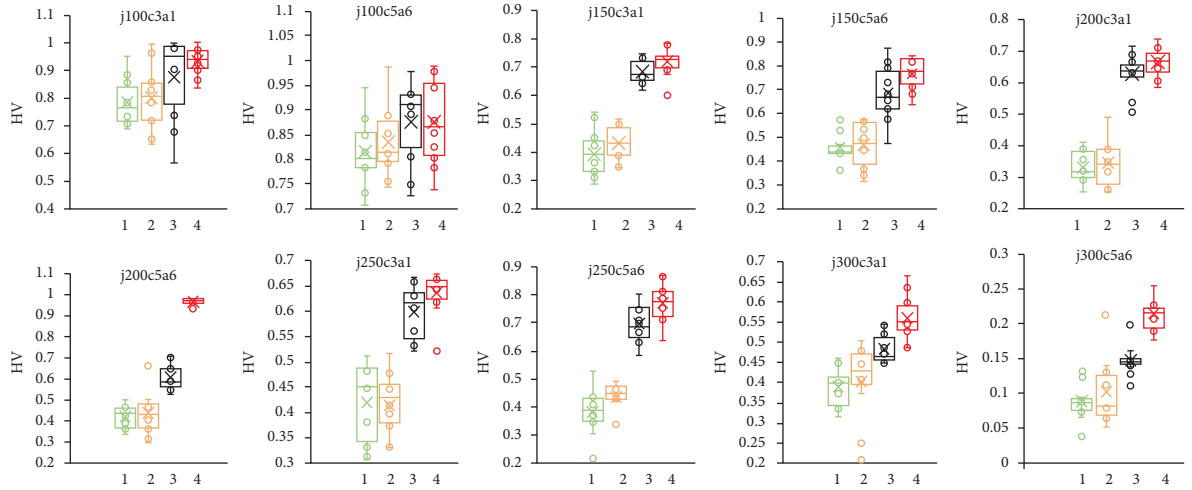

Figure 9: Box plot of the four comparative algorithms on the first instance of each instance group, where 1 represents algorithm SPEA2, 2 represents NSGAII, 3 represents algorithm PDACO, and 4 represents algorithm MODE/MPC.

