Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 2490620, 10 pages
https://doi.org/10.1155/2017/2490620

Research Article

Hybrid Differential Evolution Optimisation for Earth Observation Satellite Scheduling with Time-Dependent Earliness-Tardiness Penalties

Guoliang Li, Cheng Chen, Feng Yao, Renjie He, and Yingwu Chen

College of Information System and Management, National University of Defense Technology, Changsha, Hunan, China

Correspondence should be addressed to Guoliang Li; worldchinali@126.com

Received 23 May 2017; Accepted 24 July 2017; Published 22 August 2017

Academic Editor: Thomas Hanne

We study the order acceptance and scheduling (OAS) problem with time-dependent earliness-tardiness penalties in a single agile earth observation satellite environment where orders are defined by their release dates, available processing time windows ranging from earliest start date to deadline, processing times, due dates, sequence-dependent setup times, and revenues. The objective is to maximise total revenue, where the revenue from an order is a piecewise linear function of its earliness and tardiness with reference to its due date. We formulate this problem as a mixed integer linear programming model and develop a novel hybrid differential evolution (DE) algorithm under self-adaptation framework to solve this problem. Compared with classical DE, hybrid DE employs two mutation operations, scaling factor adaptation and crossover probability adaptation. Computational tests indicate that the proposed algorithm outperforms classical DE in addition to two other variants of DE.

1. Introduction

Over the last decade, the order acceptance and scheduling (OAS) problem has been widely applied in make-to-order systems. In the existing literature, an order completed past the due date incurs a penalty. There is no reward or penalty for order completion before the promised date, for example, the printing and laminating company described by Oğuz et al. [1].

The just-in-time philosophy, which is attracting more attention currently, refers to the opinion that earliness, in addition to tardiness, should be discouraged. It has promoted the study of scheduling problems in which orders are preferred to be completed just at their respective due dates, and both early and tardy products are penalised. We indeed need to consider the penalty for the early completion of orders in an OAS system for some cases, which is the motivation in Section 2.

A variety of OAS problems under different settings and with various objectives have been reviewed by Slotnick [2]. The problem is classified into categories in terms of problem characteristics, objectives, and solution methods. There are also some papers to study a single machine scheduling problem with the objective of minimising the total tardiness [3, 4]. In the existing literature, we focus on reviewing studies on the OAS problem with time-related penalties and sequence-dependent setup times.

For the single machine total tardiness problem with sequence-dependent setup times, Anghinolfi and Paolucci [5] proposed a novel discrete particle swarm optimisation (DPSO) approach to solve this nondeterministic polynomial time- (NP-) hard problem, and Tasgetiren et al. [6] provided a discrete differential evolution (DE) algorithm for the same problem. Oğuz et al. [1] studied the OAS problem, where orders were defined by their release dates, processing times, due dates, deadlines, sequence-dependent setup times, revenues, and tardiness penalty weights. The authors assumed that the revenue of each accepted order is a function of its tardiness and deadline. They provided a mixed integer linear programming (MILP) formulation of the problem and developed three heuristic algorithms to solve this problem. Cesaret et al. [7] studied the OAS problem with the same setting as Oğuz et al. [1] and developed a tabu search algorithm. The proposed tabu search outperforms

the three heuristics proposed by Oğuz et al. [1] in terms of solution quality. For the same problem setting, Lin and Ying [8] and Cheng et al. [9] developed an artificial bee colony algorithm and improved genetic algorithm. The deviation from the upper bound of the benchmark instances was further improved by the two algorithms.

There are some studies on the scheduling problem with earliness/tardiness penalties in a single machine environment. Hassin and Shani [10] studied several scheduling problems. They considered earliness and tardiness (E/T) penalties and developed polynomial time algorithms for special cases of these problems with nonexecution penalties. Chen et al. [11] formulated a mathematical model for production scheduling subject to multiple constraints, including sequence-dependent setup costs, nonexecution penalties, and E/T penalties. The authors considered the E/T penalty weight as a special value (equal to one) to calculate. Baker [12] assumed that processing times follow normal distributions and due dates are decisions. The author developed a branch and bound algorithm to determine optimal solutions to minimise the total expected E/T costs and tested some heuristic procedures. Shabtay [13] analysed a model that integrates due date assignment and scheduling decisions. The objective was to minimise the total weighted earliness, tardiness, and due date assignment penalties.

To the best of our knowledge, the tardiness penalty and earliness penalty are seldom jointly considered for OAS problems in current research. In this study, we simultaneously consider the tardiness penalty and earliness penalty for the OAS problem with a sequence-dependent setup time on a single agile earth observation satellite. Inspired by agile earth observation satellite scheduling, we formulated a general MILP model for this problem. The formulated general MILP model can be also applied in manufacturing system, logistics management, and make-to-order systems, as oversubscribed scheduling problem also exists in these fields.

The remainder of the paper is organised as follows. We illustrate the motivation for this paper in Section 2. In Section 3, we formally define the OAS problem with time-dependent earliness-tardiness penalties. In Section 4, we present the mathematical model for this problem. In Section 5, we provide details for the proposed algorithm for solving the problem. In Section 6, we present the experimental studies. In Section 7, we provide the conclusion.

2. Motivation

With the expansion of earth observation satellites (EOSs) application, such as earth resource exploration, forest fire warning, volcano eruption discovery, and earthquake surveillance, EOSs become more limited and expensive relative to numerous orders submitted by users from different departments. The satellite operation firms that can operate on a make-to-order basis realize benefits in terms of timely image availability and decreased risk to imaging randomness. On the other hand, limitations on satellite resource force these firms to be selective on user orders. Users giving orders typically quote a due image quality and may penalise the firm for not-good quality image. This translates into reduced revenue and even loss of users.

Our motivation originates from the agile earth observation satellite (AEOS) scheduling problem. Compared with the traditional earth observation satellite, the AEOS is mobile along three axes (roll, pitch, and yaw). This mobility considerably increases the length of the time windows for imaging targets. The pitch angle required for the satellite to image the target changes in the time window. When the satellite flies over the target, the pitch angle required is zero and the best quality image is obtained. If the target is observed earlier or later than this time, a relatively larger pitch angle is required and a deterioration of image quality is incurred, as illustrated in Figure 1, which is partially taken from Lemaître et al. [14].

Consequently, the order processing time window for this target is from the earliest start time to the latest end time. Earth observation satellites are scarce and oversubscribed resources. A satellite can at most image one target at any time moment and cannot image all the user-requested targets because of its limited capacity and ability. When imaging targets in reality, the pitch angles required always differ from one target to another; therefore, the setup times depend on the imaging sequence of the targets. Thus, simultaneous decisions have to be made to choose which targets are to be imaged and fix the appropriate start time of imaging. We formulate this problem as an OAS problem, considering the time-dependent earliness and tardiness penalties with the objective of maximising total revenue. In the remainder of this paper, we refer to this problem as the OAS with time-dependent earliness-tardiness penalties (OASET) problem.

