Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 8696985, 24 pages
https://doi.org/10.1155/2017/8696985

Research Article

An Efficient Multiobjective Backtracking Search Algorithm for Single Machine Scheduling with Controllable Processing Times

Chao Lu, Liang Gao, Xinyu Li, Qi Wang, Wei Liao, and Qingyao Zhao

State Key Lab of Digital Manufacturing Equipment & Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China

Correspondence should be addressed to Xinyu Li; lixinyu@mail.hust.edu.cn

Received 2 October 2016; Revised 13 January 2017; Accepted 13 February 2017; Published 14 March 2017

Academic Editor: Franck Massa

The scheduling problem with controllable processing times (CPT) is one of the most important research topics in the scheduling field due to its widespread application. Because of the complexity of this problem, a majority of research mainly addressed single-objective small scale problems. However, most practical problems are multiobjective and large scale issues. Multiobjective metaheuristics are very efficient in solving such problems. This paper studies a single machine scheduling problem with CPT for minimizing total tardiness and compression cost simultaneously. We aim to develop a new multiobjective discrete backtracking search algorithm (MODBSA) to solve this problem. To accommodate the characteristic of the problem, a solution representation is constructed by a permutation vector and an amount vector of compression processing times. Furthermore, two major improvement strategies named adaptive selection scheme and total cost reduction strategy are developed. The adaptive selection scheme is used to select a suitable population to enhance the search efficiency of MODBSA, and the total cost reduction strategy is developed to further improve the quality of solutions. For the assessment of MODBSA, MODBSA is compared with other algorithms including NSGA-II, SPEA2, and PAES. Experimental results demonstrate that the proposed MODBSA is a promising algorithm for such scheduling problem.

1. Introduction

The scheduling problem with controllable processing times (CPT) has received increasing attention in manufacturing fields. The CPT denotes that operation duration of a job can be compressed or expanded by adjusting the available resources like fuel, equipment, manpower, and so on [1–4]. Most classical scheduling problems assume that the job processing times are constant values. However, this assumption sometimes violates practical production. We can observe that job processing times are controllable in some cases. For example, in the chemical industry, the processing times of a chemical substance can be compressed by catalyzer or expanded by inhibitor, which requires additional costs [5]. In the CNC manufacturing industry, the job processing times can be controlled by adjusting the cutting speed or the feed rate, which also entails more costs [6, 7]. Therefore, the consideration of CPT in scheduling problems may be more applicable for some manufacturing systems.

This paper studies a single machine scheduling problem with CPT (SSPWCPT) for the following reasons: it fills the gap where a multiobjective evolutionary approach for the large scale SSPWCPT with multiple criteria has been rarely reported. Most existing methods to deal with single-objective SSPWCPT can be classified into two categories, namely, exact and approximate approaches. Exact approaches like branch and bound algorithm have been successfully applied to small scale SSPWCPT. Unfortunately, the exact methods are incapable of solving large scale SSPWCPT. On the contrary, the approximate approaches can solve large scale SSPWCPT within an acceptable time. It is therefore important to conduct a study on efficiency and effectiveness of approximate algorithms for this studied problem. However, purely single-objective SSPWCPT cannot fully reflect the requirements of the real-world scheduling applications. Thus, the problem considers two important criteria, namely, total tardiness and total compression cost. The two objectives are widely accepted in single machine with CPT because they can affect satisfaction of customer and profits of enterprise. One straightforward strategy for addressing the multiobjective optimization problem (MOP) is to combine multiple objectives into a scalar function by giving fixed weights to each objective function [8]. Nevertheless, in most practical scheduling problems, multiple criteria are usually in conflict with each other [9–11]. In addition, the objective weight is difficult to determine due to different objective scales. Therefore, it is better to handle multiple objectives with knowledge about Pareto dominance. Pareto-based multiobjective evolutionary algorithm (MOEA) is suitable for solving multiobjective scheduling problems since it can yield nondominated solutions in a single run [4, 12, 13].

Recently, backtracking search algorithm (BSA) [14] is a promising method for solving single-objective scheduling problem due to its high convergence speed and ease of implementation. The core idea of BSA is that dual population is utilized to search for an optimal solution during search process. Based on the effectiveness of BSA and characteristics of the MOP, a novel multiobjective discrete backtracking search algorithm (MODBSA) is proposed to solve this multiobjective SSPWCPT. Although multiobjective backtracking search algorithm has been developed, it is mainly used to address continuous optimization problems [15]. The main reason for adopting this MODBSA is that the problem under study is NP-hard [16] and BSA has been demonstrated to be an effective approach for solving this category of problem [17, 18]. Furthermore, dual population in BSA can be utilized to store information of different population for keeping diversity of population. In addition, to the best of the authors' knowledge, there exists no research about multiobjective BSA in the field of scheduling problem in literature. These reasons drive us to develop an efficient multiobjective algorithm based on BSA for this discrete optimization problem.

To achieve good performance of the proposed algorithm on SSPWCPT, several strategies of this method are developed. According to the characteristic of the addressed problem, the proposed MODBSA uses a two-part encoding scheme. The first part represents the job permutation and the second one denotes the amount of compression processing times of job. Moreover, an adaptive scheme is proposed to select a suitable population for enhancing search efficiency. Meanwhile, a total cost reduction strategy is proposed to improve convergence toward the efficient solution.

The remainder of the paper is organized as follows. In Section 2, some relevant work is described. In Section 3, a definition of the studied problem is stated. In Section 4, the proposed MODBSA for the SSPWCPT is elaborated. The experimental results of the proposed MODBSA are presented and analyzed in Section 5. Conclusions and future work are given in Section 6.

2. Literature Review

The SSPWCPT has been extensively studied since Vickson [19] initiated the CPT in a single machine scheduling problem. However, most single-objective methods or weighted sum methods have been used to solve SSPWCPT in previous literature. For example, Janiak and Kovalyov [20] addressed the SSPWCPT with deadlines and processing time which was a linear decreasing function of the amount of compression. They proposed an $O(n \log n)$ algorithm for small scale cases with the objective to minimize the total resource consumption. Shabtay and Kaspi [21] considered a single machine scheduling problem for minimizing total weighted completion time. They presented and analyzed some special cases that were solvable by using polynomial time algorithms. They also gave some heuristic algorithms and a dynamic programming for the general case. Kayan and Akturk [6] studied a bicriteria scheduling problem on a single CNC machine. They proposed an exact algorithm and four heuristic methods to find a set of discrete points to approximate the continuous trade-off curve on small scale problem. Cheng et al. [22] considered a single machine scheduling problem where both job processing times and release dates were controllable. They proposed an $O(n^2)$ algorithm to solve this problem for minimizing the sum of makespan and the total compression cost. Yin and Wang [23] presented a heuristic to address a single machine scheduling problem with CPT and learning effect. The objective of this problem is to minimize a cost function including makespan, total completion, and total absolute differences in completion times. Xu et al. [24] proposed a polynomial time algorithm of $O(n^2)$ for this small scale problem with the objective of minimizing the total tardiness. Tseng et al. [25] proposed a net benefit compression (NBC) heuristic to minimize total tardiness plus compression cost on a SSPWCPT. Kayvanfar et al. [26, 27] extended the work of Tseng et al. [25] by designing a net benefit compression-net benefit expansion (NBC-NBE) algorithm. Yedidsion et al. [28] proved the complexity of a single machine scheduling problem with CPT for several related criteria. Yin et al. [29] addressed a single machine batch delivery scheduling problem with an assignable common due date and CPT. They provided an $O(n^5)$ dynamic programming algorithm and developed an $O(n \log n)$ algorithm to find the optimal solution for minimizing a cost function consisting of earliness, tardiness, job holding, and due date assignment. Nearchou [8] considered this SSPWCPT and the objective of this problem was to minimize total weighted completion time plus the compression cost. Several single-objective population-based metaheuristics were adopted to solve this problem. Giglio [30] considered a class of single machine family scheduling problems, characterized by unreliable behavior of the machine, CPT, sequence-dependent setups, and due dates. The objective of this problem is to minimize the sum of the total weighted tardiness, the total weighted consumption cost, and the total setup cost. He proposed a dynamic programming approach to solve this problem. A survey on scheduling with CPT was provided by Shabtay and Steiner [31]. In this paper, we only consider the single machine scheduling problem with CPT; thus, the scheduling problems in other environment are not reviewed.

As stated previously, the main criterion of most research is a single-objective or combined objective with the weighted sum approach. However, multiobjective scheduling problem should be the trend for the real-life scheduling production in the future. The majority of the previous research is focused on heuristics for addressing small scale problems. However, the research on addressing the studied problem using metaheuristics is relatively scarce. In fact, metaheuristics are very efficient in solving such type of large scale scheduling problems. To the best of authors' knowledge, efficient MOEAs based on BSA have not yet been applied to SSPWCPT in the previous literature. Thus, the motivation of this study from a theoretical perspective is to develop an efficient MOEA for minimizing both total tardiness and total compression cost on SSPWCPT. Nevertheless, the no free lunch theory implies that algorithm's performance is sensitive to the problem considered. Therefore, we should develop a more efficient MOEA than other high-performing metaheuristics if some characteristics of the problem as well as some techniques are considered.

3. Problem Formulation

The scheduling problem under study can be described as follows. A set of n independent jobs is available at time zero on a single machine. One machine can process at most one job at a time and job preemption is not allowed. Each job has a normal discrete processing time. The normal job processing times can be compressed by allocating additional resources, which entails compression cost. Assume that resources can only be assigned in discrete quantities and the job with normal processing time incurs no extra cost.

The parameters and decision variables used throughout the paper are as shown in the Notations.

The problem is an extension of the scheduling problem with the fixed processing time [32]. It can be defined as a multiobjective mathematical model for minimizing total tardiness and total compression cost simultaneously, which is formulated as follows:

$$\begin{aligned} \min \quad & f_1 = \sum_{j=1}^{n} T_j \\ \min \quad & f_2 = \sum_{j=1}^{n} c_j \cdot x_j \end{aligned} \tag{1}$$

$$\text{Subject to} \quad My_{jk} + S_j - (p_k - x_k) \geq S_k; \quad j,k = 1,\ldots,n,\ j < k \tag{2}$$

$$M\left(1 - y_{jk}\right) + S_k - \left(p_j - x_j\right) \geq S_j; \quad j,k = 1,\ldots,n,\ j < k \tag{3}$$

$$S_j + p_j - x_j - d_j \leq T_j; \quad j = 1,\ldots,n \tag{4}$$

$$m_j \geq x_j; \quad j = 1,\ldots,n \tag{5}$$

$$T_j \geq 0,\ x_j \geq 0,\ S_j \geq 0,\ y_{jk} = 0 \text{ or } 1, \quad j,k = 1,\ldots,n,\ j < k. \tag{6}$$

Equations (1) define the objective to minimize the total tardiness and the total compression cost. Constraints (2) and (3) guarantee the precedence relationship and define that only one job can be processed at any instance in time. Constraints (4) impose the tardiness of each job in a sequence. Constraints (5) limit the amount of compression processing time for all jobs. Constraints (6) show nonnegativity of variables. This studied problem is NP-hard since the single machine total tardiness problem is already NP-hard [16]. It is still valuable for us to research and explore such a problem.

4. The Proposed MODBSA for SSPWCPT

In this section, firstly we give a basic background on multiobjective optimization and then describe the original BSA and present a framework of the proposed MODBSA. Finally, the main improvement strategies of the MODBSA for optimizing SSPWCPT are elaborated.

4.1. Background on Multiobjective Optimization. To better understand the proposed MODBSA for solving the above problem, we begin with a brief introduction of the basic concept of MOEA. Without loss of generality, a multiobjective optimization problem (MOP) can be formally defined as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \min\left[f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\right] \\ & \mathbf{x} = \left(x_1, x_2, \ldots, x_n\right) \in R^n \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geq 0, \quad i = 1,\ldots,k \\ & h_j(\mathbf{x}) = 0, \quad j = 1,\ldots,p, \end{aligned} \tag{7}$$

where $f_m(\mathbf{x})$ indicates the mth subobjective function, $\mathbf{x}$ is a vector of the solution, which should satisfy the above constraints, R^n is the decision variable space, and $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are inequality constraint and equality constraint, respectively. k and p denote the total number of inequality constraints and equality constraints, respectively.

Let $\mathbf{a}$ and $\mathbf{b} \in R^n$, and a vector $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$ dominates another vector $\mathbf{b} = [b_1, b_2, \ldots, b_n]^T$ (denoted by $\mathbf{a} \prec \mathbf{b}$) if $\mathbf{a}$ is not inferior to $\mathbf{b}$ for any of m objectives and $\mathbf{a}$ is superior to $\mathbf{b}$ for at least one objective. A solution $\mathbf{x}^* \in R^n$ is a Pareto optimal vector if there is not any solution $\mathbf{x} \in R^n$ that dominates $\mathbf{x}^*$. The corresponding objective function in the objective space forms the Pareto optimal front point $f(\mathbf{x}^*)$. For a Pareto optimal solution, the improvement in any objective will incur the deterioration of at least another objective. A set of all the Pareto optimal solutions is called Pareto optimal set (PS^*), while the set of all Pareto optimal front vectors is called the Pareto optimal front (PF^*). The main goal of multiobjective optimization is to find PF^*. However, in general, a Pareto front consists of a large number of points. Therefore, a good Pareto front contains a limited number of points which should be as close as possible to the PF^* and uniformly spread as well.

4.2. Brief Introduction of Backtracking Search Algorithm (BSA). The original BSA is described before presenting the MODBSA, and its main steps include five processes:

```
Input: $Mutant$, $mixrate$, $N$ and $D$.
Output: $T$: Trial population.
(1) $\text{map}(1:N, 1:D) = 1$
(2) if $a < b \mid a, b \sim U(0,1)$ then
(3)     for $i$ from 1 to $N$
(4)         $\text{map}_{i,u(1:\lceil \text{mixrate} \cdot \text{rand}(D) \rceil)} = 0 \mid u = \text{permuting}(\langle 1, 2, \ldots, D \rangle)$
(5)     end
(6) else
(7)     for $i$ from 1 to $N$
(8)         $\text{map}_{i,\text{randi}(D)} = 0$
(9)     end
(10) end
(11) $T \coloneqq \text{Mutant}$
(12) for $i$ from 1 to $N$
(13)     for $j$ from 1 to $D$
(14)         if $\text{map}_{i,j} = 1$ then $T_{i,j} \coloneqq P_{i,j}$
(15)     end
(16) end
```

Algorithm 1: Crossover operation.

initialization, selection 1, mutation, crossover, and selection II [14]. Note that the original BSA is proposed for solving continuous problems; thus, the decision variables are real numbers.

4.2.1. Initialization. Randomly initialize a population P, which can be formulated as follows:

$$P_{i,j} = L_j + \left(U_j - L_j\right) \times \text{random} \qquad i = (1, 2, \ldots, N), \quad j = (1, 2, \ldots, D), \tag{8}$$

where N and D are the population size and the number of decision variables, respectively, random is a real number value uniformly distributed in $[0, 1]$, L_j and U_j denote the lower and upper bound for the jth decision variable of the ith solution.

4.2.2. Selection I. To obtain the search direction, the aim of BSA's Selection I stage is to determine the historical population old P. The initial historical population is defined using the following formulation:

$$\text{old}\, P_{i,j} = L_j + \left(U_j - L_j\right) \times \text{random} \qquad i = (1, 2, \ldots, N), \quad j = (1, 2, \ldots, D). \tag{9}$$

BSA has a choice to generate the old P at the beginning of each generation through the following form:

$$\text{if } a < b \text{ then old}\, P \coloneqq P \mid a, b \in [0, 1], \tag{10}$$

where $\coloneqq$ is the initialization operation; a and b are real number values in the range $[0, 1]$. The above equation defines that the BSA's historical population (old P) is from either the previous population or the old P itself. Once old P is generated, a permutation function is adopted to randomly alter the position of the individuals in old P.