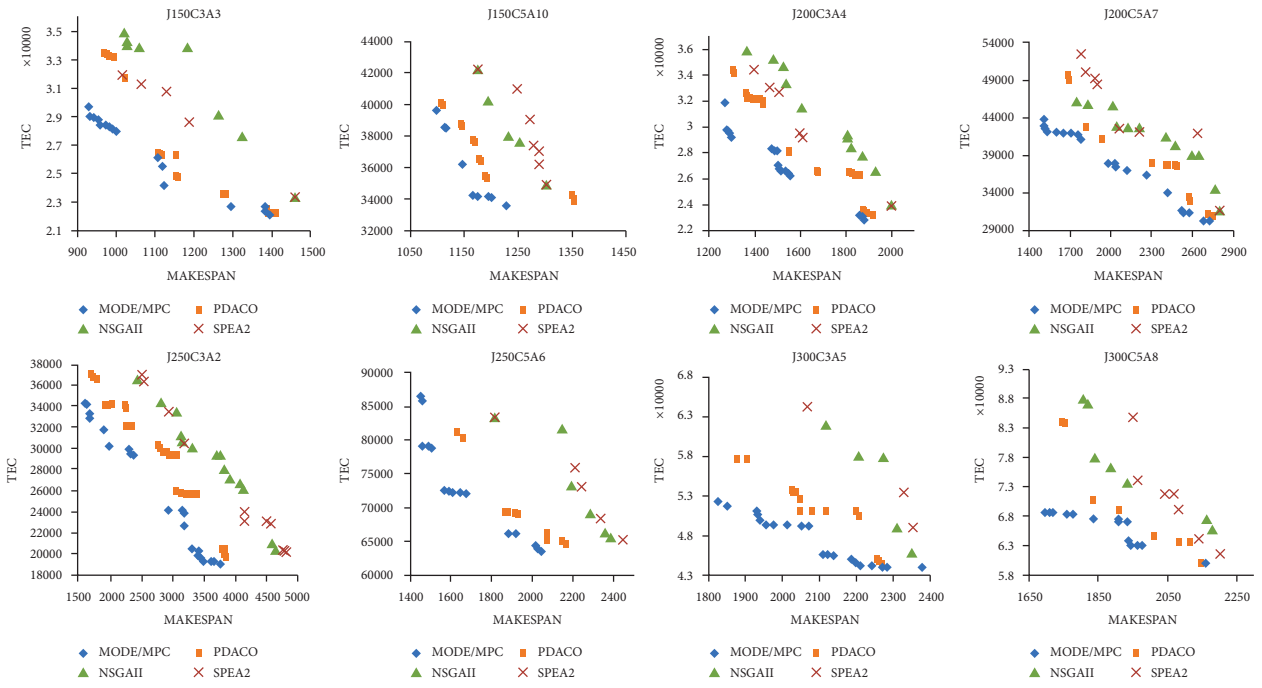

Figure 10: The scatter diagram of Pareto frontier solutions obtained by four different algorithms in eight instances.

in Figure 9 shows the statistical results of the four algorithms on the first instance of each instance group. To compare the solution quality obtained by the four algorithms on medium and large-scale instances, the distributions of the solutions on 8 different instances with 150,200,250,300 jobs are shown in Figure 10, where x-axis and y-axis denote the values of makespan and TEC, respectively.

Table 7 presents the mean value of four comparative algorithms in terms of C. According to the mean value of C on each group, algorithm MODE/MPC outperforms SPEA2 and NSGAII in all instance groups. Comparing MODE/MPC with PDACO, the mean value of C (MODE/MPC, PDACO) is greater in 9 instance groups than C (PDACO, MODE/MPC), except the small-scale instance group j100c3a$*$. Therefore, MODE/MPC outperforms PDACO from the point of C metric. In Table 8, the mean values of HV gotten by MODE/MPC in all instance groups are greater than those of the other three algorithms, denoting that the hypervolume surrounded by the Pareto frontier solutions obtained by algorithm MODE/MPC is better than that of the other three algorithms. Therefore, the Pareto frontier solutions obtained by algorithm MODE/MPC have better distribution and convergence. In Figure 9, for each instance, the box of MODE/MPC is better than the other three algorithms; this proved that MODE/MPC outperformed the other three algorithms in terms of HV. At the same time, the smaller the length of the box, the better the robustness of the algorithm. Therefore, the robustness of MODE/MPC is also better than other three algorithms. In terms of NNS, except for instance group j100c3a$*$, MODE/MPC got more nondominated solutions in the other 9 instance groups. From the perspective of NNS metric, the larger the problem scale, the better the MODE/MPC performance compared with the other three algorithms. In Figure 10, it is obvious that the frontier solutions of MODE/MPC are closer to the Pareto frontier solution and have better distribution than the other three algorithms. Based on the above analysis, the MODE/MPC outperforms the SPEA2, NSGAII, and PDACO in addressing UPBPMSP.

6. Conclusion

In this paper, an algorithm MODE/MPC is proposed to solve the scheduling problem of UPBMP to minimize the TEC and makespan simultaneously. Firstly, more constraints are considered here, such that the batch processing machine has different capacity and power, the jobs to be processed have different arrival time, different processing time, and size, and some jobs need specific machines and specific processing power. Secondly, the algorithm consists of two populations, and each of them has different search centers, so as to realize division of labor and cooperation and ensure the diversity and distribution of solutions. The VNS and the initialization strategy further improve the performance of the algorithm MODE/MPC. The experiment results show that the MODE/MPC significantly outperforms NSGAII, SPEA2, and PACO. In the future, the research can be expanded from the following aspects.