3. Problem Definition

In a single AEOS environment, a set of jobs are initialised at the beginning of the planning horizon. The satellite can process one job at most at any time moment without preemption. There are no precedence constraints among the jobs, and the production system does not break down. The description of the job is shown in Figure 2.

For convenience, the notations used in the remainder of the paper are summarised in the Notations.

Note that the setup time depends on the processing sequence of two consecutive jobs. Thus s_{ij} is not necessarily equal to s_{ji}.

The revenue of a job is defined as a piecewise linear function that depends on the earliest start date, processing time, due date, and deadline as illustrated in Figure 3. The job is not processed yet in region A and the job is not completed yet in region B. In both regions, we obtain no revenue. In regions C and D, the revenue gained decreases linearly with its earliness or tardiness, respectively. In region E, the manufacturer does not accept the job, which is completed beyond its deadline, and no revenue is gained. If the job is completed exactly at the due date, the maximum revenue is obtained.

Given a sequence σ_S of the selected job set S, we can calculate the completion time c_j of each job $j \in S$. Let $T_j(\sigma_S)$ be the tardiness of job j in sequence σ_S. We can calculate the

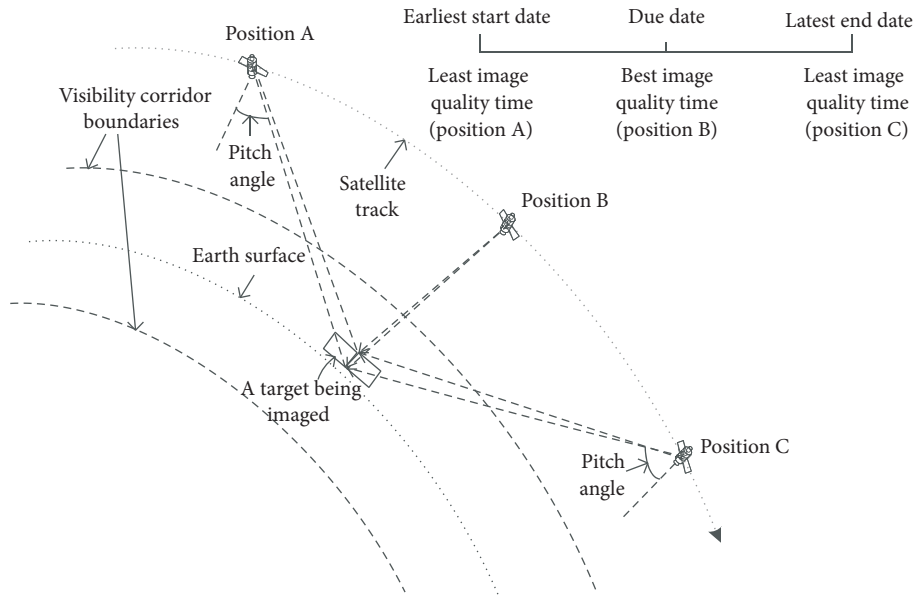

Figure 1: Illustration of the AEOS imaging target on the sea.

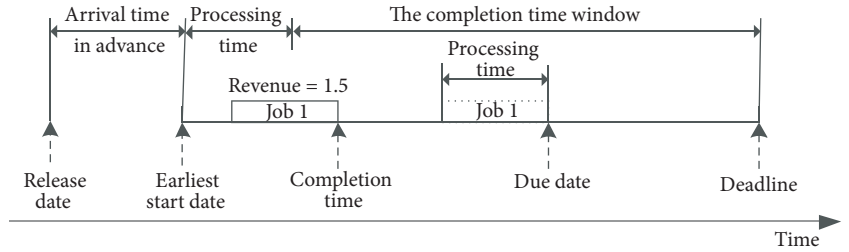

Figure 2: Description of job j.

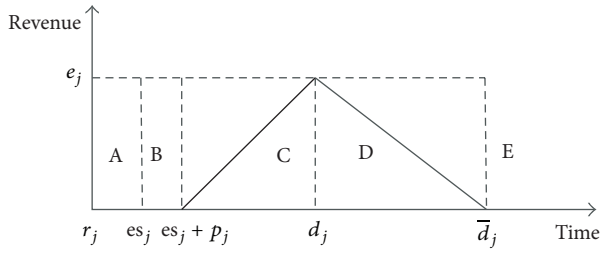

Figure 3: Revenue function for job j.

tardiness of job j using the formula $T_j(\sigma_S) = \max\{0, c_j - d_j\}$. The earliness of the job is $E_j(\sigma_S) = \max\{0, d_j - c_j\}$. Thus, the revenue of job j is calculated as $R_j(\sigma_S) = \max\{0, e_j - ew_j \cdot E_j(\sigma_S) - tw_j \cdot T_j(\sigma_S)\}$. Consequently, the total revenue gained from the sequence σ_S is $\mathrm{TR}(\sigma_S) = \Sigma_{j \in S} R_j(\sigma_S)$. The OASET problem is to determine the set S and sequence σ_S when $\mathrm{TR}(\sigma_S)$ is maximum.

This problem reduces to the OAS problem studied by Oğuz et al. [1] if the unit earliness penalty cost is set to zero and the release date equals the earliest start date for all orders. The OASET problem generalises the OAS problem, which has been proved to be NP-hard; thus it is still of NP-hard complexity.

4. The Mathematical Model

In this section, we formulate the OASET problem as an MILP model. We apply two binary variables, I and y, to denote order acceptance and sequencing decisions, respectively:

$$y_{ij} = \begin{cases} 1, & \text{if job } i \text{ precedes job } j,\ i, j \in J,\ i \neq j \\ 0, & \text{otherwise,} \end{cases}$$
$$I_i = \begin{cases} 1, & \text{if job } i \text{ is selected,}\ i \in J \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