4.2.3. Mutation. The form of a trial population T (i.e., offspring pupation) by the mutation operation can be written as follows:

$$T = P + (\text{old}\, P - P) \times F, \tag{11}$$

where F is a scale factor which controls the amplitude of the search-direction matrix (old $P - P$). The value $F = 3 \cdot \text{random}$, where random is a real number uniformly distributed in $[0, 1]$. Since the historical population is employed in the calculation of the search-direction matrix, BSA takes some advantages of previous generations to obtain a trial population.

4.2.4. Crossover. The final trial population T is obtained by crossover in BSA. The crossover of BSA works as follows. First, compute a binary integer-valued matrix (map) whose dimension is $N \times D$. It denotes that the individuals of T are produced by using the relevant individuals of P. Then, if $\text{map}_{i,j} = 1$, where $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, D\}$, T is updated with $T_{i,j} \coloneqq P_{i,j}$. Algorithm 1 provides the crossover operation of BSA.

In Algorithm 1, $\lceil\ \rceil$ represents the ceiling function (in line 4). The rand parameter indicates a random real number in $[0, 1]$. The mixrate parameter controls the number of elements of individuals that will mutate in a trial individual by using $\lceil \text{mixrate} \cdot \text{rand}(D) \rceil$. Two strategies are employed to define BSA's map. In the first strategy, mixrate is used to define map (in lines 3–5). In the second strategy, only one randomly selected element is going to mutate (in lines 7–9).

4.2.5. Selection II. In the stage of BSA's Selection II, T_i is used to update the P_i if it is better than P_i in terms of the fitness value. If the best solution P_{best} outperforms the global optimal solution found so far, the global optimal solution is replaced by P_{best} [14].

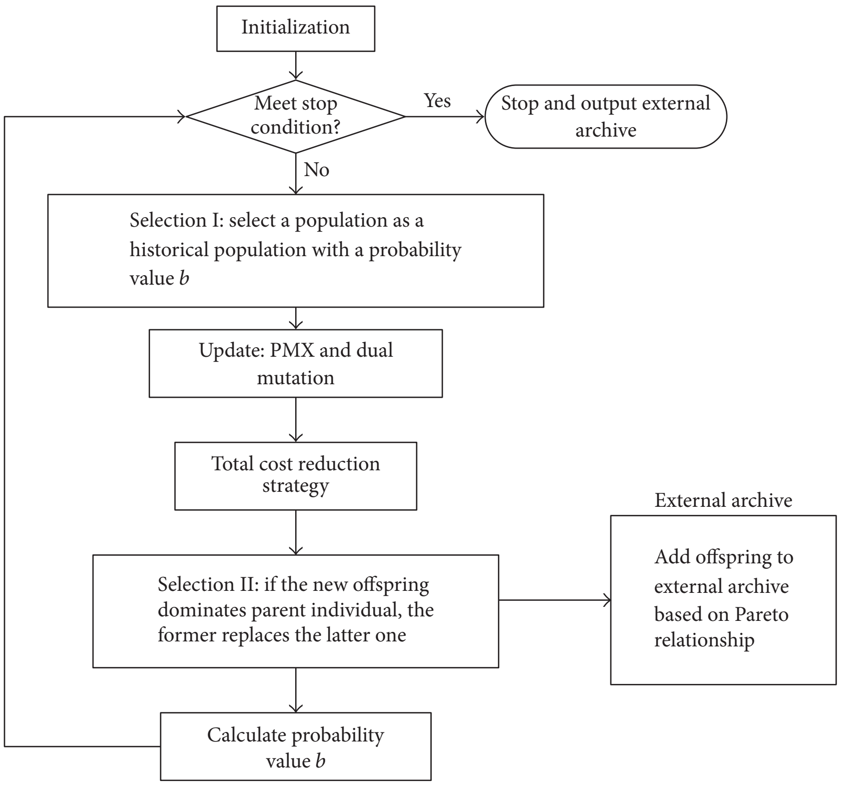

Initialization
Meet stop condition?
Yes
Stop and output external archive
No
Selection I: select a population as a historical population with a probability value b
Update: PMX and dual mutation
Total cost reduction strategy
External archive
Selection II: if the new offspring dominates parent individual, the former replaces the latter one
Add offspring to external archive based on Pareto relationship
Calculate probability value b

Figure 1: Flowchart of the MODBSA.

4.3. Framework of MODBSA for SSPWCPT. The original BSA is designed for addressing continuous single-objective optimization problems. However, the problem under consideration is a multiobjective combinatorial NP-hard issue. This paper is aimed at developing a MODBSA to solve SSPWCPT. A basic flowchart of MODBSA is shown in Figure 1.

The framework of the MODBSA consists of 6 steps summarized below.

Input

(i) A stopping condition

(ii) N: the population size

(iii) Input other parameters

Output. Pareto front (PF) and Pareto solutions (PS) are stored in an external archive.

Step 1 (initialization). Generate the initial and historical population according to the proposed solution representation stated in Section 4.4.2.

Step 2 (stopping condition). If stopping criterion is met, then stop and output PS and PF. Otherwise, go to Step 3.

Step 3 (Selection I). Calculate the selection probability value (denoted as b) of the population selected based on adaptive selection mechanism. This adaptive mechanism can select a suitable population as a historical population. Section 4.4.3 gives details of the adaptive selection mechanism.

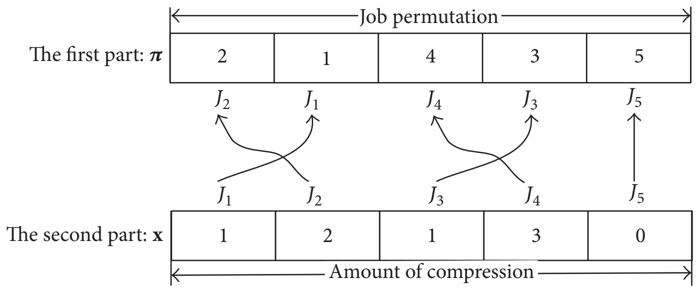
Step 4 (update). Perform update operation on the population. This update operator is described in Section 4.4.4.

Step 5 (total cost reduction strategy). The total cost reduction strategy can improve exploitation of the algorithm. This strategy is presented in Section 4.4.5.

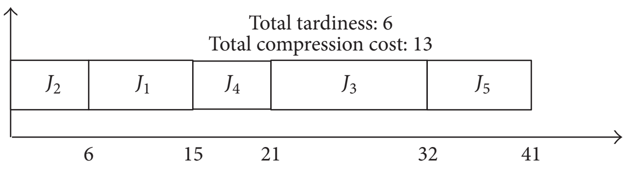
Step 6 (Selection II). This stage of the MODBSA is different from that of the basic BSA. The offspring solutions are evaluated with regard to both fitness values (i.e., total tardiness and total compression cost). First, all the individuals in the population are sorted according to a nondominated sorting technique. This fast nondominated sorting method can be described as follows. First, for each solution we compute two entities: (1) domination count n_p, the number of solutions that dominate the solution p, and (2) S_p, a set of solutions that the solution p dominates. The domination count of all solutions in the first nondominated level is equal to zero. Now, for each solution p with $n_p = 0$, we visit each individual q from its set S_p and reduce its domination count by one. When for any individual q the domination count becomes zero, it will be put into a separate set Q where the individuals belong to the second nondominated level. This procedure continues until all nondominated levels are identified. That is, each individual has a rank equal to its nondominance level. Then, within each front or rank a crowding distance strategy

Parameters	Job				
	J_1	J_2	J_3	J_4	J_5
Normal processing time (p)	10	8	12	9	9
Maximum amount of compression (m)	5	2	3	3	2
Due time (d)	22	30	40	26	35
Unit cost of compression (c)	0.5	1.0	3.0	2.5	2.0

(a) Instance



(b) Encoding



(c) Decoding Gantt chart

Figure 2: An example of solution representation.

is used to define an ordering among individuals. To achieve wide spread Pareto fronts (solutions differently balancing total tardiness and total compression cost), individuals with a large crowding distance are better than ones with a smaller crowding distance when they are the same nondominated level. The external archive is used to store nondominated solutions found so far. This external archive has a maximum size. To obtain PF with a uniform spread, the crowding distance technique is also employed to remove solutions with the worst crowding distance from the archive when this archive is full.

4.4. The Main Improvements for the Proposed MODBSA on SSPWCPT. In this subsection, a new solution representation is stated and the main improvement strategies are elaborated.

4.4.1. Solution Representation. One of the important issues when applying MOEA lies in its solution representation where individuals include information associated with the problem considered. Unlike the other scheduling problems with fixed processing times, a SSPWCPT has to deal with the job sequence and the amount of compression processing time of jobs simultaneously. Thus, we propose a new encoding scheme which contains two parts. The first part represents the job permutation, that is, π. The second one denotes the amount of compression processing times of jobs, that is, $\mathbf{x}$. Although Nearchou [8] presented a two-part encoding for this problem, he adopted subrange keys based on random key encoding scheme, which generates information redundancy. We used this proposed scheme to effectively avoid information redundancy.

To illustrate this solution representation, Figure 2 gives an example of a solution representation for a 5-job instance. This solution has two parts: (1) the first part: the job permutation, namely, $\pi = (2, 1, 4, 3, 5)$; (2) the second one: the amount of compression processing times of job, namely, $\mathbf{x} = (1, 2, 1, 3, 0)$. There exists a corresponding relationship between the two parts. Each job has a corresponding amount of compression processing time. For the $\pi(i)$ job denoted by J_k, its corresponding amount of compression processing time is x_k. For example, the 1st job (i.e., $\pi(1)$) in the sequence is job J_2, the amount of compression of J_2 is x_2 (i.e., 2). Similarly, the 2nd job (i.e., $\pi(2)$) in the sequence is J_1 and the corresponding amount of compression of J_1 is x_1 (i.e., 1). Therefore, the corresponding amount vector of compression is $(2, 1, 3, 1, 0)$ for a sequence π. In this manner, a feasible schedule is easily generated. In addition, this encoding scheme has two advantages below:

(1) Solution structure is simple as it contains two parts, namely, job sequence and amount of compression of job processing times.

(2) In general, this discrete encoding scheme can reduce information redundancy compared with previous

random keys encoding scheme like subrange keys [8]. This view has been further certified in Section 5.4.

4.4.2. Initialization. The MODBSA begins with a population of N initial individuals. As stated previously, each solution is constructed by a permutation vector and a vector of compression processing times. To ensure the high quality and good diversity of solutions, the job permutation of one individual is based on nondescending order of due time [33]. In detail, this solution is based on the property that if we have job l and job m that satisfy $p_l < p_m$ and $d_l < d_m$ $(l, m = 1, 2, \ldots, n)$ then there exists an optimal processing sequence where job l precedes job m. The amount of compression processing times of another individual is set to zero in order to minimize total compression cost. The other initial individuals are randomly generated in the feasible range as follows:

$$P_{i,j} = \begin{pmatrix} \pi(i,1), \pi(i,2), \ldots, \pi(i,n) \\ x(i,1), x(i,2), \ldots, x(i,n) \end{pmatrix}, \tag{12}$$

where $\pi(i,n)$ denotes the nth job in the ith individual to be processed on the machine and $x(i,n)$ presents the amount of compression for job n in the ith individual.

4.4.3. Selection I Based on the Adaptive Mechanism. The dual population scheme is a core idea of the BSA. The dual population is based on a random selection scheme, which can assist algorithm to maintain population diversity. However, this strategy cannot ensure a good convergence toward the optimal solutions, since too much emphasis on diversity would cause pure random search [15]. To improve convergence performance, an adaptive selection scheme is presented. This adaptive selection scheme can select a suitable population as the current historical population. To simplify the calculation of the selection probability of population P as the current historical population, we record the population P that is chosen to participate in update operation. After the initial population and historical population are generated, a population is selected as the current historical population based on (10) so that each population has an equal selection probability. Afterwards, the current historical population is updated by an adaptive selection scheme during the search process. Let bs (bs is not a random number at this time) represent the probability of updating the historical population old P by replacing it with the current population P. The adaptive selection scheme can be stated as follows.

Step 1. After population P is updated, calculate the update probability of historical population old P; namely, $bs = np/|A|$, where np is the number of nondominated solutions and $|A|$ represents the total number of the nondominated solutions in the external archive at the current iteration. The updating of old P is executed by simply replacing the old P with the current population P.

Step 2. Use the roulette-wheel approach to select a population from each population.

This selection strategy is simple yet efficient for improving performance of MODBSA. It implies that the selection probability is proportional to the number of nondominated solutions from the population P. To avoid the situation where all solutions are obtained from the same population throughout the iterations, the population P has a minimum selection probability of $b_{\min}$. That is, after the calculation of the selection probability of population, if $bs < bs_{\min}$, then set $bs = bs_{\min}$. In this work, $bs_{\min} = 0.2$.

Before updating population, select two parents in which one is from the current old P (note that the current old P is either from old P or from P according to the adaptive mechanism) and the other is from P. In addition, the efficiency of this adaptive scheme has been proven in Section 5.5.

4.4.4. Update Operator. It is evident that crossover and mutation operators of original BSA are not suitable for solving the SSPWCPT. To overcome this problem, we change the crossover and mutate operators to the traditional crossover and mutation operators.

Crossover can explore unknown areas of solution search space. For the first part (i.e., π), partially mapped crossover (PMX) [34] is adopted to update the permutation part and has been widely used in the scheduling field. For the second part, two-point crossover is used to update amount of compression processing times in this paper. The detailed steps of the crossover are as follows.

For the first part (see Figure 3), consider the following.

Step 1. Select the substring: randomly generate two crossover points and define the substring between two points as matching area (i.e., yellow area).

Step 2. Exchange the substrings: generate temporary offspring by swapping the matching area of two parents. Note that only the elements in the matching area (yellow area) are exchanged. We can find that temporary offspring is unfeasible when exchanging the substrings. For instance, job 5 appears twice in temporary offspring 1. Similarly, job 1 appears twice in temporary offspring 2.

Step 3. Mapped relationship: determine mapped relationship of the elements in conflict. When the same job sequence is assigned more than once, the mapped relationship of the sequence in crossover segments is defined.

Step 4. Legalize the offspring: make the job permutation part feasible by using the information from the mapped relationship without any changes to the substring (keep the substring unchanged).

For the second part (see Figure 4), consider the following.

Step 1. Select the substring: randomly generate two points and define the substring between two points as exchange area.

Step 2. Exchange the substring: exchange the substrings between the two points.

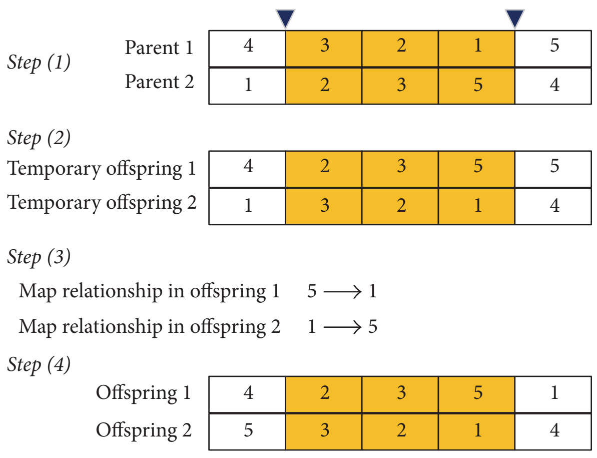


FIGURE 3: PMX crossover in the first part.