Constraints: in addition to the constraints considered in this paper, there are other constraints that can be considered, such as different deadlines of jobs, machine malfunction, and different speeds of machines.

Algorithm: in the real production environment, with the change of time, the production objectives will also change. Therefore, the dynamic multiobjective production scheduling algorithm will become a research direction.

Data Availability

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

Conflicts of Interest

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

Acknowledgments

This work was supported by Liaoning Educational Committee Program (LJKZ0489and LJKZ0486); open fund projects of Artificial Intelligence Key Laboratory of Sichuan Province (2020RYJ04);

References

[1] F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1–13, 2020.

[2] C. Song, "A hybrid multi-objective teaching-learning based optimization for scheduling problem of hybrid flow shop with unrelated parallel machine," *IEEE Access*, vol. 99, p. 1, 2021.

[3] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete Jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[4] F. Zhao, L. Zhao, L. Wang, and H. Song, "An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion," *Expert Systems with Applications*, vol. 160, Article ID 113678, 2020.

[5] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, "Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5094–5109, 2020.

[6] S. Zhou, X. Li, N. Du, Y. Pang, and H. Chen, "A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost," *Computers & Operations Research*, vol. 96, no. 8, pp. 55–68, 2018.

[7] C. Zhu-Min, "A heuristic algorithm for hybrid flow shop's master scheduling," *Industrial Engineering & Management*, vol. 14, no. 5, pp. 69–78, 2009.

[8] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, "A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution," *Applied Soft Computing*, vol. 75, pp. 728–749, 2019.

[9] T. Gu, S. Li, Y. Lin, and X. Wu, "Research on the re-entrant batch discrete flow shop scheduling for periodic annealing

furnace as batch processor," *Journal of Mechanical Engineering*, vol. 56, no. 2, pp. 220–232, 2020.

[10] H. Yuan and G. Guo, "Vehicle cooperative optimization scheduling in transportation cyber physical systems," *Acta Automatica Sinica*, vol. 45, no. 1, pp. 143–152, 2019.

[11] L. Tang, H. Gong, J. Liu, and F. Li, "Bicriteria scheduling on a single batching machine with job transportation and deterioration considerations," *Naval Research Logistics*, vol. 61, no. 4, pp. 269–285, 2014.

[12] Y. Zarook, J. Rezaeian, R. Tavakkoli-Moghaddam, and I. Mahdavi, "Minimization of makespan for the single batch-processing machine scheduling problem with considering aging effect and multi-maintenance activities," *International Journal of Advanced Manufacturing Technology*, vol. 76, no. 9, pp. 1879–1892, 2015.

[13] H. Zhou, J. Pang, P.-K. Chen, and F.-D. Chou, "A modified particle swarm optimization algorithm for a batch-processing machine scheduling problem with arbitrary release times and non-identical job sizes," *Computers & Industrial Engineering*, vol. 123, no. 9, pp. 67–81, 2018.

[14] Z. Xin, X. Li, and J. Wang, "Local search algorithm with path relinking for single batch-processing machine scheduling problem," *Neural Computing & Applications*, vol. 28, no. 1, pp. 313–326, 2017.

[15] İ. Muter, "Exact algorithms to minimize makespan on single and parallel batch processing machines," *European Journal of Operational Research*, vol. 285, no. 2, pp. 470–483, 2020.

[16] X. Li, Y. Huang, Q. Tan, and H. Chen, "Scheduling unrelated parallel batch processing machines with non-identical job sizes," *Computers & Operations Research*, vol. 40, no. 12, pp. 2983–2990, 2013.

[17] F. Chou and H. Wang, "Minimizing total weighted tardiness on parallel batch-processing machine scheduling problems with varying machine capacities," *Applied Mechanics and Materials*, vol. 110-116, pp. 3906–3913, 2012.