MILP is

$$\max \quad \sum_{i=1}^{n} R_i \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1,\ j \neq i}^{n+1} y_{ij} = I_i, \quad \forall i = 0, \ldots, n, \tag{3}$$

$$\sum_{j=0,\ j\neq i}^{n} y_{ji} = I_i, \quad \forall i = 0,\ldots,n+1, \tag{4}$$

$$c_i + \left(s_{ij} + p_i\right) y_{ij} + \overline{d}_i \left(y_{ij} - 1\right) \leq c_j, \quad \forall i = 0,\ldots,n,\ \forall j = 1,\ldots,n+1,\ i \neq j, \tag{5}$$

$$\left(\mathrm{es}_j + p_j\right) I_j \leq c_j, \quad \forall i = 0,\ldots,n,\ \forall j = 1,\ldots,n+1,\ i \neq j, \tag{6}$$

$$c_i \leq \overline{d}_i I_i, \quad \forall i = 0,\ldots,n+1, \tag{7}$$

$$b_i + p_i = c_i, \quad \forall i = 0,\ldots,n+1, \tag{8}$$

$$T_i \leq c_i - d_i, \quad \forall i = 0,\ldots,n+1, \tag{9}$$

$$T_i \leq \left(\overline{d}_i - d_i\right) I_i, \quad \forall i = 0,\ldots,n+1, \tag{10}$$

$$T_i \geq 0, \quad \forall i = 0,\ldots,n+1, \tag{11}$$

$$E_i \geq d_i - c_i, \quad \forall i = 0,\ldots,n+1, \tag{12}$$

$$E_i \leq \left(d_i - \mathrm{es}_j - p_j\right) I_i, \quad \forall i = 0,\ldots,n+1, \tag{13}$$

$$E_i \geq 0, \quad \forall i = 0,\ldots,n+1, \tag{14}$$

$$R_i \leq e_i I_i - ew_i E_i - tw_i T_i, \quad \forall i = 1,\ldots,n, \tag{15}$$

$$R_i \geq 0, \quad \forall i = 1,\ldots,n, \tag{16}$$

$$c_0 = 0, \quad c_{n+1} = \max_{i=1,\ldots,n} \left\{\overline{d}_i\right\}, \tag{17}$$

$$I_0 = 1, \quad I_{n+1} = 1, \quad I_i \in \{0,1\}, \tag{18}$$

$$y_{ij} \in \{0,1\}, \quad \forall i = 0,\ldots,n,\ \forall j = 1,\ldots,n+1,\ i \neq j. \tag{19}$$

In this model, we add the definition of earliness for each job with constraints set (12), (13), and (14) and then update constraint (15) to calculate the revenue the firm can gain when job i is accepted and incurs tardiness T_i or earliness E_i. Additionally, the procession time window is characterised by the earliest start date and deadline; therefore, we use the earliest start date es_j in constraint (6).

5. Hybrid Differential Evolutionary Algorithm

Differential evolution (DE), which was first proposed by Storn and Price [15], has become arguably one of the most efficient stochastic real-parameter optimisation algorithms. DE is very simple and easy to implement. Despite this, DE has exhibited better performance than several other algorithms [16]. Based on classical DE, many variants have greatly improved performance for a variety of applications. We also propose a hybrid DE (HDE) to solve the problem in Section 5.5.

5.1. Parameter Vector. For an OASET problem with all n jobs, an n-dimensional real vector $x = (x_1, x_2, \ldots, x_n)$ represents an individual, where $0 \leq x_j \leq 1,\ j = 1,\ldots,n$. The value of x_j for job j represents the proportion of the part time before job j is completed in the completion interval $[\mathrm{es}_j + p_j, \overline{d}_j]$. The completion time c_j of job j is calculated by $c_j = (\mathrm{es}_j + p_j) + x_j(\overline{d}_j - \mathrm{es}_j - p_j)$. In such a representation, the completion time of each job j is located in the completion time window $\mathrm{CTW}_j = [\mathrm{es}_j + p_j, \overline{d}_j]$.

5.2. Graph Representation. Given a parameter vector, the completion time of each job is fixed. Therefore, the revenue that might be gained from each job is also fixed. To maximise the total revenue, we need to select a subset of jobs that satisfy the sequence-dependent setup time constraints. First, we introduce a directed acyclic graph G of the problem under a given parameter vector. The procedure is described as follows.

Step 1. Compute the procession start time b_j and completion time c_j of each job $j \in J$ under a given parameter vector.

Step 2. Sort the jobs in ascending order by their procession start times, suppose the sorted sequence of all the jobs J is σ_J, and calculate the penalised revenue $R_j(\sigma_J)$ for each job $j \in J$.

Step 3. First, add n nodes to graph G, where each node j ($j = 1,\ldots,n$) represents job j, then as illustrated in the mathematical model, add two dummy nodes including one denoted by "0" that represents the source node and another one denoted by "$n+1$" that represents the sink node to G.

Step 4. For each job i in σ_J and for each job j after job i in σ_J, if the sequence-dependent setup time constraint is satisfied, that is, $c_i + s_{ij} \leq b_j$, then add a directed arc$\langle i, j\rangle$ weighted $R_j(\sigma_J)$ to G.

Step 5. For each job j in σ_J, add a directed arc$\langle 0, j\rangle$ weighted $R_j(\sigma_J)$ and a directed arc$\langle i, n+1\rangle$ weighted zero to G.

Consider the following example: suppose there are four incoming jobs and the data for the problem are as follows: $n = 4$, $\mathrm{es}_j = [5, 10, 13, 20]$, $p_j = [5, 13, 10, 12]$, $\overline{d}_j = [40, 63, 53, 57]$, $e_j = [2, 4, 1, 5]$, $d_j = [30, 43, 36, 47]$, and $s_{ij} = [0, 2, 1, 1; 3, 0, 4, 1; 2, 3, 0, 4; 3, 2, 1, 0]$. Suppose the parameter vector $x = (0.5, 0.4, 0.6, 0.8)$, and we can obtain the start and completion times as $b_j = [20, 26, 31, 40]$ and $c_j = [25, 39, 41, 52]$, respectively; then the penalised revenues of the four jobs are calculated as $R_1 = 1.5$, $R_2 = 3.2$, $R_3 = 0.7$, $R_4 = 2.5$. Figure 4 illustrates the jobs sorted in ascending order by their start times. We can observe that Jobs 1 and 2 are penalised by earliness, whereas Jobs 3 and 4 are penalised by tardiness. The constructed directed graph from the jobs is illustrated in Figure 5.

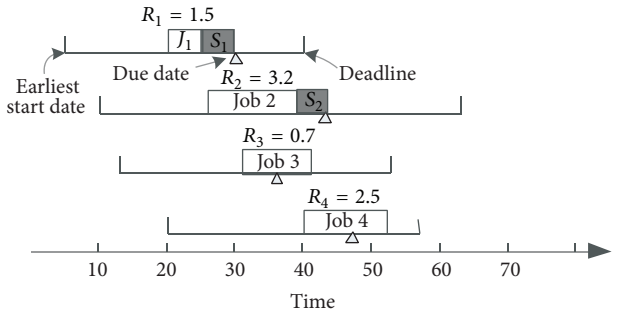

FIGURE 4: Example of jobs in ascending order by their procession start time.