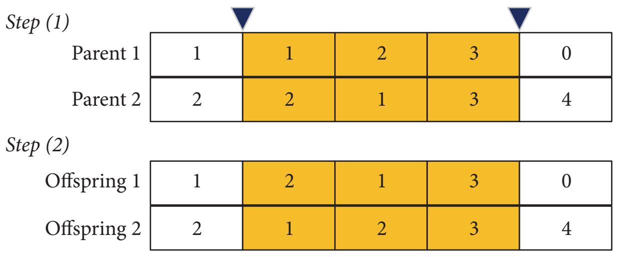


FIGURE 4: Two-point crossover in the second part.

Mutation operator assists the algorithm to escape from local optima. In this research, the mutation operator is composed of two mutation techniques with the probability of 0.5, respectively. That is, when this mutation operation is performed, one of the two techniques is executed. Therefore, the algorithm performs either the first technique or the second technique.

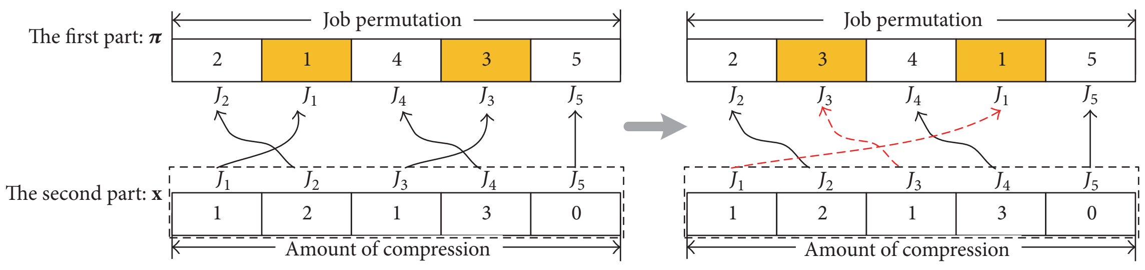
The first technique called swap mutation is only applied in the permutation part (i.e., the first part). The second one is only applied in the amount of resource compression part (i.e., the second part). To explain this mutation operator, an example is illustrated in Figure 5. For the first mutation technique as presented in Figure 5(a), the original sequence of jobs is $\pi = (2,1,4,3,5)$. When the two positions are randomly selected (e.g., the 2nd and 4th positions), their corresponding job 1 and job 3 are swapped while the second part remains unchanged. That is, the sequence of jobs becomes $\pi = (2,3,4,1,5)$ after performing the first technique. For the second mutation technique as shown in Figure 5(b), firstly two positions are randomly selected (e.g., the 2nd and 4th positions), and then the corresponding new feasible integer numbers are generated to replace the original values. That is, the amount value of compression of job 2 in the 2nd position is updated to a new value 1, while the amount value of job 4 in 4th position is replaced by new value 2. These new values are randomly generated in their own ranges. Therefore, the offspring after update operator is still feasible.

4.4.5. Total Cost Reduction Strategy. Metaheuristics are usually combined with local search approaches, which may assist in searching for good solutions since they introduce an idea "greediness" within the metaheuristic [35]. In this paper, however, any local search strategies are not applied in the proposed algorithm. Instead, a release cost procedure is developed to further improve quality of solutions. This procedure does not change the job processing sequence but reduces the total compression cost while keeping the same total tardiness. According to the characteristic of the problem, it can be observed that by adjusting amount of compression processing time of job the compression cost can be further reduced without affecting total tardiness. Thus, it can improve the quality of a solution for a given job sequence to some extent. The computational complexity of this heuristic is $O(n)$. The main steps of the proposed total reduction strategy are below.

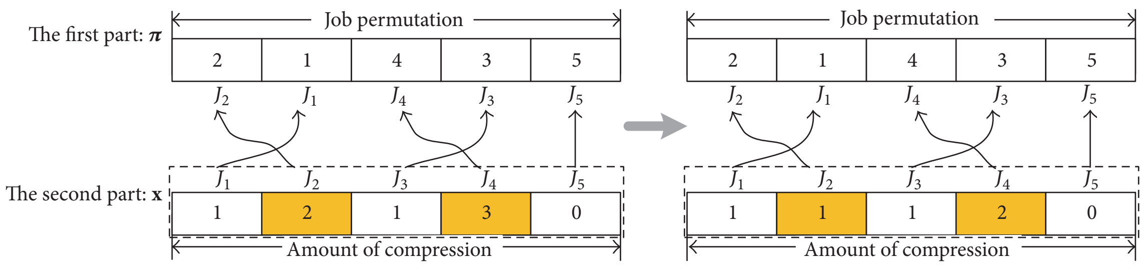
Step 1. Let $i = n$; $S_C = \emptyset$; $S_{E(j)} = \emptyset, \forall j = 1,\ldots,n$ ($\emptyset$ is an empty set). S_C is a set of release cost. $S_{E(i)}$ is earliness set in which jobs follow the $\pi(i)$ job.

Step 2. Compute $C_{\pi(i)} - d_{\pi(i)}$. If $C_{\pi(i)} - d_{\pi(i)} \geq 0$, go to Step 4. Otherwise go to Step 3.

Step 3. While $C_{\pi(i)} - d_{\pi(i)} < 0$ then perform the following loop until $C_{\pi(i)} - d_{\pi(i)} \geq 0$ or $i < 1$.


The first part: $\boldsymbol{\pi}$
Job permutation
2 1 4 3 5
J_2 J_1 J_4 J_3 J_5
The second part: $\mathbf{x}$
J_1 J_2 J_3 J_4 J_5
1 2 1 3 0
Amount of compression
Job permutation
2 3 4 1 5
J_2 J_3 J_4 J_1 J_5
1 2 1 3 0
Amount of compression

(a) The first mutation technique


The first part: $\boldsymbol{\pi}$
Job permutation
2 1 4 3 5
J_2 J_1 J_4 J_3 J_5
The second part: $\mathbf{x}$
J_1 J_2 J_3 J_4 J_5
1 2 1 3 0
Amount of compression
Job permutation
2 1 4 3 5
1 1 1 2 0
Amount of compression

(b) The second mutation technique

Figure 5: Mutation operator.

Table 1: Data of Example 1.

π	2 (J_2)	3 (J_3)	4 (J_4)	1 (J_1)	5 (J_5)
$p_{\pi(i)}$	5	5	8	7	7
$x_{\pi(i)}$	1	1	1	4	0
$p^a_{\pi(i)}$	4	4	7	3	7
$d_{\pi(i)}$	5	10	13	20	26
$C_{\pi(i)}$	4	8	15	18	25
$c_{\pi(i)}$	0.2	0.3	0.1	0.5	0.4

Step 3.1. Compute earliness time of the $\pi(i)$ job, namely, $E_{\pi(i)} = |C_{\pi(i)} - d_{\pi(i)}|$, and then update the set $S_{E(i)} = \bigcup_{j=i}^{n} S_{E(j)} \bigcup \{E_{\pi(i)}\}$.

Step 3.2. Obtain the current minimum earliness $mE_{\pi(i)} = \min S_{E(i)}$ and calculate release cost $\text{Cost}_{\pi(i)} = c_{\pi(i)} \cdot \min\{mE_{\pi(i)}, x_{\pi(i)}\}$, and then put the value $\text{Cost}_{\pi(i)}$ into the set S_C, $i = i - 1$.

Step 4. Find the job with the maximum value Cost from S_C if S_C is not an empty set. This job is denoted by $J_{\pi(k)}$. Update its amount of compression processing time $x_{\pi(k)} = x_{\pi(k)} - \min\{mE_{\pi(k)}, x_{\pi(k)}\}$ and the completion time of jobs following the $\pi(k)$ job.

To explain this total reduction strategy, an example is given below.

Example 1. Consider a 5-job instance in Table 1 and Figure 6 with a given job sequence $\pi = (2, 3, 4, 1, 5)$ and amount vector of compression processing time $\mathbf{x} = (4, 1, 1, 1, 0)$. The corresponding total tardiness and total compression cost are 2 and 2.6, respectively. Perform the total reduction strategy in the following steps.

Step 1. Let $i = 5$; $S_C = \emptyset$; $S_{E(j)} = \emptyset, \forall j = 1, \ldots, 5$.

Step 2. Compute $C_{\pi(5)} - d_{\pi(5)} = 25 - 26 = -1 < 0$ and go to Step 3.

Step 3. Conduct the following operation.

Step 3.1. Compute earliness time $E_{\pi(5)} = |C_{\pi(5)} - d_{\pi(5)}| = 1$ and $S_{E(5)} = \{1\}$.

Step 3.2. Obtain the minimum earliness found so far $mE_{\pi(5)} = \min S_{E(5)} = 1$; compute the release cost $\text{Cost}_{\pi(5)} = c_{\pi(5)} \cdot \min\{mE_{\pi(5)}, x_{\pi(5)}\} = 0.4 \times 0 = 0$; put the value $\text{Cost}_{\pi(i)}$ into the set $S_c = \{0\}$; $i = i - 1$.

Step 2. Compute $C_{\pi(4)} - d_{\pi(4)} = -2 < 0$ and go to Step 3.

Step 3. Conduct the following operation.

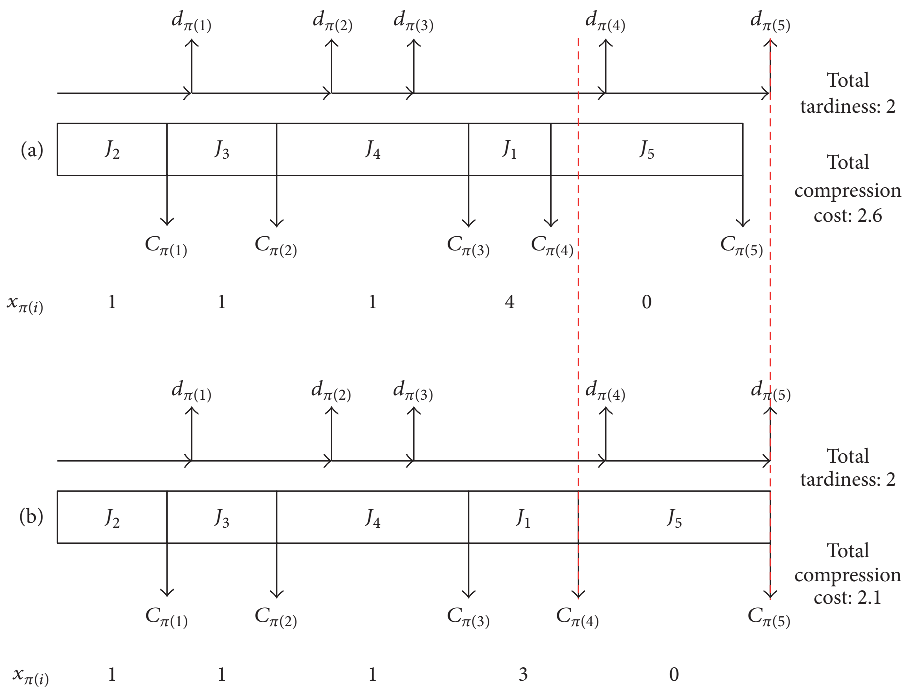


Figure 6: Gantt chart of Example 1 before (a) and after (b) using the total cost reduction strategy.

Step 3.1. $E_{\pi(4)} = |C_{\pi(4)} - d_{\pi(4)}| = 2$ and $S_{E(4)} = S_{E(5)} \cup \{E_{\pi(4)}\} = \{1, 2\}$.

Step 3.2. $mE_{\pi(4)} = \min S_{E(4)} = 1$; $\text{Cost}_{\pi(4)} = c_{\pi(4)} \cdot \min\{1, 4\} = 0.5 \times 1 = 0.5$; $i = i - 1$; $S_c = \{0, 0.5\}$.

Step 2. Compute $C_{\pi(3)} - d_{\pi(3)} = 2 > 0$ and go to Step 4.

Step 4. Select the job with the maximum cost value from S_c, namely, the $\pi(4)$ job. Update the amount of processing time compressed: $x_{\pi(4)} = x_{\pi(4)} - \min\{mE_{\pi(4)}, x_{\pi(4)}\} = 4 - \min\{1, 4\} = 3$. Meanwhile, the corresponding total tardiness and total compression cost are updated to 2 and 2.1, respectively. Note that the total tardiness is fixed but the total compression cost is reduced.

5. Experimental Study

This section is devoted to assessing the performance of the proposed algorithm MODBSA on SSPWCPT. The experimental studies include the following four aspects:

(1) Evaluate efficiency of the proposed solution representation.

(2) Evaluate efficiency of the adaptive selection scheme in the MODBSA.

(3) Test performance on the total cost reduction strategy of the MODBSA.

(4) Compare the MODBSA with other MOEAs on the instances.

In the following subsections, performance metrics, test function, and parameter setting are described at first, and then the experimental studies are further investigated step by step.

Table 2: Parameter setting of the instance.

Input variables	Value
Number of jobs (n)	5
Normal processing time (p_i)	[10, 12, 8, 7, 6]
Maximum amount of compression for job (m_i)	[3, 4, 3, 5, 4]
Due time (d_i)	[29, 10, 15, 9, 20]
Unit cost of compression (c_i)	[0.5, 2, 2.5, 1, 1.5]

5.1. Performance Metrics

As mentioned previously, the final result is usually not a single optimal value rather than a set of optimal solutions for MOPs. To explain this problem, the parameters of a specific instance are provided in Table 2.

The PF obtained by MODBSA on this instance is shown in Figure 7. Some nondominated solutions and the corresponding objective values are summarized in Table 3. It can be observed that the obtained results consist of some trade-off solutions. It is also an interesting observation that there are two nondominated solutions that correspond to the same front point (i.e., solution 2). Unlike single-objective problem, the high quality results of MOP not only have good convergence but also evenly distribution along PF*. Therefore, how to evaluate results found by MOEAs is important for users.

To evaluate the results obtained by MOEAs, some metrics including the Spread [36], GD, and IGD [37] should be adopted as follows.

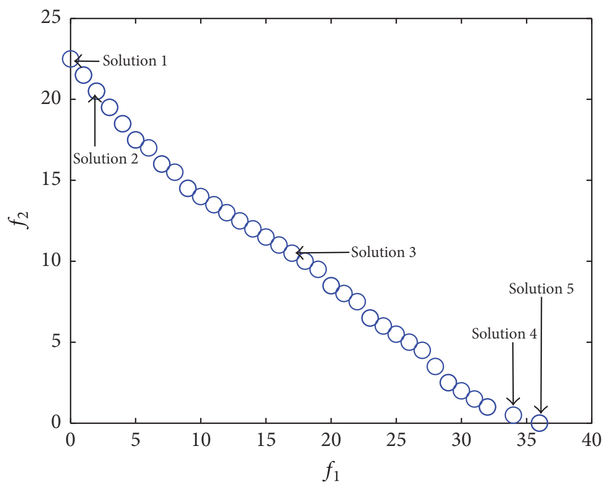


FIGURE 7: Pareto front obtained by MODBSA for this scheduling instance.

TABLE 3: The corresponding results for the instance in Figure 7.