[18] I. V. Lerner and Y. E. Lozovik, "Mott exciton in a quasi-two-dimensional semiconductor in a strong magnetic field," *Journal of Experimental and Theoretical Physics*, vol. 51, no. 3, pp. 559–574, 1980.

[19] J. Huang and L. Wang, "Makespan minimization on single batch-processing machine considering preventive maintenance," in *Proceedings of the 2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pp. 294–298, Singapore, April 2018.

[20] A. H. Kashan, B. Karimi, and F. Jolai, "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes," *International Journal of Production Research*, vol. 44, no. 12, pp. 2337–2360, 2006.

[21] X. P. Guo, "A variable neighborhood based memetic algorithm for scheduling single batch processing machine with non-identical job sizes," *Applied Mechanics and Materials*, vol. 197, pp. 489–495, 2012.

[22] N. Rafiee Parsa, B. Karimi, and S. M. Moattar Husseini, "Minimizing total flow time on a batch processing machine using a hybrid max-min ant system," *Computers & Industrial Engineering*, vol. 99, no. 9, pp. 372–381, 2016.

[23] F. Jolai Ghazvini and L. Dupont, "Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes," *International Journal of Production Economics*, vol. 55, no. 3, pp. 273–280, 1998.

[24] Z.-h. Jia, H. Zhang, W.-t. Long, J. Y.-T. Leung, K. Li, and W. Li, "A meta-heuristic for minimizing total weighted flow time on parallel batch machines," *Computers & Industrial Engineering*, vol. 125, pp. 298–308, 2018.

[25] Z.-h. Jia, X.-x. Zhuo, J. Y.-T. Leung, and K. Li, "Integrated production and transportation on parallel batch machines to minimize total weighted delivery time," *Computers & Operations Research*, vol. 102, pp. 39–51, 2019.

[26] Z. Jia, J. Yan, J. Y. T. Leung, K. Li, and H. Chen, "Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities," *Applied Soft Computing*, vol. 75, pp. 548–561, 2019.

[27] S. Zhou, M. Jin, and N. Du, "Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times," *Energy*, vol. 209, 2020.

[28] R. Zhang, P.-C. Chang, S. Song, and C. Wu, "A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes," *Knowledge-Based Systems*, vol. 116, no. 15, pp. 114–129, 2017.

[29] O. Shahvari and R. Logendran, "An Enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes," *Computers & Operations Research*, vol. 77, pp. 154–176, 2017.

[30] S.-y. Qian, Z.-h. Jia, and K. Li, "A multi-objective evolutionary algorithm based on adaptive clustering for energy-aware batch scheduling problem," *Future Generation Computer Systems*, vol. 113, pp. 441–453, 2020.

[31] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[32] Z. Shengchao, X. Lining, Z. Xu, N. Du, L. Wang, and Q. Zhang, "A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1430–1442, 2021.

[33] J. Liang, P. Wang, L. Guo, and B. Qu, "Multi-objective flow shop scheduling with limited buffers using hybrid self-adaptive differential evolution," *Memetic Computing*, vol. 11, no. 6, 2019.

[34] W. Zhang, Y. Wang, Y. Yang, and M. Gen, "Hybrid multi-objective evolutionary algorithm based on differential evolution for flow shop scheduling problems," *Computers & Industrial Engineering*, vol. 130, pp. 661–670, 2019.

[35] J. E. C. Arroyo and J. Y.-T. Leung, "Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times," *Computers & Operations Research*, vol. 78, pp. 117–128, 2017.

[36] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," *Technical Report Gloriastrasse*, pp. 1–21, 2001.

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[38] Z. H. Jia, Y. Wang, and Y. W. Zhang, "A bi-objective synergy optimization algorithm of ant colony for scheduling on non-identical parallel batch machines," *Acta Automatica Sinica*, vol. 46, no. 6, pp. 1121–1135, 2020.