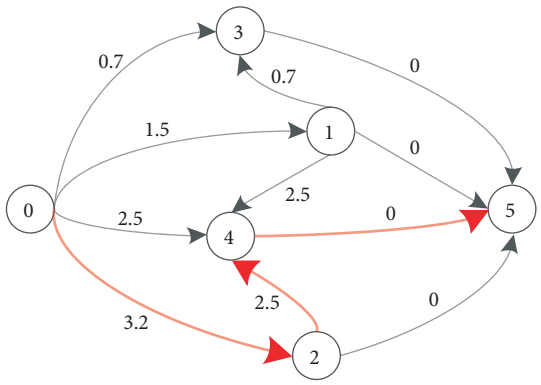

FIGURE 5: Directed graph constructed from the jobs.

5.3. Graph-Based Fitness Evaluation. Based on the directed acyclic graph constructed under a given parameter vector, selecting a subset of jobs maximising the total net revenue is equivalent to determining the longest path from the source node "0" to sink node "$n + 1$" [17]. The polynomial time algorithm exists to solve the longest path problem in a directed acyclic graph. We modify the $O(n^2)$ Bellman-Ford shortest path algorithm to determine the longest path in the graph. The nodes, except the source node and sink node in the path, represent the selected jobs in the solution. In Figure 5, the longest path, which is illustrated by bold arcs, is 0-2-4-5 with total weight $3.2 + 2.5 = 5.7$.

5.4. Classical DE. The classical DE algorithm proposed by Storn and Price [15] can be summarised as follows. The initial population comprising N_p individuals $\{x_k^0 = (x_{k1}^0, x_{k2}^0, \ldots, x_{kD}^0)\}$, $k = 1, \ldots, N_p$, is randomly generated, where $lb_q \le x_{kq}^0 \le ub_q$ and D is the dimension of the problem. DE generates a new population through mutation, crossover, and selection operators.

(1) Mutation. This operation generates new target vectors according to

$$v_k^g = x_{r_1}^g + F\left(x_{r_2}^g - x_{r_3}^g\right), \tag{20}$$

where r_1, r_2, and r_3 are different integers randomly chosen from $\{1, \ldots, N_p\}$ and different from the vector index k. F is a constant factor and is located in $[0, 2]$, according to Storn and Price [15].

(2) Crossover. This operation forms the trial vectors according to

$$u_{kq}^g = \begin{cases} v_{kq}^g, & \text{if rand}(0,1) \le \text{cr or } q = q_{\text{rand}}, \\ x_{kq}^g, & \text{otherwise}, \end{cases} \tag{21}$$

where cr is the constant crossover rate, which is often set to 0.9, and q_{rand} is a randomly chosen index from 1 to D to ensure that the trial vector u_k^g does not duplicate x_k^g.

(3) Selection. This operation determines which parent vector is selected for the next generation from vector x_k^g and trial vector u_k^g according to

$$x_k^{g+1} = \begin{cases} u_k^g, & \text{if } f\left(u_k^g\right) \text{ is not worse than } f\left(x_k^g\right), \\ x_k^g, & \text{otherwise}. \end{cases} \tag{22}$$

5.5. Hybrid DE Method. Our proposed HDE combines several components of other algorithms under a self-adaptation framework. The DE/current-to-pbest with archive in Adaptive Differential Evolution with Optional External Archive (JADE) [18] and the widely used DE/rand are combined under self-adaptation strategies. Considering the plateaus in the fitness landscape, the neighbourhood search in Neighbourhood Search Differential Evolution (NSDE) [19], which adopts two random distribution generators for tuning the scaling factor, is applied in our algorithm to help the algorithm escape from plateaus.

5.5.1. Mutation. In addition to (20), there are several other commonly used mutation operations [20]:

$$v_k^g = x_{\text{best}}^g + F\left(x_{r_1}^g - x_{r_2}^g\right), \tag{23}$$

$$v_k^g = x_{\text{best}}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) + F\left(x_{r_3}^g - x_{r_4}^g\right), \tag{24}$$

$$v_k^g = x_k^g + F\left(x_{\text{best}}^g - x_k^g\right) + F\left(x_{r_1}^g - x_{r_2}^g\right). \tag{25}$$

We apply self-adaptation strategies [21], and two mutation operations are selected as candidates. One is DE/ran/1, as shown in (20), and the other is DE/current-to-pbest proposed by Zhang and Sanderson [18], given as

$$v_k^g = x_k^g + F_k\left(x_{\text{pbest}}^g - x_k^g\right) + F_k\left(x_{r_1}^g - \tilde{x}_{r_2}^g\right), \tag{26}$$

where x_{pbest}^g is randomly chosen as one of the top $100p\%$ individuals in the current population and $\tilde{x}_{r_2}^g$ is randomly chosen from the union of the current population and archive. The archive is initiated to be empty. After each generation, the parent solutions that fail in the selection of (22) are added to the archive. We randomly remove some individuals in the archive to keep the size within N_p if the size exceeds N_p [19].

The mutation process applied here is the same as Self-adaptive Differential Evolution (SaDE) [21], except that (25)

used in SaDE is replaced by (26). The trial vector is produced as

$$v_k^g = \begin{cases} x_{r_1}^g + F_k\left(x_{r_2}^g - x_{r_3}^g\right), & \text{if rand}(0,1) < mp, \\ x_k^g + F_k\left(x_{\text{pbest}}^g - x_k^g\right) + F_k\left(x_{r_1}^g - \tilde{x}_{r_2}^g\right), & \text{otherwise,} \end{cases} \quad (27)$$

where mp is initialised as 0.5. After each generation, the number of trial vectors successfully entering the next generation generated by (20) and (26) is recorded as ns_1 and ns_2. The number of trial vectors that fail entering the next generation generated by (20) and (26) is recorded as nf_1 and nf_2. The four numbers are aggregated within a specific number of generations and the probability mp is updated as

$$mp = \frac{ns_1\left(ns_2 + nf_2\right)}{ns_2\left(ns_1 + nf_1\right) + ns_1\left(ns_2 + nf_2\right)}. \quad (28)$$

5.5.2. Scaling Factor Adaptation. For our problem described in Section 3, we can determine that there are possibly many plateaus in the fitness landscape; that is, there are many vectors with the same objective value. As the population converges, the difference between two vectors $|x_1^g - x_2^g|$ tends to be small. To escape from a plateau, various scaling factors are required. We apply the self-adaptive neighbourhood search strategy [19] to set the value of scaling factor:

$$F_k^g = \begin{cases} N_k(0.5, 0.3), & \text{if rand}(0,1) < fp, \\ \delta_k, & \text{otherwise,} \end{cases} \quad (29)$$

where $N_k(0.5, 0.3)$ is a randomly generated number according to a normal distribution of mean 0.5 and standard deviation 0.3, δ_k denotes a Cauchy random number generator with scale parameter $t = 1$, and fp is initialised as 0.5 and updated using (28), as does SaDE.