Number	Solution	Detailed information		f_1	f_2
1	$[4, 2, 3, 5, 1]$	Job sequence π	$[J_4, J_2, J_3, J_5, J_1]$	0	22.5
	$[1, 4, 3, 5, 1]$	Amount of compression $\mathbf{x}_\pi$	$[5, 4, 3, 1, 1]$		
2	$[4, 2, 3, 5, 1]$	Job sequence π	$[J_4, J_2, J_3, J_5, J_1]$	2.0	20.5
	$[2, 4, 2, 5, 1]$	Amount of compression $\mathbf{x}_\pi$	$[5, 4, 2, 1, 2]$		
2	$[4, 2, 3, 5, 1]$	Job sequence π	$[J_4, J_2, J_3, J_5, J_1]$	2.0	20.5
	$[1, 4, 1, 5, 3]$	Amount of compression $\mathbf{x}_\pi$	$[5, 4, 1, 3, 1]$		
3	$[4, 2, 3, 5, 1]$	Job sequence π	$[J_4, J_2, J_3, J_5, J_1]$	17.0	10.5
	$[3, 2, 0, 5, 0]$	Amount of compression $\mathbf{x}_\pi$	$[5, 2, 0, 0, 3]$		
4	$[4, 3, 5, 1, 2]$	Job sequence π	$[J_4, J_3, J_5, J_1, J_2]$	34.0	0.5
	$[1, 0, 0, 0, 0]$	Amount of compression $\mathbf{x}_\pi$	$[0, 0, 0, 1, 0]$		
5	$[4, 3, 5, 1, 2]$	Job sequence π	$[J_4, J_3, J_5, J_1, J_2]$	36.0	0.0
	$[0, 0, 0, 0, 0]$	Amount of compression $\mathbf{x}_\pi$	$[0, 0, 0, 0, 0]$		

(1) Spread (Δ). It is a diversity performance index that assesses the distribution of the obtained solutions in the front. This metric can be formulated as follows:

$$\Delta = \frac{\sum_{j=1}^{m} d_j^e + \sum_{i=1}^{|\mathrm{PF}|} \left| d_i - \overline{d} \right|}{\sum_{j=1}^{m} d_j^e + |\mathrm{PF}| \cdot \overline{d}}, \tag{13}$$

where d_i is the Euclidean distance of each point in PF to its closest point in PF, $\overline{d}$ represents the mean value of all d_i, d_j^e denotes the Euclidean distance between the extreme solutions in the jth objective and the boundary solutions of the PF*, |PF| is the number of PF, and m is the number of objectives. If the spread value is zero, then all the members of Pareto optimal front are evenly spaced. Lower values indicate better distribution and diversity.

(2) Generational Distance (GD). It is a convergence indicator, which represents how far the obtained PF is from PF*. It can be formulated as follows:

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{|\mathrm{PF}|} D_i^2}}{|\mathrm{PF}|}, \tag{14}$$

where |PF| means the number of PF points and D_i is the Euclidean distance between the ith member of PF obtained and the nearest member of PF*. A low GD value represents a good convergence performance. A normalization method is used in this metric.

(3) Inverse Generational Distance (IGD). It is a variant of the GD but represents a combined or comprehensive indicator. It measures the average distance between each solution consisting of the optimal Pareto front and obtained front. IGD can be defined as follows:

$$\mathrm{IGD} = \frac{\sum_{x \in \mathrm{PF}^*} \mathrm{dist}(x, \mathrm{PF})}{|\mathrm{PF}^*|}, \tag{15}$$

where $|\mathrm{PF}^*|$ is the number of the optimal Pareto fronts; $\mathrm{dist}(x, \mathrm{PF})$ is the Euclidean distance between x and the nearest member of the approximation. Fronts with a lower IGD value are desirable. This metric uses a normalization method.

It should be mentioned that the true PF* of the studied problem may be unknown; therefore, the nondominated solutions obtained by different MOEAs on each instance in

Table 4: Data set distribution.

Input variables	Distribution
Number of jobs (n)	10, 30, 50, 80, 100, 200
Normal processing time (p_i)	DU(20, 50)
Crash processing time (p_i^c)	DU($0.5 * p_i, p_i$)
Due time (d_i)	$d_i = P(1-r) + (P/1.1 + P(1-r))/(n-1)$
Unit cost of compression (c_i)	$U(0.5, 2.5)$

Table 5: Mean and standard deviation value of all metrics between MODBSA$_{\text{sk}}$ and MODBSA.

Problem	GD (mean/std)		Spread (mean/std)		IGD (mean/std)	
	MODBSA$_{\text{sk}}$	MODBSA	MODBSA$_{\text{sk}}$	MODBSA	MODBSA$_{\text{sk}}$	MODBSA
50_02	8.76e−03/9.9e−03	**8.41e−03/1.1e−02**	5.83e−01/3.1e−01	**5.61e−01/3.3e−01**	3.02e−03/5.6e−04	**6.16e−04/1.5e−04**
50_04	**1.73e−02/1.6e−02**	1.87e−02/2.6e−02	**7.15e−01/4.1e−01**	7.79e−01/3.7e−01	3.01e−03/6.1e−04	**8.38e−04/3.6e−04**
50_06	3.32e−02/4.8e−02	**3.23e−02/5.5e−02**	7.41e−01/4.1e−01	**7.38e−01/4.7e−01**	3.29e−03/7.3e−04	**9.67e−04/4.2e−04**
50_08	**3.56e−02/4.8e−02**	3.77e−02/5.7e−02	**7.83e−01/4.0e−01**	7.87e−01/5.1e−01	3.83e−03/1.2e−03	**6.02e−04/2.0e−04**
50_10	**2.97e−02/5.0e−02**	3.06e−02/3.5e−02	8.68e−01/1.8e−01	**6.86e−01/4.7e−01**	8.36e−03/2.4e−03	**9.29e−04/4.1e−04**
80_02	1.68e−02/1.4e−02	**1.00e−02/1.3e−02**	**6.41e−01/3.2e−01**	7.75e−01/3.4e−01	3.85e−03/5.7e−04	**7.88e−04/4.2e−04**
80_04	3.28e−02/3.2e−02	**1.47e−02/1.7e−02**	**7.38e−01/3.2e−01**	9.40e−01/3.8e−01	3.92e−03/8.9e−04	**7.99e−04/3.3e−04**
80_06	5.48e−02/6.0e−02	**2.65e−02/4.3e−02**	**6.74e−01/3.6e−01**	9.91e−01/4.0e−01	5.16e−03/1.1e−03	**6.20e−04/2.1e−04**
80_08	7.41e−02/6.8e−02	**5.24e−02/6.3e−02**	**9.12e−01/3.5e−01**	1.07e+00/4.3e−01	6.95e−03/2.1e−03	**6.87e−04/2.4e−04**
80_10	**3.67e−02/5.1e−02**	8.90e−02/5.8e−02	9.65e−01/1.7e−01	**8.50e−01/4.5e−01**	2.18e−02/5.3e−03	**7.46e−04/3.6e−04**
100_02	1.54e−02/1.3e−02	**6.98e−03/6.5e−03**	**5.88e−01/2.4e−01**	7.65e−01/3.3e−01	5.07e−03/6.2e−04	**1.21e−03/4.3e−04**
100_04	3.67e−02/3.9e−02	**2.13e−02/2.2e−02**	**8.31e−01/3.5e−01**	9.09e−01/4.1e−01	4.76e−03/1.0e−03	**7.09e−04/2.6e−04**
100_06	5.17e−02/5.5e−02	**3.79e−02/4.3e−02**	**8.85e−01/3.3e−01**	9.78e−01/4.1e−01	7.93e−03/2.1e−03	**8.36e−04/3.7e−04**
100_08	7.68e−02/8.1e−02	**6.26e−02/8.1e−02**	**9.38e−01/3.0e−01**	1.09e+00/3.7e−01	1.27e−02/3.5e−03	**6.18e−04/2.8e−04**
100_10	**4.30e−02/4.9e−02**	1.30e−01/7.7e−02	9.67e−01/1.4e−01	**9.19e−01/4.2e−01**	3.44e−02/6.8e−03	**6.72e−04/2.6e−04**
200_02	1.94e−02/1.3e−02	**1.36e−02/6.5e−03**	**7.96e−01/2.4e−01**	8.76e−01/3.0e−01	7.92e−03/5.8e−04	**8.33e−04/3.0e−04**
200_04	6.54e−02/4.1e−02	**3.74e−02/2.7e−02**	**8.81e−01/2.9e−01**	1.18e+00/2.3e−01	1.61e−02/2.3e−03	**9.47e−04/3.2e−04**
200_06	1.11e−01/8.9e−02	**7.63e−02/6.2e−02**	**9.91e−01/2.4e−01**	1.19e+00/3.3e−01	2.37e−02/4.3e−03	**1.02e−03/3.3e−04**
200_08	**8.46e−02/6.5e−02**	1.89e−01/1.0e−01	**1.01e+00/1.7e−01**	1.20e+00/2.5e−01	5.24e−02/9.8e−03	**5.96e−04/2.5e−04**
200_10	**8.61e−02/9.3e−02**	4.74e−01/1.1e−01	**9.60e−01/7.2e−02**	1.02e+00/4.2e−01	1.58e−01/2.0e−02	**7.12e−04/2.2e−04**
Hit rate	7/20	13/20	15/20	5/20	0/20	20/20

all the independent runs are regarded as PF* on that instance [38].

5.2. *Description of Test Function.*

The instances generated are defined as shown in Table 4. There are six different numbers of jobs ($n = 10, 30, 50, 80, 100, 200$), where the normal processing times and the crash processing times are drawn from the discrete uniform distributions DU(20, 100) and DU($0.5 \cdot p_i, p_i$), respectively. In due date calculation, $P = \sum_{i=1}^{n} p_i$, and r is the discrete value from 0.2 to 1.0 with the step size of 0.2. The unit cost of compression is generated from a uniform distribution ranging between 0.5 and 2.5. Each instance can be labelled in the form of "n_r". For example, "10_02" represents the fact that the problem is featured by 10 jobs and r equal to 0.2.

5.3. *Experimental Settings.*

All algorithms are coded in Java on the platform jMetal [39]. Experimental tests are implemented on a computer with Intel Core i5, 2.39 GHz, 4 GB RAM, with a Windows 8 operating system.

Parameter settings can affect the performance of the algorithm. The pilot experiments demonstrated that the population size and archive size were sensitive to the problem scale. Therefore, for various variants of MODBSA in Sections 5.4–5.6, the maximum number of function evaluations (NFEs) is 25,000 for 10-job and 30-job instances, 35,000 for 50-job and 80-job instances, and 45,000 for 100-job and 200-job instances. The population size and the external archive size are set to 50 for 10-job and 30-job instances, 80 for 50-job and 80-job instances, and 100 for 100-job and 200-job instances. The historical population size is equal to the population size. In Section 5.7, the parameter

Table 6: Wilcoxon signed rank test results based on the best metrics for each instance with 30 independent runs (a level of significant $\alpha = 0.05$).

Number	GD			Δ			IGD		
	MODBSA versus MODBSA$_{sk}$			MODBSA versus MODBSA$_{sk}$			MODBSA versus MODBSA$_{sk}$		
	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win
50_02	243	222	$8.29e-01/=$	256	209	$6.29e-01/=$	465	0	$1.73e-06/+$
50_04	231	234	$9.75e-01/=$	217	248	$7.50e-01/=$	465	0	$1.73e-06/+$
50_06	248	217	$7.50e-01/=$	229	326	$9.43e-01/=$	465	0	$1.73e-06/+$
50_08	222	243	$8.29e-01/=$	326	229	$9.43e-01/=$	465	0	$1.73e-06/+$
50_10	195	270	$4.41e-01/=$	326	139	$5.45e-02/=$	465	0	$1.73e-06/+$
80_02	336	129	$3.33e-02/+$	155	310	$1.11e-01/=$	465	0	$1.73e-06/+$
80_04	369	96	$5.00e-03/+$	119	346	$1.96e-02/-$	465	0	$1.73e-06/+$
80_06	349	116	$1.66e-02/+$	98	367	$5.70e-02/-$	465	0	$1.73e-06/+$
80_08	446	19	$1.13e-05/+$	157	308	$1.20e-01/=$	465	0	$1.73e-06/+$
80_10	108	357	$1.04e-02/-$	281	184	$3.19e-01/=$	465	0	$1.73e-06/+$
100_02	465	0	$1.73e-06/+$	117	348	$1.75e-02/-$	465	0	$1.73e-06/+$
100_04	396	69	$7.71e-04/+$	190	275	$3.82e-01/=$	465	0	$1.73e-06/+$
100_06	431	34	$4.45e-05/+$	175	325	$5.71e-02/=$	465	0	$1.73e-06/+$
100_08	432	33	$4.07e-05/+$	140	325	$5.71e-02/-$	465	0	$1.73e-06/+$
100_10	198	267	$4.78e-01/=$	246	219	$7.81e-01/=$	465	0	$1.73e-06/+$
200_02	325	140	$5.71e-02/=$	164	301	$1.59e-01/=$	465	0	$1.73e-06/+$
200_04	465	0	$1.73e-06/+$	73	392	$1.01e-03/-$	465	0	$1.73e-06/+$
200_06	389	76	$1.29e-03/+$	98	367	$5.71e-03/-$	465	0	$1.73e-06/+$
200_08	31	434	$3.41e-05/-$	80	385	$1.70e-03/-$	465	0	$1.73e-06/+$
200_10	0	465	$1.73e-06/-$	0	465	$1.73e-06/-$	465	0	$1.73e-06/+$
$+/=/-$		10/7/3			0/12/8			20/0/0	

settings of MODBSA and its compared MOEAs can be found in related subsection. Each experiment was conducted 30 independent times on each test problem for each algorithm.

The optimal results are highlighted with *bold* in Tables 5, 7, 9, 12, 13, and 14. Due to the stochastic characteristic of all candidate MOEAs, the statistical analysis is necessary to provide confidential comparisons. A Wilcoxon sign rank test [38, 40] is used to test the significant difference between the results obtained by different algorithms. The confidence level for all tests is set to 95% (corresponding to $\alpha = 0.05$). The sign "+" indicates that our proposed MODBSA algorithm performs significantly better than the second best algorithm on average. While "−" represents the fact that the MODBSA algorithm is significantly worse than the best algorithm, the "=" sign denotes that there is no significant difference between MODBSA and the best or second best MOEA. R^+ represents the sum of ranks for the problem where the MODBSA performs better than its competitor. R^- denotes the sum of ranks for the opposite.

5.4. Efficiency of Solution Representation. To test the efficiency of the proposed solution representation, it is compared with subrange keys for MODBSA on medium and large scale problems. In this study, MODBSA$_{sk}$ represents MODBSA based on subrange keys. More detailed information on subrange keys can be found in Nearchou [8]. Since subrange key is a real-coded scheme, the operators in MODBSA$_{sk}$ are different from that in MODBSA. The update operators of MODBSA$_{sk}$ are as follows: the simulated binary crossover (SBX) and polynomial mutation are used. The distribution indexes in both SBX and the polynomial mutation are set to 20. The crossover rate is 0.9, and mutation rate is 0.2. In addition, the proposed total cost reduction strategy is also adopted in MODBSA$_{sk}$, but MODBSA$_{sk}$ requires converting real-coded scheme to discrete-coded scheme in this phase. MODBSA includes the proposed solution representation. The crossover rate is 0.9 and mutation rate is 0.2. Table 5 shows the mean and standard deviation metrics on these algorithms over 30 independent runs. Table 6 reports the significant test results over 30 runs.

Table 5 reveals that MODBSA obtains the optimal results on 13, 5, and 20 out of 20 test instances for GD, Spread, and IGD metrics, while MODBSA$_{sk}$ achieves the best values on 7, 15, and 0 problems, respectively. Table 6 records the p values of the Wilcoxon signed rank test. We can clearly observe from Table 6 that MODBSA has higher "+" counts than its compared algorithm in terms of GD and IGD. It means MODBSA is significantly better than MODBSA$_{sk}$ for GD and IGD metrics. This may be because the proposed discrete

Table 7: Mean and standard deviation value of all metrics between MODBSA$_{rs}$ and MODBSA.