5.5.3. Crossover Probability Adaptation. The crossover probability cr_k for each individual k at generation g is generated using the procedure in JADE. In JADE, cr_k is generated according to a normal distribution of mean μ_{cr} and standard deviation 0.1,

$$\text{cr}_k = N_k\left(\mu_{\text{cr}}, 0.1\right), \quad (30)$$

and truncated to $[0, 1]$, where μ_{cr} is initialised as 0.5 and at each generation updated according to

$$\mu_{\text{cr}} = (1 - c) \cdot \mu_{\text{cr}} + c \cdot \text{mean}_A\left(S_{\text{cr}}\right), \quad (31)$$

where S_{cr} is the set of all successful cr_i's at generation g and $\text{mean}_A(S_{\text{cr}})$ is the usual arithmetic mean of S_{cr}. We set c as a constant value 0.1.

6. Experimental Studies

6.1. Data Generation. Five datasets are generated with 20, 40, 60, 80, and 100 jobs. In each dataset, 10 instances are generated. Each instance is denoted as n_a, where n is the number of jobs and a is the index of the instance in the dataset with n jobs. Bülbül et al. [22] proposed a method for generating test instances for the single machine earliness/tardiness scheduling problem. We apply this method and add the deadline and setup time generation methods to generate our instances. On each instance, the processing time p_j is a random number generated uniformly in $[p_{\min}, p_{\max}]$; the due date d_j is generated in $[1 - \text{TF} - \text{RDD}/2, 1 - \text{TF} + \text{RDD}/2] * (P_T + \max_i s_{ij} + r_j)$, where P_T is the total processing time of all jobs; TF is the tardiness factor, which is a coarse measure of the proportion of jobs that might be tardy; RDD determines the interval length of the due date; and the term $\max_i s_{ij} + r_j$ is added to p_T to prevent the possible initial infeasibility of d_j. Earliest start date es_j is generated uniformly in $[0, p_j]$, setup time s_{ij} is generated in $[s_{\min}, s_{\max}]$, deadline $\overline{d}_j$ is set as $d_j + (d_j - \text{es}_j)\mu_j$, and revenue e_j is a randomly generated number in $[e_{\min}, e_{\max}]$. These parameters are listed in Table 1.

Table 1: Parameters of the instances.

n	TF	RDD	p	es	u	e	s
20	0.2	0.4	[1, 10]	$[0, p]$	[0.5, 1.5]	[1, 100]	[1, 10]
40	0.4	0.4	[1, 30]	$[0, p]$	[0.5, 1.5]	[1, 100]	[1, 30]
60	0.4	0.4	[1, 30]	$[0, p]$	[0.5, 1.5]	[1, 100]	[1, 30]
80	0.5	0.7	[1, 50]	$[0, p]$	[0.5, 1.5]	[1, 100]	[1, 50]
100	0.5	0.7	[1, 50]	$[0, p]$	[0.5, 1.5]	[1, 100]	[1, 50]

6.2. Experimental Results. Our proposed HDE was implemented in *C#* and run on a Windows 7 laptop with Intel i5 2.4 GHz CPU, 4 GB RAM. The initial population size of all the problems was set to 40. The algorithm stopped if it reached a maximum of 2,000 generations. On each instance, the algorithm ran 10 times ($R = 10$) independently. The maximum, minimum, and average fitness in the 10 runs were recorded. The results of DE [15], NSDE [19], and JADE [18] were also recorded. The results of datasets with different jobs are given in Tables 2–5. The average relative percentage deviation (Δ_{avg}) was calculated as follows:

$$\Delta_{\text{avg}} = \frac{\left(\sum_{h=1}^{R}\left(\left(\left(\text{HDE}_h - \text{REF}_h\right) * 100\right)/\text{REF}_h\right)\right)}{R}, \quad (32)$$

where HDE_h and REF_h are the fitness function values generated by the HDE algorithm and the reference algorithms (including DE, NSDE, and JADE algorithms), respectively, for each run and R is the number of runs. Similarly, $\Delta_{\min}$ and $\Delta_{\max}$ denote the minimum and maximum of the relative percentage deviations, respectively, from the reference fitness function values taken.

6.3. Performance Comparisons. For small problems with 20 jobs, Tables 2 and 3 show that the HDE algorithm performed better than DE and NSDE and slightly better than JADE on all instances. Compared with DE and NSDE, the average relative percentage improvements for HDE are approximately 3 generally, of which the best is 5.08 and the

Table 2: Results of instances with 20/40/60/80/100 jobs.

Inst.	DE			NSDE			JADE			HDE		
#	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.
20_1	836.47	811.95	819.04	851.27	783.545	821.115	843.869	820.49	836.29	**852.341**	**833.416**	**840.963**
20_2	969.11	908.87	953.89	988.67	954.969	967.922	**996.83**	**974.458**	987.751	**996.83**	**974.458**	**994.109**
20_3	**800.08**	754.56	777.57	**800.08**	779.363	793.686	**800.086**	798.966	799.862	**800.086**	**799.526**	**800.03**
20_4	676.93	662.25	670.87	676.93	663.677	669.599	676.932	671.143	674.831	**681.508**	**676.297**	**678.186**
20_5	717.77	695.73	715.56	722.14	673.279	707.92	730.549	**717.772**	722.908	**735.078**	**717.772**	**728.084**
20_6	660.78	643.15	664.63	667.5	623.801	647.91	674.238	662.518	669.152	**676.869**	**665.314**	**670.833**
20_7	919.88	870.99	894.31	889.12	856.146	873.107	922.333	906.014	916.702	**923.28**	**906.2**	**917.493**
20_8	853.73	823.15	839.68	853.32	802.8	836.655	850.445	829.265	842.917	**857.393**	**839.954**	**850.34**
20_9	949.99	928.13	945.05	947.83	892.909	933.954	**959.687**	**947.839**	953.782	**959.687**	**947.839**	**959.687**
20_10	862.58	838.65	849.86	874.1	833.726	846.712	893.635	871.062	875.81	**900.533**	**872.007**	**879.481**
40_1	1595.26	1402.16	1463.04	1712.12	1621.59	1699.5	1717	1687.76	1701.26	**1729.54**	**1693.08**	**1707.02**
40_2	1377.29	1196.84	1266.99	1416.87	1312.61	1386.17	1390.03	1339.62	1371.18	**1433.84**	**1359.49**	**1390.26**
40_3	1626.41	1496.34	1523.3	1735.79	1665.99	1713.65	1735.41	1697.96	1716.73	**1753.6**	**1712.86**	**1730.7**
40_4	1496.03	1344.65	1441.79	1680.41	1623.85	1657.36	1654.68	1616.88	1639.1	**1698.67**	**1630.16**	**1672.43**
40_5	1441.48	1232.3	1357.81	1500.7	1442.24	1482.26	1494.96	1462.92	1477.82	**1508.03**	**1463.55**	**1485.16**
40_6	1467.25	1307.92	1366.92	1542.98	**1501.29**	1518.1	1537.66	1498.75	1512.25	**1601.94**	1499.45	**1535.97**
40_7	1453.56	1162.93	1294.58	1481.54	1411.71	1436.01	1444.77	1366.48	1417.48	**1513.09**	**1439.01**	**1471.98**
40_8	1474.71	1263.84	1341.16	1563.11	1521.93	1541.41	1556.34	1493.89	1524.58	**1647.62**	**1551.47**	**1602.46**
40_9	1588.41	1427.11	1493.87	**1701.77**	1619.24	1654.27	1666.8	1628.73	1641.21	1675.59	**1633.27**	**1656.46**
40_10	1588.45	1498.84	1534.2	1772.63	1723.65	1744.85	1750.21	1720	1737.35	**1791.48**	**1739.41**	**1767.51**
60_1	1859.12	1606.16	1699.93	2035.27	1940.84	1974.44	2014.4	1911.84	1966.63	**2189.82**	**2025.62**	**2082.76**
60_2	1775.26	1601.86	1663.94	1845.34	1719.59	1784.33	1812.69	1712.96	1776.55	**1897.61**	**1775.71**	**1826.88**
60_3	2028.47	1775.78	1876.88	2262.13	2199.62	2226.44	2260.3	2170.49	2215.18	**2353.86**	**2243.52**	**2297.64**
60_4	2029.7	1739.1	1889.56	2202.05	2086.9	2132.79	2170.13	2086.76	2127.79	**2293.52**	**2134.51**	**2197.05**
60_5	1663.97	1415.24	1518.34	1885.15	1790.28	1835.64	1853.82	1767.9	1805.46	**1953.92**	**1810.95**	**1910.9**
60_6	1992.41	1717.8	1812.46	2135.81	2028.63	2065.48	2099.19	2020	2059.51	**2245.29**	**2108.26**	**2164.33**
60_7	1577.11	1361.45	1470.49	1756.22	1694.52	1723.78	1736.44	1668.62	1694.04	**1901.9**	**1721.12**	**1822.51**
60_8	1775.26	1601.86	1663.94	1837.46	1741.04	1782.12	1834.49	1730.91	1772.52	**1897.61**	**1775.71**	**1826.88**
60_9	1807.99	1631.31	1717.5	2016.02	1977.31	1994.65	1995.79	1961.74	1979.25	**2112.76**	**1964.93**	**2052.71**
60_10	1590.36	1312.63	1462.55	1687.3	1630.37	1652.81	1686.2	1624.02	1651.24	**1858.75**	**1673.97**	**1769.35**
80_1	2565.46	2286.64	2429.22	2996.47	2881.69	2927.6	2972.08	2863.14	2911.3	**3206.42**	**3007.91**	**3109.53**
80_2	1928.65	1533.76	1694.58	2136.52	2033.39	2090.27	2107.15	2026.51	2054.53	**2418.29**	**2077.37**	**2257.14**
80_3	2244.23	1875.51	2066.67	2685.5	2519.95	2580.19	2632.86	2482.45	2551.86	**2898.92**	**2711.93**	**2808.54**
80_4	2069.18	1837.77	1948.81	2476.69	2341.81	2401.47	2457.13	2331.49	2390.61	**2703.5**	**2492.65**	**2606.62**
80_5	2450.54	2181.99	2297.12	2893.45	2750.84	2817.56	2883.17	2746.47	2799.28	**3192.97**	**2864.96**	**3025.25**
80_6	2386.89	2204.16	2305.98	2837.29	2733.9	2792.33	2815.91	2727.22	2771	**3100.14**	**2967.28**	**3052.55**
80_7	1399.89	1197.22	1277.5	1681.35	1599.59	1640.23	1646.24	1597.21	1623.29	**1832.94**	**1749.74**	**1806.45**
80_8	1315.23	1060.92	1149.7	1584.4	1476.98	1529.79	1534.29	1452.42	1480.41	**1799.09**	**1645.86**	**1709.29**
80_9	1689.73	1504.58	1587.68	2200.69	2085.73	2139.15	2163.52	2052.65	2113.84	**2519.33**	**2251.68**	**2350.1**
80_10	1876.4	1717.31	1796.99	2290.83	2209.35	2255.55	2302.91	2202.32	2251.91	**2591.89**	**2336.44**	**2456.6**
100_1	3057.05	2760.86	2896.32	3504.57	3398.93	3440.02	3450.02	3371.43	3409.17	**3839.05**	**3573.97**	**3705.15**
100_2	2983.57	2782.57	2874.2	3423.13	3313.85	3368.36	3389.63	3279.75	3345.85	**3692.74**	**3457.92**	**3561.3**
100_3	3072.78	2868.31	2924.7	3526.53	3376.8	3444.16	3519.48	3369.34	3426.93	**3684.51**	**3497.25**	**3582**
100_4	2415.22	2278.91	2343.85	3072.28	2915.9	2979.83	3057.24	2899.56	2988.27	**3434.57**	**3032.15**	**3264.59**
100_5	2399.5	2246.56	2373.22	3011.52	2897.64	2954.22	3008.43	2887.4	2958.51	**3258.9**	**2982.67**	**3166.27**
100_6	2295.1	2095.9	2189.06	2749.53	2652.42	2710.99	2779.42	2663.04	2699.91	**3016.09**	**2812.66**	**2907.65**
100_7	2365.51	2217.68	2282.7	2876.97	2809.52	2849	2896.83	2802.45	2853.45	**3190.67**	**3066.55**	**3130.63**
100_8	2242.73	2093.76	2141.2	2659.71	2601.16	2631.64	2672.85	2603.13	2636.12	**2912.96**	**2734.37**	**2844.21**
100_9	2201.86	2130.07	2166.64	2800.94	2722.91	2765.65	2820.16	2720.1	2780.49	**2976.67**	**2860.61**	**2924.41**
100_10	2194.5	2060.68	2126.02	2682.42	2593.32	2641.54	2716.2	2622.2	2667.54	**2957.39**	**2841.34**	**2893.87**

Table 3: The solution improvements by HDE compared to DE/NSDE/JADE for 20/40 jobs.