Problem	GD (mean/std)		Spread (mean/std)		IGD (mean/std)	
	MODBSA$_{rs}$	MODBSA	MODBSA$_{rs}$	MODBSA	MODBSA$_{rs}$	MODBSA
50_02	7.58e−03/7.1e−03	**1.61e − 03/2.2e − 03**	**2.79e − 01/1.5e − 01**	5.76e−01/2.9e−01	**6.41e − 04/1.7e − 04**	7.36e−04/2.2e−04
50_04	1.68e−02/2.8e−02	**1.82e − 03/2.0e − 03**	**3.05e − 01/1.3e − 01**	6.61e−01/3.9e−01	1.11e−03/4.8e−04	**9.85e − 04/4.8e − 04**
50_06	2.91e−02/4.3e−02	**2.22e − 03/1.3e − 03**	**2.92e − 01/6.9e − 02**	7.44e−01/4.7e−01	1.35e−03/5.2e−04	**8.20e − 04/3.3e − 04**
50_08	4.67e−02/5.7e−02	**4.55e − 03/6.3e − 03**	**3.51e − 01/2.1e − 01**	7.62e−01/5.4e−01	1.29e−03/4.4e−04	**5.44e − 04/2.1e − 04**
50_10	3.75e−02/4.7e−02	**1.60e − 02/9.8e − 03**	**6.12e − 01/1.5e − 01**	8.24e−01/4.7e−01	4.42e−03/1.5e−03	**9.21e − 04/4.0e − 04**
80_02	1.16e−02/1.1e−02	**3.31e − 03/3.1e − 03**	**3.68e − 01/2.0e − 01**	6.82e−01/3.6e−01	1.03e−03/3.4e−04	**6.31e − 04/1.8e − 04**
80_04	3.04e−02/2.9e−02	**4.97e − 03/5.9e − 03**	**4.40e − 01/2.2e − 01**	9.35e−01/3.4e−01	2.64e−03/9.9e−04	**1.03e − 03/4.7e − 04**
80_06	4.88e−02/5.3e−02	**6.37e − 03/6.5e − 03**	**4.54e − 01/2.1e − 01**	9.86e−01/4.2e−01	4.01e−03/1.7e−03	**7.12e − 04/3.1e − 04**
80_08	7.84e−02/7.2e−02	**1.17e − 02/1.2e − 02**	**5.86e − 01/2.4e − 01**	1.05e+00/4.5e−01	5.32e−03/1.8e−03	**7.04e − 04/2.9e − 04**
80_10	5.42e−02/5.5e−02	**3.34e − 02/9.3e − 03**	**7.75e − 01/1.0e − 01**	9.82e−01/4.4e−01	1.26e−02/3.0e−03	**7.21e − 04/3.7e − 04**
100_02	1.74e−02/1.2e−02	**7.22e − 03/1.0e − 02**	**5.00e − 01/2.7e − 01**	8.37e−01/3.4e−01	2.25e−03/8.6e−04	**7.75e − 04/4.1e − 04**
100_04	3.00e−02/3.2e−02	**4.99e − 03/3.6e − 03**	**5.04e − 01/1.7e − 01**	9.00e−01/4.1e−01	3.58e−03/1.0e−03	**9.49e − 04/3.0e − 04**
100_06	4.97e−02/5.7e−02	**9.95e − 03/8.7e − 03**	**6.18e − 01/2.5e − 01**	9.75e−01/3.9e−01	5.34e−03/1.5e−03	**8.68e − 04/3.4e − 04**
100_08	6.28e−02/5.9e−02	**1.42e − 02/7.6e − 03**	**6.93e − 01/1.8e − 01**	1.06e+00/3.8e−01	8.61e−03/2.1e−03	**6.33e − 04/2.5e − 04**
100_10	5.95e−02/6.8e−02	**5.51e − 02/1.5e − 02**	**8.81e − 01/9.2e − 02**	1.03e+00/3.6e−01	2.05e−02/3.1e−03	**7.15e − 04/2.8e − 04**
200_02	2.13e−02/1.7e−02	**9.63e − 03/5.7e − 03**	**7.52e − 01/2.1e − 01**	8.70e−01/3.1e−01	9.41e−03/1.7e−03	**7.14e − 04/2.9e − 04**
200_04	6.01e−02/3.9e−02	**1.27e − 02/3.5e − 03**	**7.83e − 01/1.5e − 01**	1.14e+00/2.5e−01	1.55e−02/3.2e−03	**7.55e − 04/3.3e − 04**
200_06	5.90e−02/4.0e−02	**1.38e − 02/4.7e − 03**	**8.51e − 01/1.2e − 01**	1.20e+00/2.0e−01	1.84e−02/2.9e−03	**7.90e − 04/2.3e − 04**
200_08	6.74e−02/5.7e−02	**5.91e − 02/1.9e − 02**	**9.38e − 01/9.7e − 02**	1.09e+00/3.5e−01	3.97e−02/6.3e−03	**6.31e − 04/2.2e − 04**
200_10	**9.75e − 02/7.8e − 02**	2.59e−01/4.4e−02	**9.64e − 01/5.4e − 02**	1.17e+00/3.1e−01	1.09e−01/1.1e−02	**8.00e − 04/3.0e − 04**
Hit rate	1/20	19/20	20/20	0/20	1/20	19/20

encoding scheme can avoid information redundancy during the search process compared with the encoding scheme based on subrange keys. Meanwhile, MODBSA is significantly worse than MODBSA$_{sk}$ in terms of Spread metric on most instances. The reason behind it is that the encoding scheme based on subrange keys may have a great choice to search for different areas of the search space and thus improve search diversity, although it can lead to information redundancy.

5.5. Efficiency of Adaptive Strategy. To test the efficiency of the adaptive mechanism in MODBSA, we compare the MODBSA with MODBSA based on random selection mechanism on the 20 medium and large scale problems. In this experiment, MODBSA$_{rs}$ denotes MODBSA with random selection mechanism. Note that adaptive mechanism is included in MODBSA. The other parameter settings of both MOEAs are the same for a fair comparison. Table 7 reports the statistical metrics on two strategies over 30 independent runs. Table 8 summarizes the p values of Wilcoxon signed rank test.

From Tables 7 and 8, it can be observed that MODBSA completely dominates MODBSA$_{rs}$ in terms of IGD metric

Table 8: Wilcoxon signed rank test results based on the best metrics for each instance with 30 independent runs (a level of significant $\alpha = 0.05$).

Number	GD			Δ			IGD		
	MODBSA versus MODBSA$_{rs}$			MODBSA versus MODBSA$_{rs}$			MODBSA versus MODBSA$_{rs}$		
	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win
50_02	319	146	$7.52e-03/+$	33	432	$4.07e-05/-$	160	305	$1.36e-01/=$
50_04	383	82	$1.96e-03/+$	63	402	$4.90e-04/-$	465	0	$1.73e-06/+$
50_06	370	95	$4.68e-03/+$	49	416	$1.60e-04/-$	398	67	$6.64e-04/+$
50_08	423	42	$8.91e-06/+$	105	360	$8.70e-03/-$	464	1	$1.92e-06/+$
50_10	337	128	$3.16e-02/+$	115	350	$1.57e-02/-$	465	0	$1.73e-06/+$
80_02	336	129	$3.33e-02/+$	69	396	$7.71e-04/-$	436	29	$2.84e-05/+$
80_04	369	96	$5.00e-03/+$	18	447	$1.02e-05/-$	461	4	$2.60e-06/+$
80_06	349	116	$1.66e-02/+$	43	422	$9.71e-05/-$	465	0	$1.73e-06/+$
80_08	433	32	$3.72e-05/+$	55	410	$2.61e-04/-$	465	0	$1.73e-06/+$
80_10	357	108	$1.04e-02/+$	130	335	$3.50e-02/-$	465	0	$1.73e-06/+$
100_02	438	27	$2.37e-05/+$	75	390	$1.20e-03/-$	458	7	$3.52e-06/+$
100_04	465	0	$1.73e-06/+$	54	411	$2.41e-04/-$	465	0	$1.73e-06/+$
100_06	431	34	$4.45e-05/+$	60	405	$3.88e-04/-$	465	0	$1.73e-06/+$
100_08	465	0	$1.73e-06/+$	54	411	$2.41e-04/-$	465	0	$1.73e-06/+$
100_10	198	267	$4.78e-01/=$	137	328	$4.95e-02/-$	465	0	$1.73e-06/+$
200_02	378	87	$2.77e-03/+$	157	308	$1.20e-01/=$	465	0	$1.73e-06/+$
200_04	337	128	$3.16e-02/+$	43	422	$9.71e-05/-$	465	0	$1.73e-06/+$
200_06	389	76	$1.29e-03/+$	11	454	$5.22e-06/-$	465	0	$1.73e-06/+$
200_08	434	31	$3.41e-05/+$	109	356	$1.11e-02/-$	465	0	$1.73e-06/+$
200_10	44	421	$1.06e-04/-$	77	388	$1.40e-03/-$	465	0	$1.73e-06/+$
$+/=/-$		18/1/1			0/1/19			19/1/0	

on most instances. However, such advantage will no longer exist when only Spread metric is considered. The poor distribution performance of MODBSA may be associated with the characteristic of the problem. More specifically, this type of scheduling problem may be a multimodal optimization issue which contains several optimal solutions corresponding to the same objective value (i.e., the second nondominated solution on the case in Table 3). Therefore, the distribution distance between Pareto fronts is very crowded. MODBSA can obtain better results than MODBSA$_{rs}$ in terms of GD metric. In summary, the proposed MODBSA based on adaptive selection mechanism is superior to MODBSA$_{rs}$ on most instances. This means that adaptive selection can enhance search efficiency. Besides, the results computed by the proposed MODBSA are more stable, which indicates that the adaptive selection strategy can strengthen the stability of the MODBSA.

5.6. Efficiency of Total Cost Reduction in Proposed Algorithm. To prove the efficiency of the MODBSA with the total cost reduction strategy, it is compared with MODBSA without the total cost reduction. In this experiment, MODBSA$_{ntcr}$ denotes the MODBSA without the total cost reduction strategy. MODBSA itself includes the total cost reduction strategy. The parameter settings of both MOEAs are the same as the above experiments. Table 9 records the statistical metrics on different algorithms over 30 independent runs. Table 10 shows the test results based on the best metrics for each instance with 30 independent runs.

Table 9 presents that the MODBSA is superior or competitive to MODBSA$_{ntcr}$ in terms of all metrics on most instances. From Table 10, it can be clearly observed that the MODBSA with the total cost reduction strategy has a significant better performance than the one without this strategy on most instances. It means that MODBSA using the total cost reduction strategy has good convergence and coverage performance compared with MODBSA without total cost reduction. It also implies that the exploitation ability can be improved by the adoption of the total cost reduction technique in MODBSA for solving SSPWCPT.

5.7. Comparison MODBSA with Other Algorithms. To further assess the performance of the MODBSA on these scheduling problems, MODBSA is compared with well-known MOEAs: NSGA-II [36], PAES [41], and SPEA2 [42]. To fit the characteristic of the addressed problem and make a fair comparison, we modified these considered MOEAs. All MOEAs use the same population size and the NFEs as stated in Section 5.3. Moreover, the initial population is generated based on the proposed encoding scheme and strategy for all MOEAs. All MOEAs adopt the same operators including crossover,

Table 9: Mean and standard deviation value of all metrics between MODBSA$_{\text{ntcr}}$ and MODBSA.

Problem	GD (mean/std)		Spread (mean/std)		IGD (mean/std)	
	MODBSA$_{\text{ntcr}}$	MODBSA	MODBSA$_{\text{ntcr}}$	MODBSA	MODBSA$_{\text{ntcr}}$	MODBSA
50_02	9.70e−03/1.3e−02	**9.32e−03/1.2e−02**	**5.81e−01/3.3e−01**	6.17e−01/2.9e−01	1.68e−03/4.4e−04	**1.57e−03/3.3e−04**
50_04	1.91e−02/2.9e−02	**1.18e−02/1.9e−02**	6.88e−01/4.4e−01	**5.67e−01/3.8e−01**	8.85e−04/4.8e−04	**7.82e−04/2.6e−04**
50_06	**1.81e−02/2.5e−02**	3.50e−02/4.2e−02	**7.37e−01/4.1e−01**	8.31e−01/4.7e−01	1.63e−03/5.3e−04	**1.60e−03/5.8e−04**
50_08	3.12e−02/3.5e−02	**2.58e−02/3.5e−02**	7.97e−01/4.8e−01	**7.01e−01/4.9e−01**	7.59e−04/3.9e−04	**7.07e−04/1.9e−04**
50_10	2.13e−02/2.7e−02	**1.86e−02/2.7e−02**	7.14e−01/4.5e−01	**6.82e−01/4.3e−01**	1.31e−03/4.2e−04	**1.26e−03/4.5e−04**
80_02	1.72e−02/1.2e−02	**1.23e−02/1.3e−02**	8.35e−01/3.1e−01	**6.59e−01/3.7e−01**	6.85e−04/2.5e−04	**6.60e−04/2.8e−04**
80_04	4.33e−02/3.7e−02	**3.84e−02/3.7e−02**	1.00e+00/3.9e−01	**9.13e−01/4.4e−01**	**6.50e−04/2.5e−04**	7.60e−04/3.0e−04
80_06	4.60e−02/4.1e−02	**3.82e−02/3.2e−02**	**8.56e−01/4.5e−01**	9.72e−01/4.4e−01	6.59e−04/2.8e−04	**6.15e−04/1.9e−04**
80_08	7.27e−02/6.6e−02	**5.76e−02/5.2e−02**	1.03e+00/4.5e−01	**1.00e+00/4.6e−01**	8.15e−04/3.9e−04	**7.90e−04/3.7e−04**
80_10	4.83e−02/5.1e−02	**4.18e−02/6.0e−02**	9.40e−01/4.4e−01	**8.23e−01/4.4e−01**	**8.25e−04/3.9e−04**	9.26e−04/4.5e−04
100_02	1.91e−02/1.8e−02	**1.84e−02/1.5e−02**	8.37e−01/3.2e−01	**8.30e−01/3.3e−01**	**9.26e−04/3.5e−04**	9.49e−04/3.8e−04
100_04	**4.65e−02/4.5e−02**	5.34e−02/3.9e−02	**9.70e−01/4.1e−01**	1.07e+00/3.5e−01	**9.00e−04/3.9e−04**	9.43e−04/3.7e−04
100_06	**3.39e−02/4.4e−02**	5.19e−02/5.4e−02	**7.79e−01/4.8e−01**	1.03e+00/3.8e−01	7.12e−04/2.7e−04	**6.59e−04/2.5e−04**
100_08	**6.17e−02/6.1e−02**	8.69e−02/7.6e−02	**1.02e+00/4.2e−01**	1.10e+00/4.0e−01	7.50e−04/3.0e−04	**7.31e−04/3.6e−04**
100_10	**4.67e−02/4.9e−02**	5.57e−02/5.8e−02	**9.74e−01/4.1e−01**	9.81e−01/4.3e−01	9.08e−04/3.3e−04	**7.50e−04/2.5e−04**
200_02	**2.08e−02/1.5e−02**	2.26e−02/1.5e−02	**8.70e−01/3.1e−01**	9.20e−01/2.6e−01	**7.73e−04/2.7e−04**	7.91e−04/2.8e−04
200_04	5.07e−02/4.3e−02	**4.97e−02/4.0e−02**	**1.04e+00/3.3e−01**	1.13e+00/3.5e−01	7.19e−04/3.1e−04	**6.79e−04/2.9e−04**
200_06	**8.77e−02/7.0e−02**	9.73e−02/6.8e−02	**1.19e+00/2.6e−01**	1.19e+00/3.2e−01	**8.01e−04/2.8e−04**	9.01e−04/3.4e−04
200_08	9.08e−02/6.7e−02	**8.21e−02/1.3e−01**	1.21e+00/2.7e−01	**1.11e+00/4.1e−01**	8.52e−04/3.5e−04	**8.20e−04/3.6e−04**
200_10	**7.32e−02/5.9e−02**	8.68e−02/7.6e−02	1.14e+00/2.9e−01	**1.13e+00/3.1e−01**	9.57e−04/4.2e−04	**9.49e−04/3.7e−04**
Hit rate	8/20	12/20	10/20	10/20	6/20	14/20

mutation, and total cost reduction as mentioned in this paper only if they exist corresponding operators. The other parameters are summarized in Table 11. 30 independent runs are implemented for each MOEA on each test instance.