Inst.	REF = DE			REF = NSDE			REF = JADE		
#	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}
20_1	1.9	2.64	2.68	0.13	6.36	2.42	1	1.58	0.56
20_2	2.86	7.22	4.22	0.83	2.04	2.71	0	0	0.64
20_3	0	5.96	2.89	0	2.59	0.8	0	0.07	0.02
20_4	0.68	2.12	1.09	0.68	1.9	1.28	0.68	0.77	0.5
20_5	2.41	3.17	1.75	1.79	6.61	2.85	0.62	0	0.72
20_6	2.43	3.45	0.93	1.4	6.65	3.54	0.39	0.42	0.25
20_7	0.37	4.04	2.59	3.84	5.85	5.08	0.1	0.02	0.09
20_8	0.43	2.04	1.27	0.48	4.63	1.64	0.82	1.29	0.88
20_9	1.02	2.12	1.55	1.25	6.15	2.76	0	0	0.62
20_10	4.4	3.98	3.49	3.02	4.59	3.87	0.77	0.11	0.42
20_Ave.	*1.65*	*3.67*	*2.25*	*1.34*	*4.74*	*2.69*	*0.44*	*0.43*	*0.47*
40_1	8.42	20.75	16.68	1.02	4.41	0.44	0.73	0.32	0.34
40_2	4.11	13.59	9.73	1.2	3.57	0.3	3.15	1.48	1.39
40_3	7.82	14.47	13.62	1.03	2.81	0.99	1.05	0.88	0.81
40_4	13.55	21.23	16	1.09	0.39	0.91	2.66	0.82	2.03
40_5	4.62	18.77	9.38	0.49	1.48	0.2	0.87	0.04	0.5
40_6	9.18	14.64	12.37	3.82	−0.12	1.18	4.18	0.05	1.57
40_7	4.1	23.74	13.7	2.13	1.93	2.5	4.73	5.31	3.84
40_8	11.73	22.76	19.48	5.41	1.94	3.96	5.87	3.85	5.11
40_9	5.49	14.45	10.88	−1.54	0.87	0.13	0.53	0.28	0.93
40_10	12.78	16.05	15.21	1.06	0.91	1.3	2.36	1.13	1.74
40_Ave.	*8.18*	*18.04*	*13.7*	*1.57*	*1.82*	*1.19*	*2.61*	*1.42*	*1.83*

Table 4: The solution improvements by HDE compared to DE/NSDE/JADE for 60/80 jobs.

Inst.	REF = DE			REF = NSDE			REF = JADE		
#	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}
60_1	17.79	26.12	22.52	7.59	4.37	5.49	8.71	5.95	5.91
60_2	6.89	10.85	9.79	2.83	3.26	2.38	4.68	3.66	2.83
60_3	16.04	26.34	22.42	4.05	2	3.2	4.14	3.36	3.72
60_4	13	22.74	16.27	4.15	2.28	3.01	5.69	2.29	3.26
60_5	17.43	27.96	25.85	3.65	1.15	4.1	5.4	2.44	5.84
60_6	12.69	22.73	19.41	5.13	3.92	4.79	6.96	4.37	5.09
60_7	20.59	26.42	23.94	8.3	1.57	5.73	9.53	3.15	7.58
60_8	6.89	10.85	9.79	3.27	1.99	2.51	3.44	2.59	3.07
60_9	16.86	20.45	19.52	4.8	−0.63	2.91	5.86	0.16	3.71
60_10	16.88	27.53	20.98	10.16	2.67	7.05	10.23	3.08	7.15
60_Ave.	*14.51*	*22.2*	*19.05*	*5.39*	*2.26*	*4.12*	*6.46*	*3.1*	*4.82*
80_1	24.98	31.54	28.01	7.01	4.38	6.21	7.88	5.06	6.81
80_2	25.39	35.44	33.2	13.19	2.16	7.98	14.77	2.51	9.86
80_3	29.17	44.6	35.9	7.95	7.62	8.85	10.11	9.24	10.06
80_4	30.66	35.63	33.75	9.16	6.44	8.54	10.03	6.91	9.04
80_5	30.3	31.3	31.7	10.35	4.15	7.37	10.75	4.31	8.07
80_6	29.88	34.62	32.38	9.26	8.54	9.32	10.09	8.8	10.16
80_7	30.93	46.15	41.41	9.02	9.39	10.13	11.34	9.55	11.28
80_8	36.79	55.14	48.67	13.55	11.43	11.73	17.26	13.32	15.46
80_9	49.1	49.66	48.02	14.48	7.96	9.86	16.45	9.7	11.18
80_10	38.13	36.05	36.71	13.14	5.75	8.91	12.55	6.09	9.09
80_Ave.	*32.53*	*40.01*	*36.97*	*10.71*	*6.78*	*8.89*	*12.12*	*7.55*	*10.1*

Table 5: The solution improvements by HDE compared to DE/NSDE/JADE for 100 jobs.

Inst.	REF = DE			REF = NSDE			REF = JADE		
#	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}
100_1	25.58	29.45	27.93	9.54	5.15	7.71	11.28	6.01	8.68
100_2	23.77	24.27	23.91	7.88	4.35	5.73	8.94	5.43	6.44
100_3	19.91	21.93	22.47	4.48	3.57	4	4.69	3.8	4.53
100_4	42.21	33.05	39.28	11.79	3.99	9.56	12.34	4.57	9.25
100_5	35.82	32.77	33.42	8.21	2.93	7.18	8.33	3.3	7.02
100_6	31.41	34.2	32.83	9.69	6.04	7.25	8.52	5.62	7.69
100_7	34.88	38.28	37.15	10.9	9.15	9.89	10.14	9.42	9.71
100_8	29.88	30.6	32.83	9.52	5.12	8.08	8.98	5.04	7.89
100_9	35.19	34.3	34.97	6.27	5.06	5.74	5.55	5.17	5.18
100_10	34.76	37.88	36.12	10.25	9.56	9.55	8.88	8.36	8.48
100_Ave.	*31.34*	*31.67*	*32.09*	*8.86*	*5.49*	*7.47*	*8.76*	*5.67*	*7.49*

worst is 0.80. Compared with JADE, the average relative percentage improvements for HDE are all less than 1, of which the best is 0.88 and the worst is 0.03. There is no significant difference between the performance of DE and NSDE, but for small problems with 40 jobs, the maximum and average fitness function values of NSDE and JADE are much better than those for DE, as shown in Table 2. HDE achieves the maximum fitness among the four algorithms. In Table 3, the average relative percentage improvement of HDE compared with DE is appropriately 10 times better than that compared with NSDE and JADE, and JADE and NSDE achieve similar performance. Compared with DE, the average relative percentage deviations for HDE are almost 14 generally, and the minimum relative percentage deviations are nearly 18. Compared with NSDE and JADE, the average relative percentage deviations for HDE are approximately 1.5 generally, and the minimum relative percentage deviations are approximately 1.6.