Tables 12–14 show the statistical results of GD, Spread, and IGD. From these tables, we can observe that the proposed MODBSA outperforms its counterparts for most instances. Especially on the comprehensive metric IGD and convergence metric GD, the outperformance of the MODBSA is overwhelming except for several problems. MODBSA is also competitive to NSGA-II with regard to Spread metric. In addition, from Table 15, MODBSA shows a significant improvement over the other MOEAs with a level of significance $\alpha = 0.05$ in terms of IGD metric. The major reasons for the good performance of the MODBSA can be explained as follows. First, the adoption of the dual population strategy can improve the diversity of population since different population may have different search directions, and thus the MODBSA can maintain a good diversity in search space. Second, to boost convergence performance, the adaptive selection mechanism can select an appropriate population as parent population for generating new candidate solutions

Table 10: Wilcoxon signed rank test results based on the best metrics for each instance with 30 independent runs (a level of significant $\alpha = 0.05$).

Number	GD			Δ			IGD		
	MODBSA versus MODBSA$_{\text{ntcr}}$			MODBSA versus MODBSA$_{\text{ntcr}}$			MODBSA versus MODBSA$_{\text{ntcr}}$		
	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win
50_02	243	222	$8.29e-01/=$	33	432	$4.07e-05/-$	305	160	$1.36e-01/=$
50_04	234	231	$9.75e-01/=$	402	63	$4.90e-04/+$	398	67	$6.64e-04/+$
50_06	94	371	$4.38e-03/-$	49	416	$1.60e-04/-$	293	172	$2.13e-01/=$
50_08	464	1	$1.92e-06/+$	360	105	$8.70e-03/+$	464	1	$1.92e-06/+$
50_10	434	31	$3.41e-05/+$	350	115	$1.57e-02/+$	465	0	$1.73e-06/+$
80_02	336	129	$3.33e-02/+$	396	69	$7.71e-04/+$	436	29	$2.84e-05/+$
80_04	369	96	$5.00e-03/+$	447	18	$1.02e-05/+$	4	461	$2.60e-06/-$
80_06	349	116	$1.66e-02/+$	43	422	$9.71e-05/-$	396	69	$7.71e-04/+$
80_08	465	0	$1.73e-06/+$	269	196	$4.53e-01/=$	465	0	$1.73e-06/+$
80_10	357	108	$1.04e-02/+$	335	130	$3.50e-02/+$	0	465	$1.73e-06/-$
100_02	257	208	$6.14e-01/=$	257	208	$6.14e-01/=$	140	325	$5.71e-02/=$
100_04	176	289	$2.45e-01/=$	54	411	$2.41e-04/-$	159	306	$1.31e-01/=$
100_06	34	431	$4.45e-05/-$	60	405	$3.88e-04/-$	465	0	$1.73e-06/+$
100_08	57	408	$3.07e-04/-$	54	411	$2.41e-04/-$	458	7	$3.52e-06/+$
100_10	198	267	$4.78e-01/=$	137	328	$4.95e-02/-$	465	0	$1.73e-06/+$
200_02	140	325	$5.71e-02/=$	157	308	$1.20e-01/=$	140	325	$5.71e-02/=$
200_04	300	165	$1.65e-01/+$	43	422	$9.71e-05/-$	465	0	$1.73e-06/+$
200_06	76	389	$1.29e-03/-$	257	208	$6.14e-01/=$	0	465	$1.73e-06/+$
200_08	434	31	$3.41e-05/+$	275	190	$3.82e-01/=$	465	0	$1.73e-06/+$
200_10	0	465	$1.73e-06/-$	279	186	$3.39e-01/=$	465	0	$1.73e-06/+$
$+/=/-$		9/6/5			6/6/8			13/5/2	

Table 11: The other parameter settings of MODBSA, NSGA-II, SPEA2, and PAES.

MODBSA	NSGA-II	SPEA2	PAES
Population size and archive size: 50 (10 and 30 jobs)	Population size: 50 (10 and 30 jobs)	Population size and archive size: 50 (10 and 30 jobs)	Population size and archive size: 50 (10 and 30 jobs)
Population size and archive size: 80 (50 and 80 jobs)	Population size: 80 (50 and 80 jobs)	Population size and archive size: 80 (50 and 80 jobs)	Population size and archive size: 80 (50 and 80 jobs)
Population size and archive size: 100 (100 and 200 jobs)	Population size: 100 (100 and 200 jobs)	Population size and archive size: 100 (100 and 200 jobs)	Population size and archive size: 100 (100 and 200 jobs)
Crossover rate: 0.9	Crossover rate: 0.9	Crossover rate: 0.9	Mutation rate: 0.2
Mutation rate: 0.2	Mutation rate: 0.2	Mutation rate: 0.2	

according to different search environment and thus improve search efficiency. Third, to further enhance convergence, the total cost reduction strategy is proposed to improve solution quality. Therefore, we can draw a conclusion that these strategies have a positive effect on the behavior of the algorithm.

Figure 8 presents the PF approximations found in the run with the best IGD value of each MOEA for three level instances with small, medium, and large scale. It is evident from Figure 8(a) that although all MOEAs can find some approximations with regard to convergence for the small scale problem, MODBSA is capable of covering more areas than other MOEAs. As is depicted in Figure 8(b), MODBSA can show better convergence and coverage performances for the medium scale problem, while the other MOEAs tend to fall into local optima. The outperformance of MODBSA can be attributed to the adaptive mechanism, by which MODBSA can search preferable solutions in different directions to enhance search diversity. We can also observe from Figure 8(c) that the MODBSA has good convergence performance compared with its MOEAs. The good performance of MODBSA on medium and large scale problems

Table 12: Mean and standard deviation of GD obtained by NSGA-II, SPEA2, PAES, and MODBSA.

Problem	NSGA-II	SPEA2	PAES	MODBSA
	Mean/std	Mean/std	Mean/std	Mean/std
10_02	$3.93e-03/1.1e-03$	$1.14e-02/1.4e-03$	$1.84e-02/5.0e-03$	**$7.94e-04/4.0e-04$**
10_04	$4.34e-03/1.4e-03$	$1.11e-02/2.3e-03$	$1.75e-02/6.8e-03$	**$1.10e-03/7.3e-04$**
10_06	$3.96e-03/2.2e-03$	$1.06e-02/2.8e-03$	$2.12e-02/2.1e-01$	**$1.06e-03/8.2e-04$**
10_08	$2.89e-03/7.4e-04$	$1.02e-02/3.1e-03$	$2.74e-02/1.2e-01$	**$1.08e-03/7.0e-04$**
10_10	$2.40e-03/1.2e-03$	$1.77e-02/1.5e-02$	$8.54e-02/7.6e-02$	**$1.08e-03/6.0e-04$**
30_02	$3.71e-03/1.2e-03$	$1.12e-02/2.1e-03$	$1.58e-02/3.3e-03$	**$1.03e-03/2.5e-04$**
30_04	$3.69e-03/1.3e-03$	$1.00e-02/4.9e-03$	$2.20e-02/2.1e-02$	**$8.56e-04/7.7e-03$**
30_06	$4.87e-03/3.3e-03$	$1.02e-02/3.5e-03$	$5.00e-02/4.0e-02$	**$7.71e-04/5.5e-04$**
30_08	$3.67e-03/2.0e-03$	$9.01e-03/3.3e-03$	$7.63e-02/7.5e-02$	**$7.93e-04/3.1e-02$**
30_10	$3.80e-03/3.1e-03$	$2.03e-02/8.6e-03$	$7.68e-02/7.1e-02$	**$9.75e-04/3.9e-02$**
50_02	**$4.18e-03/2.1e-03$**	$1.28e-02/4.6e-03$	$2.07e-02/1.2e-02$	$8.20e-03/1.6e-02$
50_04	**$5.14e-03/1.5e-03$**	$1.18e-02/5.1e-03$	$3.52e-02/2.6e-02$	$1.24e-02/4.0e-02$
50_06	**$5.26e-03/3.2e-03$**	$1.03e-02/4.4e-03$	$6.50e-02/5.9e-02$	$1.10e-02/4.0e-02$
50_08	**$6.56e-03/4.0e-03$**	$1.06e-02/5.7e-03$	$7.47e-02/9.1e-02$	$3.18e-02/5.8e-02$
50_10	$1.05e-02/5.1e-03$	$2.20e-02/1.5e-02$	$9.58e-02/7.6e-02$	**$7.99e-03/4.9e-02$**
80_02	**$5.64e-03/2.2e-03$**	$1.27e-02/5.3e-03$	$2.61e-02/1.1e-02$	$1.11e-02/1.5e-02$
80_04	**$6.64e-03/3.3e-03$**	$1.27e-02/5.7e-03$	$3.17e-02/1.3e-02$	$2.47e-02/6.0e-02$
80_06	**$8.26e-03/4.8e-03$**	$1.34e-02/6.3e-03$	$4.80e-02/2.6e-02$	$3.08e-02/7.9e-02$
80_08	**$1.06e-02/7.1e-03$**	$2.05e-02/1.3e-02$	$6.00e-02/3.9e-02$	$3.37e-02/5.9e-02$
80_10	$3.61e-02/2.3e-02$	$3.84e-02/1.7e-02$	$7.81e-02/8.9e-02$	**$2.65e-02/4.4e-02$**
100_02	$8.31e-03/1.3e-03$	$1.34e-02/8.7e-03$	$2.74e-02/7.8e-03$	**$6.86e-03/1.8e-02$**
100_04	**$1.00e-02/4.3e-03$**	$1.20e-02/5.9e-03$	$4.29e-02/1.6e-02$	$2.21e-02/3.9e-02$
100_06	**$1.28e-02/5.2e-03$**	$1.59e-02/6.1e-03$	$5.64e-02/4.9e-02$	$4.97e-02/7.4e-02$
100_08	**$1.78e-02/1.4e-02$**	$2.75e-02/1.2e-02$	$7.89e-02/3.0e-02$	$6.63e-02/9.5e-02$
100_10	$5.92e-02/1.6e-02$	$4.88e-02/1.7e-02$	$1.11e-01/4.5e-02$	**$4.84e-02/8.3e-02$**
200_02	**$9.21e-03/3.2e-03$**	$1.15e-02/6.0e-03$	$2.57e-02/7.2e-03$	$1.68e-02/8.0e-03$
200_04	$1.83e-02/8.2e-03$	**$1.73e-02/7.5e-03$**	$3.33e-02/1.9e-02$	$2.25e-02/3.8e-02$
200_06	**$3.49e-02/1.5e-02$**	$4.42e-02/1.7e-02$	$7.58e-02/1.9e-02$	$4.86e-02/9.1e-02$
200_08	**$1.22e-01/2.1e-02$**	$2.61e-01/4.0e-02$	$3.88e-01/3.8e-02$	$1.45e-01/3.2e-02$
200_10	$3.87e-01/8.2e-02$	$5.46e-01/2.4e-01$	$4.42e-01/2.0e-01$	**$1.12e-01/1.1e-01$**
Hit rate	14/30	1/30	0/30	15/30

may be based on the fact that the dual population can help to balance the exploitation and exploration of MODBSA. Figure 8 not only presents good convergence of MODBSA but also illustrates a widespread coverage of MODBSA. In addition, MODBSA can find more nondominated solutions than other MOEAs, which implies good exploration ability of MODBSA. Therefore, we can conclude that MODBSA is very suitable for addressing this type of scheduling problem.

The statistical results are plotted as boxplots in Figure 9. The vertical axis of each subfigure represents the IGD value and the horizontal axis represents the different MOEAs. The lower position of box denotes better performance. The narrower the shape of the box is, the more stable the corresponding algorithm is. Clearly, the MODBSA is overwhelming without any exception in terms of IGD metric for three above level instances. It is consistent with previous numerical analysis and our view that the MODBSA outperforms other MOEAs considered for the SSPWCPT. The reasons behind the good performance of MODBSA are as follows. First, dual population scheme makes MODBSA have a better exploration ability as that it has a greater choice to search for different unknown areas of the search space. Second, total cost reduction strategy can improve the convergence of MODBSA since the quality of solution can be improved when total cost criterion is reduced while the tardiness criterion remains unchanged.

6. Conclusions and Future Work

In this paper, a multiobjective single machine scheduling problem with CPT is studied. The objective of this study

Table 13: Mean and standard deviation of Spread obtained by NSGA-II, SPEA2, PAES, and MODBSA.

Problem	NSGA-II	SPEA2	PAES	MODBSA
	Mean/std	Mean/std	Mean/std	Mean/std
10_02	1.38e + 00/1.3e − 02	1.46e + 00/3.3e − 02	**1.01e + 00/3.0e − 02**	1.66e + 00/2.4e − 02
10_04	1.34e + 00/3.2e − 02	1.42e + 00/3.9e − 02	**1.05e + 00/1.3e − 01**	1.61e + 00/2.9e − 02
10_06	1.31e + 00/4.9e − 02	1.47e + 00/6.4e − 02	**1.16e + 00/2.1e − 01**	1.59e + 00/1.9e − 02
10_08	1.23e + 00/1.5e − 02	1.54e + 00/6.9e − 02	**1.22e + 00/2.2e − 01**	1.54e + 00/8.8e − 03
10_10	**1.08e + 00/8.5e − 02**	1.55e + 00/8.3e − 02	1.30e + 00/1.8e − 01	1.29e + 00/1.6e − 02
30_02	7.41e − 01/5.2e − 02	1.46e + 00/4.9e − 02	1.01e + 00/3.9e − 02	**4.96e − 01/8.0e − 02**
30_04	7.81e − 01/5.2e − 02	1.45e + 00/4.8e − 02	1.09e + 00/1.4e − 01	**5.79e − 01/2.6e − 01**
30_06	8.03e − 01/5.7e − 02	1.43e + 00/5.9e − 02	1.23e + 00/1.9e − 01	**5.39e − 01/2.9e − 01**
30_08	8.11e − 01/4.7e − 02	1.45e + 00/5.0e − 02	1.34e + 00/1.4e − 01	**6.15e − 01/4.1e − 01**
30_10	8.54e − 01/5.6e − 02	1.50e + 00/5.1e − 02	1.33e + 00/1.5e − 01	**6.00e − 01/4.3e − 01**
50_02	7.08e − 01/6.1e − 02	1.48e + 00/5.0e − 02	1.10e + 00/1.3e − 01	**6.58e − 01/3.1e − 01**
50_04	7.54e − 01/7.7e − 02	1.43e + 00/4.1e − 02	1.19e + 00/1.5e − 01	**7.43e − 01/4.2e − 01**
50_06	7.82e − 01/8.7e − 02	1.45e + 00/4.6e − 02	1.27e + 00/1.5e − 01	**7.69e − 01/4.7e − 01**
50_08	**7.70e − 01/5.1e − 02**	1.47e + 00/4.0e − 02	1.32e + 00/1.4e − 01	8.35e − 01/5.1e − 01
50_10	8.18e − 01/5.7e − 02	1.50e + 00/4.0e − 02	1.33e + 00/1.1e − 01	**7.02e − 01/4.7e − 01**
80_02	**7.13e − 01/4.5e − 02**	1.50e + 00/4.4e − 02	1.07e + 00/7.5e − 02	7.26e − 01/3.3e − 01
80_04	**7.58e − 01/5.2e − 02**	1.49e + 00/4.3e − 02	1.14e + 00/1.4e − 01	8.80e − 01/4.3e − 01
80_06	**7.76e − 01/7.1e − 02**	1.49e + 00/3.2e − 02	1.23e + 00/1.5e − 01	9.12e − 01/4.7e − 01
80_08	**7.95e − 01/5.2e − 02**	1.46e + 00/3.2e − 02	1.27e + 00/1.5e − 01	1.02e + 00/3.7e − 01
80_10	**8.93e − 01/4.4e − 02**	1.47e + 00/4.4e − 02	1.29e + 00/1.6e − 01	9.00e − 01/4.4e − 01
100_02	7.35e − 01/5.8e − 02	1.52e + 00/4.0e − 02	1.07e + 00/9.4e − 02	**6.68e − 01/3.7e − 01**
100_04	**7.66e − 01/8.4e − 02**	1.50e + 00/3.3e − 02	1.19e + 00/1.2e − 01	9.22e − 01/3.9e − 01
100_06	**7.79e − 01/7.6e − 02**	1.50e + 00/3.9e − 02	1.27e + 00/1.4e − 01	9.28e − 01/4.8e − 01
100_08	**8.36e − 01/6.6e − 02**	1.49e + 00/4.1e − 02	1.32e + 00/1.0e − 01	1.08e + 00/3.9e − 01
100_10	**9.20e − 01/5.4e − 02**	1.44e + 00/5.9e − 02	1.35e + 00/5.7e − 02	9.74e − 01/4.4e − 01
200_02	7.96e − 01/5.4e − 02	1.54e + 00/2.0e − 02	1.02e + 00/2.6e − 02	**7.83e − 01/1.9e − 01**
200_04	**8.00e − 01/6.0e − 02**	1.49e + 00/2.7e − 02	1.12e + 00/9.1e − 02	9.59e − 01/2.9e − 01
200_06	**8.24e − 01/4.8e − 02**	1.48e + 00/4.0e − 02	1.29e + 00/8.6e − 02	1.17e + 00/2.3e − 01
200_08	**8.12e − 01/4.6e − 02**	1.49e + 00/4.2e − 02	1.09e + 00/7.9e − 02	9.63e − 01/2.3e − 01
200_10	**1.01e + 00/8.0e − 02**	1.33e + 00/7.8e − 02	1.08e + 00/1.1e − 01	1.14e + 00/9.1e − 02
Hit rate	15/30	0/30	4/30	11/30

is to minimize the total tardiness and total compression cost simultaneously. To solve this multiobjective problem, a new multiobjective discrete backtracking search algorithm (MODBSA) is proposed. In MODBSA, a new solution representation is developed to adapt to the characteristic of the problem. Experimental results show the validity of the solution representation. To improve search diversity and efficiency, we propose two improvement strategies into the MODBSA. First, an adaptive selection scheme is designed to select a suitable population for update operation. Second, a total cost reduction strategy is embedded into the MODBSA for enhancing exploitation ability. The efficiency of each improvement strategy is separately validated by experimental studies. The MODBSA is also compared with NSGA-II, SPEA2, and PAES on 30 instances. The empirical results demonstrate that the MODBSA outperforms its rivals on most instances. In conclusion, the main contributions of this work are at least threefold.

(1) A multiobjective mathematical model of scheduling problem with CPT is constructed. A new multiobjective backtracking search algorithm is developed to solve this scheduling problem.

Table 14: Mean and standard deviation of IGD obtained by NSGA-II, SPEA2, PAES, and MODBSA.

Problem	NSGA-II	SPEA2	PAES	MODBSA
	Mean/std	Mean/std	Mean/std	Mean/std
10_02	$8.58e-04/2.3e-04$	$2.21e-03/2.3e-04$	$1.69e-02/3.8e-03$	**1.31e – 04/6.3e – 05**
10_04	$1.52e-03/6.2e-04$	$2.57e-03/3.1e-04$	$1.54e-02/3.0e-03$	**2.99e – 04/1.2e – 04**
10_06	$1.13e-03/6.3e-04$	$3.08e-03/4.1e-04$	$1.75e-02/3.9e-03$	**1.94e – 04/6.7e – 05**
10_08	$1.28e-03/3.9e-04$	$3.91e-03/5.0e-04$	$1.95e-02/3.8e-03$	**1.17e – 04/4.9e – 05**
10_10	$1.08e-03/4.3e-04$	$5.48e-03/1.0e-03$	$1.68e-02/3.1e-03$	**3.83e – 04/1.1e – 04**
30_02	$3.06e-03/1.0e-03$	$4.36e-03/7.7e-04$	$2.63e-02/3.3e-03$	**2.72e – 04/2.0e – 04**
30_04	$3.73e-03/8.8e-04$	$4.14e-03/6.2e-04$	$2.46e-02/3.6e-03$	**1.85e – 04/1.5e – 04**
30_06	$4.04e-03/8.9e-04$	$4.16e-03/6.0e-04$	$2.22e-02/4.1e-03$	**2.17e – 04/7.2e – 05**
30_08	$3.90e-03/7.5e-04$	$4.02e-03/5.1e-04$	$2.10e-02/3.1e-03$	**1.87e – 04/9.7e – 05**
30_10	$7.19e-03/1.6e-03$	$6.64e-03/8.0e-04$	$2.67e-02/4.0e-03$	**3.71e – 04/1.5e – 04**
50_02	$4.04e-03/9.5e-04$	$5.32e-03/7.0e-04$	$2.51e-02/3.4e-03$	**2.02e – 04/7.2e – 05**
50_04	$5.24e-03/1.7e-03$	$5.92e-03/5.5e-04$	$2.30e-02/3.7e-03$	**3.46e – 04/1.4e – 04**
50_06	$5.34e-03/2.0e-03$	$5.70e-03/4.6e-04$	$2.31e-02/3.5e-03$	**3.38e – 04/1.2e – 04**
50_08	$6.14e-03/1.0e-03$	$6.96e-03/5.3e-04$	$2.28e-02/2.7e-03$	**2.60e – 04/7.4e – 05**
50_10	$1.33e-02/3.3e-03$	$1.29e-02/1.3e-03$	$2.90e-02/1.1e-02$	**6.74e – 04/3.0e – 04**
80_02	$7.93e-03/1.7e-03$	$1.02e-02/1.1e-03$	$3.04e-02/2.7e-03$	**4.55e – 04/1.4e – 04**
80_04	$8.92e-03/2.0e-03$	$1.03e-02/8.6e-04$	$2.91e-02/2.1e-03$	**5.68e – 04/2.5e – 04**
80_06	$1.33e-02/5.2e-03$	$1.38e-02/1.1e-03$	$2.77e-02/1.8e-03$	**5.20e – 04/2.3e – 04**
80_08	$1.60e-02/2.9e-03$	$1.78e-02/1.1e-03$	$2.81e-02/2.1e-03$	**1.26e – 03/4.3e – 04**
80_10	$2.43e-02/5.9e-03$	$1.88e-02/1.4e-03$	$3.01e-02/2.0e-03$	**7.49e – 04/2.6e – 04**
100_02	$1.20e-02/2.5e-03$	$1.60e-02/1.2e-03$	$3.19e-02/2.7e-03$	**8.35e – 04/3.2e – 04**
100_04	$1.37e-02/5.2e-03$	$1.42e-02/8.5e-04$	$2.78e-02/2.3e-03$	**4.84e – 04/1.3e – 04**
100_06	$1.52e-02/6.1e-03$	$1.65e-02/1.4e-03$	$2.77e-02/2.2e-03$	**8.24e – 04/3.3e – 04**
100_08	$1.82e-02/4.7e-03$	$1.74e-02/1.3e-03$	$2.63e-02/2.2e-03$	**6.64e – 04/3.2e – 04**
100_10	$3.28e-02/6.9e-03$	$2.01e-02/1.4e-03$	$2.93e-02/2.5e-03$	**9.08e – 04/4.4e – 04**
200_02	$1.25e-02/3.2e-03$	$1.69e-02/8.1e-04$	$5.26e-02/6.4e-03$	**6.41e – 03/2.8e – 05**
200_04	$1.44e-02/4.4e-03$	$1.73e-02/1.2e-03$	$4.26e-02/3.8e-03$	**4.41e – 03/2.9e – 05**
200_06	$2.29e-02/4.7e-03$	$1.96e-02/2.1e-03$	$2.79e-02/1.6e-03$	**1.51e – 03/3.6e – 04**
200_08	$1.18e-01/2.3e-02$	$1.17e-01/2.0e-02$	$5.43e-01/2.6e-02$	**4.02e – 02/1.0e – 02**
200_10	$2.52e-01/4.6e-02$	$2.14e-01/3.3e-02$	$4.97e-01/1.9e-01$	**1.05e – 02/8.3e – 03**
Hit rate	0/30	0/30	0/30	30/30

(2) A new solution representation is provided for the scheduling problems with CPT. It also provides a new encoding scheme in solving scheduling problems.

(3) The adaptive selection strategy and the total cost reduction strategy both can improve the performance of the MODBSA on instances. The adaptive selection mechanism can help to enhance exploration ability of the MODBSA, while the total cost reduction strategy can improve exploitation ability.

With respect to future work, an interesting issue worth studying is extending MODBSA to more complex scheduling problems with CPT such as the dynamic job shop scheduling problem. Another research direction is to further improve the search efficiency of the algorithm by incorporating heuristic based on problem property.

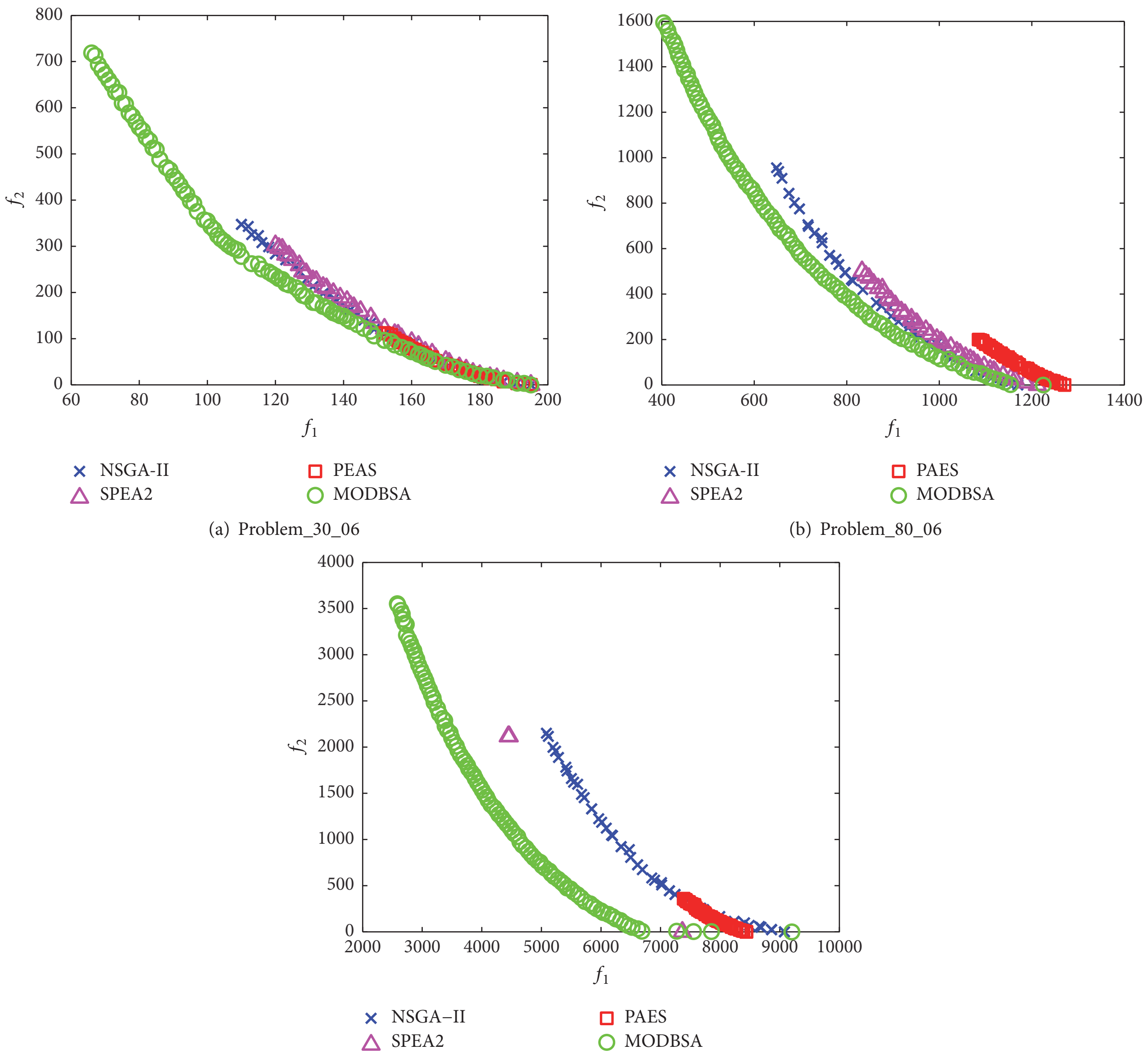
Notations

Parameters

n:	Number of jobs
J_j:	Job j
$J_{[i]}$:	Job in the ith position in a sequence
π:	Processing sequence of jobs, namely, $\pi = [J_{[1]}, \ldots, J_{[n]}]$
p_j:	Normal processing time of job j

TABLE 15: Wilcoxon signed rank test results based on the best metrics for each instance with 30 independent runs (a level of significant $\alpha = 0.05$).