For medium sized problems with 60 and 80 jobs, compared with NSDE and JADE, our proposed HDE provides significantly better results on instances 80_1, 80_3, 80_4, 80_6, 80_7, 80_8, 80_9, and 80_10, as shown in Tables 1 and 4. On those instances, even the worst fitness function value determined by HDE is better than the best fitness function values determined by NSDE and JADE. On other instances, the maximum and average fitness function values of HDE are better than those of NSDE and JADE. Both NSDE and JADE outperform classical DE; in particular, the average relative percentage improvements by HDE compared with DE are almost five times better than the ones compared to NSDE and JADE. NSDE performs slightly better than JADE in terms of all three indicators.

For large problems with 100 jobs, the results are shown in Tables 1 and 5. Our proposed HDE significantly outperforms DE, NSDE, and JADE. In 10 independent runs, the worst fitness value of HDE is even better than the best fitness value of DE, NSDE, and JADE on instances 100_1, 100_2, 100_6, 100_7, 100_8, 100_9, and 100_10. On the other three instances 100_3, 100_4, and 100_5, the maximum and average fitness of 10 runs are better than those of DE, NSDE, and JADE (the details of result are in http://pan.baidu.com/s/1eSf8jzs).

7. Conclusions

To the best of our knowledge, this is the first attempt to study the OAS problem with time-dependent earliness-tardiness penalties and sequence-dependent setup times, which is inspired by AEOS scheduling. We proposed hybrid DE algorithm under self-adaptation framework, including self-adaptation strategy for mutation operator, the self-adaptive neighbourhood search strategy for scaling factor, and crossover probability adaptation. The significance of self-adaptation framework is proved by the experimental results. Furthermore, we presented a novel method to evaluate the fitness for this problem. Firstly a directed acyclic graph (DAG) is constructed under a given parameter vector, and then the fitness evaluation phase is transformed to finding the longest path from the source node to sink node in the DAG. The graph-based fitness evaluation provides an efficient method for the similar problems. Computational results show that our proposed algorithm outperforms classical DE in addition to two variants: NSDE and JADE. The proposed method not only solves the OASET problem but also suggests an effective way to apply a continuous algorithm to combinatorial optimisation problems. In regard to future research, the dynamic OAS problem catches our attention for considering the orders come stochastically during the planning horizon.

Notations

Indices

i, j: Job index, $i, j = 1, 2, \ldots, n$

Factors

J: Set of all jobs
S: Set of selected jobs, $S \subseteq J$
σ_J: Sorted sequence of jobs in J
σ_S: Sequence of jobs in S
r_j: Release date of job j
es_j: Earliest start date of job j

p_j: Processing time of job j
d_j: Due date of job j, $d_j > \text{es}_j + p_j$; d_j is not necessarily in the middle point of es_j and $\overline{d}_j$
$\overline{d}_j$: Deadline of job j, $\overline{d}_j > d_j$
e_j: Maximum revenue of job j
tw_j: Unit tardiness penalty cost of job j, $tw_j = e_j/(\overline{d}_j - d_j)$
ew_j: Unit earliness penalty cost of job j, $ew_j = e_j/(d_j - (\text{es}_j + p_j))$
s_{ij}: Sequence-dependent setup time for job j being processed immediately after job i $(i \neq j)$
PTW_j: Procession time window of job j, $\text{PTW}_j = [\text{es}_j, \overline{d}_j]$
CTW_j: Completion time window of job j, $\text{CTW}_j = [\text{es}_j + p_j, \overline{d}_j]$
b_j: Procession start time of job j
c_j: Completion time of job j.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant nos. 61603400, 61473301, 61203180, 71501179, and 71501180.

References

[1] C. Oğuz, F. Sibel Salman, and Z. Bilgintürk Yalçin, "Order acceptance and scheduling decisions in make-to-order systems," *International Journal of Production Economics*, vol. 125, no. 1, pp. 200–211, 2010.

[2] S. A. Slotnick, "Order acceptance and scheduling: a taxonomy and review," *European Journal of Operational Research*, vol. 212, no. 1, pp. 1–11, 2011.

[3] P. Guo, W. Cheng, and Y. Wang, "A general variable neighborhood search for single-machine total tardiness scheduling problem with step-deteriorating jobs," *Journal of Industrial and Management Optimization*, vol. 10, no. 4, pp. 1071–1090, 2014.

[4] Y. K. Lin and C. S. Chong, "A tabu search algorithm to minimize total weighted tardiness for the job shop scheduling problem," *Journal of Industrial and Management Optimization*, vol. 12, no. 2, pp. 703–717, 2016.

[5] D. Anghinolfi and M. Paolucci, "A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times," *European Journal of Operational Research*, vol. 193, no. 1, pp. 73–85, 2009.

[6] M. F. Tasgetiren, Q.-K. Pan, and Y.-C. Liang, "A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times," *Computers and Operations Research*, vol. 36, no. 6, pp. 1900–1915, 2009.

[7] B. Cesaret, C. Oğuz, and F. Sibel Salman, "A tabu search algorithm for order acceptance and scheduling," *Computers and Operations Research*, vol. 39, no. 6, pp. 1197–1205, 2012.

[8] S.-W. Lin and K.-C. Ying, "Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm," *Journal of the Operational Research Society*, vol. 64, no. 2, pp. 293–311, 2013.

[9] C. Cheng, Z. Yang, L. Xing, and Y. Tan, "An improved genetic algorithm with local search for order acceptance and scheduling problems," in *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems, CIPLS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, pp. 115–122, sgp, April 2013.

[10] R. Hassin and M. Shani, "Machine scheduling with earliness, tardiness and non-execution penalties," *Computers & Operations Research*, vol. 32, no. 3, pp. 683–705, 2005.

[11] Y.-W. Chen, Y.-Z. Lu, and G.-K. Yang, "Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 36, no. 9-10, pp. 959–968, 2008.

[12] K. R. Baker, "Minimizing earliness and tardiness costs in stochastic scheduling," *European Journal of Operational Research*, vol. 236, no. 2, pp. 445–452, 2014.

[13] D. Shabtay, "Optimal restricted due date assignment in scheduling," *European Journal of Operational Research*, vol. 252, no. 1, pp. 79–89, 2016.

[14] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, vol. 6, no. 5, pp. 367–381, 2002.

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

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

[17] E. M. Arkin and E. B. Silverberg, "Scheduling jobs with fixed start and end times," *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, vol. 18, no. 1, pp. 1–8, 1987.

[18] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[19] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 1110–1116, chn, June 2008.

[20] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[21] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[22] K. Bülbül, P. Kaminsky, and C. Yano, "Preemption in single machine earliness/tardiness scheduling," *Journal of Scheduling*, vol. 10, no. 4-5, pp. 271–292, 2007.