Number	GD									Δ									IGD								
	MODBSA versus NSGA-II			MODBSA versus SPEA2			MODBSA versus PAES			MODBSA versus NSGA-II			MODBSA versus SPEA2			MODBSA versus PAES			MODBSA versus NSGA-II			MODBSA versus SPEA2			MODBSA versus PAES		
	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win	R^+	R^-	p value/win
10_02	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	0	465	1.73e − 06/−	0	465	1.73e − 06/−	0	465	1.73e − 06/−	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
10_04	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	0	465	1.73e − 06/−	0	465	1.73e − 06/−	0	465	1.73e − 06/−	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
10_06	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	0	465	1.73e − 06/−	3	462	2.35e − 06/−	0	465	1.73e − 06/−	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
10_08	463	2	2.13e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	0	465	1.73e − 06/−	208	257	6.14e − 01/ =	0	465	1.73e − 06/−	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
10_10	452	13	6.34e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	0	465	1.73e − 06/−	465	0	1.73e − 06/+	248	217	7.50e − 01/ =	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
30_02	432	33	4.07e − 05/+	462	3	2.35e − 06/+	464	1	1.92e − 06/+	456	9	4.29e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
30_04	253	212	6.73e − 01/ =	360	105	8.73e − 03/+	456	9	4.29e − 06/+	393	72	9.63e − 04/+	465	0	1.73e − 06/+	459	6	3.18e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
30_06	325	140	5.71e − 02/ =	334	131	3.68e − 02/+	434	31	3.41e − 05/+	387	78	1.50e − 03/+	465	0	1.73e − 06/+	458	7	3.52e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
30_08	231	234	9.75e − 01/ =	254	211	6.58e − 01/ =	456	9	4.29e − 06/+	337	128	3.16e − 02/+	465	0	1.73e − 06/+	460	5	2.88e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
30_10	240	225	8.77e − 01/ =	269	196	4.53e − 01/ =	441	24	1.80e − 05/+	382	83	2.10e − 03/+	465	0	1.73e − 06/+	462	3	2.35e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
50_02	104	361	8.21e − 03/−	304	161	1.41e − 01/ =	404	61	4.20e − 04/+	274	191	3.93e − 01/ =	465	0	1.73e − 06/+	449	16	8.47e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
50_04	114	351	1.48e − 02/−	167	298	1.78e − 01/ =	376	89	3.20e − 03/+	245	220	7.97e − 01/ =	465	0	1.73e − 06/+	418	47	1.36e − 04/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
50_06	117	348	1.75e − 02/−	167	298	1.78e − 01/ =	393	72	9.63e − 04/+	247	218	7.66e − 01/ =	465	0	1.73e − 06/+	438	27	2.37e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
50_08	91	374	3.60e − 03/−	114	351	1.48e − 02/−	386	79	1.60e − 03/+	214	251	7.04e − 01/ =	465	0	1.73e − 06/+	403	62	4.53e − 04/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
50_10	274	191	3.93e − 01/ =	250	215	7.19e − 01/ =	426	39	6.89e − 05/+	309	156	1.16e − 01/ =	465	0	1.73e − 06/+	440	25	1.97e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
80_02	83	382	2.10e − 03/−	260	205	5.72e − 01/ =	437	28	2.60e − 05/+	218	247	7.66e − 01/ =	465	0	1.73e − 06/+	433	32	3.72e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
80_04	79	386	1.60e − 03/−	121	344	2.18e − 02/−	258	207	6.00e − 01/+	170	295	1.99e − 01/ =	465	0	1.73e − 06/+	349	116	1.66e − 02/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
80_06	78	387	1.50e − 03/−	94	371	4.40e − 03/−	286	179	2.71e − 01/+	172	293	2.13e − 01/ =	465	0	1.73e − 06/+	348	117	1.75e − 02/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
80_08	50	415	1.74e − 04/−	112	353	1.32e − 02/−	356	109	1.11e − 02/+	114	351	1.48e − 02/−	465	0	1.73e − 06/+	393	72	9.63e − 04/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
80_10	278	187	3.49e − 01/ =	309	156	1.16e − 01/ =	403	62	4.53e − 04/+	244	221	8.13e − 01/ =	465	0	1.73e − 06/+	400	65	5.71e − 04/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
100_02	134	331	4.28e − 01/ =	326	139	5.45e − 02/ =	419	46	1.25e − 04/+	271	194	4.28e − 01/ =	465	0	1.73e − 06/+	431	34	4.45e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
100_04	75	390	1.203 − 03/−	100	365	6.40e − 03/−	391	74	1.10e − 03/+	136	329	4.72e − 02/−	465	0	1.73e − 06/+	399	66	6.16e − 04/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
100_06	70	395	8.31e − 04/−	86	379	2.60 − 03/−	337	128	3.16e − 02/+	174	291	2.29e − 01/ =	465	0	1.73e − 06/+	390	75	1.20e − 03/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
100_08	43	422	9.71e − 05/−	69	396	7.71e − 04/−	312	153	1.02e − 01/ =	107	358	9.80e − 03/−	465	0	1.73e − 06/+	370	95	4.70e − 03/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
100_10	279	186	3.39e − 01/ =	212	253	6.73e − 01/ =	448	17	9.32e − 06/+	219	246	7.81e − 01/ =	465	0	1.73e − 06/+	439	26	2.16e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
200_02	23	442	1.64e − 05/−	117	348	1.75e − 02/−	410	55	2.61e − 04/+	238	227	9.10e − 01/ =	465	0	1.73e − 06/+	446	19	1.13e − 05/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
200_04	135	330	4.49e − 02/−	185	280	3.29e − 01/ =	292	173	2.21e − 01/ =	107	358	9.80e − 03/−	465	0	1.73e − 06/+	339	126	2.85e − 02/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
200_06	131	334	3.68e − 02/−	159	306	1.31e − 01/ =	274	191	3.93e − 01/ =	17	448	9.32e − 06/−	465	0	1.73e − 06/+	331	134	4.28e − 02/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
200_08	68	397	7.16e − 04/−	457	8	3.88e − 06/+	465	0	1.73e − 06/+	93	372	4.10e − 03/−	465	0	1.73e − 06/+	363	102	7.30e − 03/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
200_10	464	1	1.92e − 06/+	464	1	1.92e − 06/+	465	0	1.73e − 06/+	29	436	2.84e − 05/−	456	9	4.29e − 06/+	143	322	6.57 − 02/ =	465	0	1.73e − 06/+	465	0	1.73e − 06/+	465	0	1.73e − 06/+
+/ = /−			7/8/15			10/12/8			27/3/0			5/13/12			26/1/3			24/2/4			30/0/0			30/0/0			30/0/0



(a) Problem_30_06

(b) Problem_80_06

(c) Problem_200_06

FIGURE 8: Pareto approximations obtained by different MOEAs with the best IGD value.

p_j^c: Crash (minimum allowable) processing time of job j

m_j: Maximum amount of compression processing time for job j, namely, $m_j = p_j - p_j^c$

p_j^a: Actual processing time of job j

c_j: Unit cost of compression processing time of job j

d_j: Due date of job j

T_j: Tardiness of job j, namely, $T_j = \max(0, C_j - d_j)$, where C_j is the completion time of job j

S_j: Start time of job j

M: An infinite positive number.

Decision Variables

x_j: Amount of compression processing time of job j

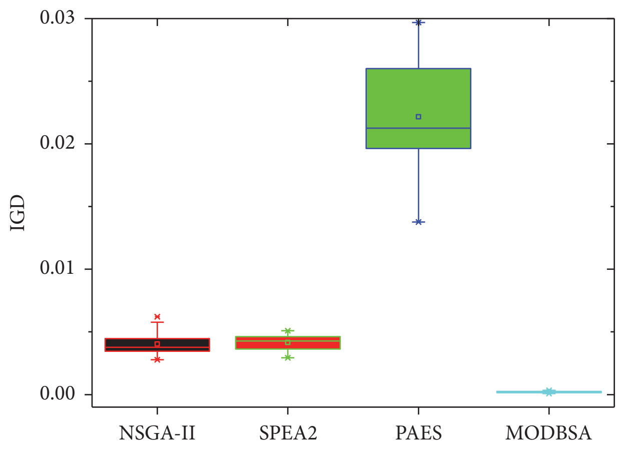
y_{jk}: It is set to 1 if J_j precedes J_k; it is set to 0 otherwise.

Conflicts of Interest

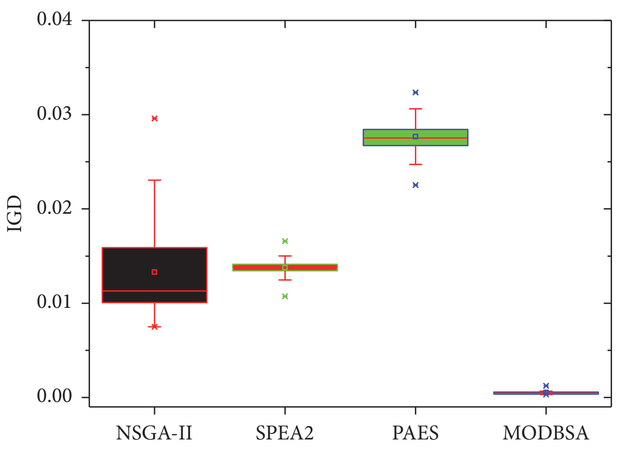
The authors declare that they have no conflicts of interest.

Acknowledgments

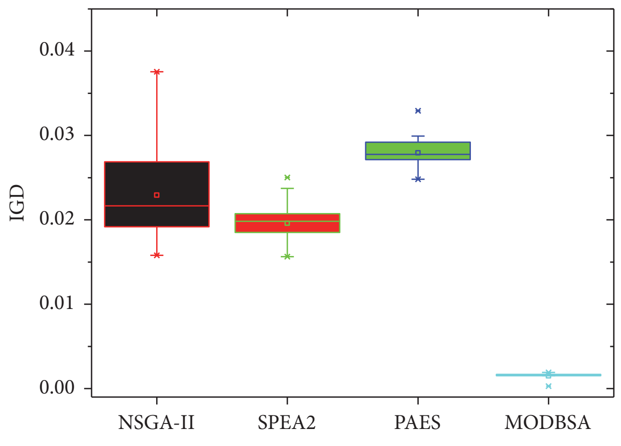
This research work was supported by the National Natural Science Foundation of China (NSFC) under Grant nos.



(a) Problem_30_06



(b) Problem_80_06



(c) Problem_200_06

Figure 9: IGD of boxplot for different MOEAs.

51375004 and 51435009, National Key Technology Support Program under Grant no. 2015BAF01B04, and Youth Science & Technology Chenguang Program of Wuhan under Grant no. 2015070404010187.

References

[1] B. Mor and G. Mosheiov, "Batch scheduling of identical jobs with controllable processing times," *Computers & Operations Research*, vol. 41, pp. 115–124, 2014.

[2] T. C. E. Cheng, C. Oguz, and X. D. Qi, "Due-date assignment and single machine scheduling with compressible processing times," *International Journal of Production Economics*, vol. 43, no. 1, pp. 29–35, 1996.

[3] X. L. Liang, H. P. Chen, and R. Xu, "Approximately optimal algorithms for scheduling a single machine with general convex resource functions," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 9, pp. 920–935, 2015.

[4] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *Journal of Cleaner Production*, vol. 144, pp. 228–238, 2017.

[5] V. Kayvanfar, G. M. Komaki, A. Aalaei, and M. Zandieh, "Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times," *Computers & Operations Research*, vol. 41, pp. 31–43, 2014.

[6] R. K. Kayan and M. S. Akturk, "A new bounding mechanism for the CNC machine scheduling problems with controllable processing times," *European Journal of Operational Research*, vol. 167, no. 3, pp. 624–643, 2005.

[7] C. Lu, X. Li, L. Gao, W. Liao, and J. Yi, "An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times," *Computers & Industrial Engineering*, vol. 104, pp. 156–174, 2017.

[8] A. C. Nearchou, "Scheduling with controllable processing times and compression costs using population-based heuristics," *International Journal of Production Research*, vol. 48, no. 23, pp. 7043–7062, 2010.

[9] L. Yedidsion, "Bi-criteria and tri-criteria analysis to minimize maximum lateness makespan and resource consumption for scheduling a single machine," *Journal of Scheduling*, vol. 15, no. 6, pp. 665–679, 2012.

[10] L. N. Van Wassenhove and K. R. Baker, "A bicriterion approach to time/cost trade-offs in sequencing," *European Journal of Operational Research*, vol. 11, no. 1, pp. 48–54, 1982.

[11] Y. Zhang, H. Zhang, and C. Lu, "Study on parameter optimization design of drum brake based on hybrid cellular multiobjective genetic algorithm," *Mathematical Problems in Engineering*, vol. 2012, Article ID 734193, 23 pages, 2012.

[12] Y. Yang, “A modified biogeography-based optimization for the flexible job shop scheduling problem,” *Mathematical Problems in Engineering*, vol. 2015, Article ID 184643, 10 pages, 2015.

[13] C. Lu, L. Gao, X. Li, and S. Xiao, “A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry,” *Engineering Applications of Artificial Intelligence*, vol. 57, pp. 61–79, 2017.

[14] P. Civicioglu, “Backtracking search optimization algorithm for numerical optimization problems,” *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[15] C. Lu, L. Gao, X. Li, and P. Chen, “Energy-efficient multi-pass turning operation using multi-objective backtracking search algorithm,” *Journal of Cleaner Production*, vol. 137, pp. 1516–1531, 2016.

[16] J. Du and J. Y.-T. Leung, “Minimizing total tardiness on one machine is NP-hard,” *Mathematics of Operations Research*, vol. 15, no. 3, pp. 483–495, 1990.

[17] Q. Lin, L. Gao, X. Li, and C. Zhang, “A hybrid backtracking search algorithm for permutation flow-shop scheduling problem,” *Computers & Industrial Engineering*, vol. 85, pp. 437–446, 2015.

[18] K. Ayan and U. Kiliç, “Optimal power flow of two-terminal HVDC systems using backtracking search algorithm,” *International Journal of Electrical Power and Energy Systems*, vol. 78, pp. 326–335, 2016.

[19] R. G. Vickson, “Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine,” *Operations Research*, vol. 28, no. 5, pp. 1155–1167, 1980.

[20] A. Janiak and M. Y. Kovalyov, “Single machine scheduling subject to deadlines and resource dependent processing times,” *European Journal of Operational Research*, vol. 94, no. 2, pp. 284–291, 1996.

[21] D. Shabtay and M. Kaspi, “Minimizing the total weighted flow time in a single machine with controllable processing times,” *Computers & Operations Research*, vol. 31, no. 13, pp. 2279–2289, 2004.

[22] T. C. Cheng, M. Y. Kovalyov, and N. V. Shakhlevich, “Scheduling with controllable release dates and processing times: makespan minimization,” *European Journal of Operational Research*, vol. 175, no. 2, pp. 751–768, 2006.

[23] N. Yin and X.-Y. Wang, “Single-machine scheduling with controllable processing times and learning effect,” *International Journal of Advanced Manufacturing Technology*, vol. 54, no. 5-8, pp. 743–748, 2011.

[24] K. Xu, Z. Feng, and L. Ke, “Single machine scheduling with total tardiness criterion and convex controllable processing times,” *Annals of Operations Research*, vol. 186, pp. 383–391, 2011.

[25] C.-T. Tseng, C.-J. Liao, and K.-L. Huang, “Minimizing total tardiness on a single machine with controllable processing times,” *Computers & Operations Research*, vol. 36, no. 6, pp. 1852–1858, 2009.

[26] V. Kayvanfar, I. Mahdavi, and G. H. M. Komaki, “Single machine scheduling with controllable processing times to minimize total tardiness and earliness,” *Computers and Industrial Engineering*, vol. 65, no. 1, pp. 166–175, 2013.

[27] V. Kayvanfar, I. Mahdavi, and G. M. Komaki, “A drastic hybrid heuristic algorithm to approach to JIT policy considering controllable processing times,” *International Journal of Advanced Manufacturing Technology*, vol. 69, no. 1–4, pp. 257–267, 2013.

[28] L. Yedidsion, D. Shabtay, and M. Kaspi, “Complexity analysis of an assignment problem with controllable assignment costs and its applications in scheduling,” *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, vol. 159, no. 12, pp. 1264–1278, 2011.

[29] Y. Yin, T. C. E. Cheng, S.-R. Cheng, and C.-C. Wu, “Single-machine batch delivery scheduling with an assignable common due date and controllable processing times,” *Computers and Industrial Engineering*, vol. 65, no. 4, pp. 652–662, 2013.

[30] D. Giglio, “Optimal control strategies for single-machine family scheduling with sequence-dependent batch setup and controllable processing times,” *Journal of Scheduling*, vol. 18, no. 5, pp. 525–543, 2015.

[31] D. Shabtay and G. Steiner, “A survey of scheduling with controllable processing times,” *Discrete Applied Mathematics*, vol. 155, no. 13, pp. 1643–1666, 2007.

[32] M. L. Pinedo, *Scheduling, Theory, Algorithms, and Systems*, Springer Science & Business Media, Berlin, Germany, 2012.

[33] L. Yedidsion, D. Shabtay, and M. Kaspi, “A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine,” *Journal of Scheduling*, vol. 10, no. 6, pp. 341–352, 2007.

[34] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-wesley Reading, Menlo Park, Calif, USA, 1989.

[35] Z. Xie, C. Zhang, X. Shao, W. Lin, and H. Zhu, “An effective hybrid teaching-learning-based optimization algorithm for permutation flow shop scheduling problem,” *Advances in Engineering Software*, vol. 77, pp. 35–47, 2014.

[36] 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.

[37] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[38] C. Lu, S. Xiao, X. Li, and L. Gao, “An effective multi-objective discrete grey Wolf optimizer for a real-world scheduling problem in welding production,” *Advances in Engineering Software*, vol. 99, pp. 161–176, 2016.

[39] J. J. Durillo and A. J. Nebro, “JMetal: a Java framework for multi-objective optimization,” *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[40] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[41] J. D. Knowles and D. W. Corne, “The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization,” in *Proceedings of the Conference on Evolutionary Computation*, Washington, DC, USA, July 1999.

[42] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: improving the strength Pareto evolutionary algorithm,” in *Proceedings of the of Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pp. 95–100, Athens, Greece, September 2001.