Hindawi
Complexity
Volume 2020, Article ID 8835359, 11 pages
https://doi.org/10.1155/2020/8835359

Research Article

An Enhanced Differential Evolution Algorithm with Fast Evaluating Strategies for TWT-NFSP with SSTs and RTs

Rong Hu,[1,2] **Xing Wu**,[2,3] **Bin Qian**,[1] **Jian L. Mao**,[1] **and Huai P. Jin**[1]

[1]*Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China*
[2]*Faculty of Mechanical & Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China*
[3]*Yunnan Vocational College of Mechanical and Electrical Technology, Kunming 650500, China*

Correspondence should be addressed to Xing Wu; xwu@kust.edu.cn

Received 19 August 2020; Revised 30 September 2020; Accepted 26 October 2020; Published 19 November 2020

Academic Editor: Shubo Wang

The no-wait flow-shop scheduling problem with sequence-dependent setup times and release times (i.e., the NFSP with SSTs and RTs) is a typical NP-hard problem. This paper proposes an enhanced differential evolution algorithm with several fast evaluating strategies, namely, DE_FES, to minimize the total weighted tardiness objective (TWT) for the NFSP with SSTs and RTs. In the proposed DE_FES, the DE-based search is adopted to perform global search for obtaining the promising regions or solutions in solution space, and a fast local search combined with three presented strategies is designed to execute exploitation from these obtained regions. Test results and comparisons with two effective meta-heuristics show the effectiveness and robustness of DE_FES.

1. Introduction

The no-wait flow-shop scheduling problem (NFSP) has been widely studied for more than 30 years [1, 2]. Nevertheless, literature reviews on production scheduling with no-wait or setup constraints in [1, 3] manifest that the NFSP with sequence-dependent setup times (SSTs) and release times (RTs) has not received the attention it needs. In fact, SSTs and RTs are two kinds of constraints widely existing in the real-life NFSPs. These constraints can be found in pharmaceutical processing, metal processing, and chemical processing [4–8]. Moreover, in the current increasingly fierce global competition, many enterprises are trying to reduce the tardy jobs in order to maintain customer satisfaction and avoid the loss of customer orders. Hence, it is important to design an effective algorithm to minimize the total weighted tardiness objective (TWT) for the NFSP with SSTs and RTs (i.e., $F_m/\text{no}-\text{wait}, \text{ST}_{sd}, r_j/\sum w_j T_j$).

Optimization algorithms have been successfully used to deal with a variety of important engineering problems [9–14]. In recent years, evolutionary algorithms have become an important class of optimization algorithms for addressing production scheduling problems [15–19]. Differential evolution (DE) is a very competitive evolutionary algorithm for solving continuous optimization problem [20–22]. It iteratively executes three key operators, i.e., *mutation*, *crossover*, and *selection*. Due to its easy implementation, simple mechanism, and quick convergence, DE has been applied to addressing various optimization problems in academia and industry. Nevertheless, due to its continuous nature, the studies on DE for combinatorial optimization problems are restricted. Tasgetiren et al. [23] developed a novel DE via introducing the *interchange*-based local search and the smallest position value (SPV) rule for the flow-shop scheduling problems (FSPs), whose criterion is to minimize makespan. Onwubolu et al. [24] presented a new approach based on DE for the FSPs. They considered three criteria, i.e., mean flowtime, makespan, and total tardiness. Pan et al. [25] designed a discrete DE to address the FSPs effectively. Qian et al. [26] proposed a hybrid DE (HDE) for the NFSP with makespan criterion. By reasonably utilizing the corresponding problem's properties, HDE is of excellent search ability. Wang et al. [27] designed an effective discrete DE for the blocking FSPs. Hu et al. [28] designed an

effective DE, namely, DE_NTJ, for the NFSP with SSTs and RTs. The criterion is to minimize the number of tardy jobs. Qian et al. [29] developed a DE with two speed-up methods (DE_TSM) to minimize the total completion time for the NFSP with SSTs and RTs. Tang et al. [30] presented an improved DE (IDE) to deal with a dynamic scheduling problem in steelmaking-continuous casting production. IDE can obtain better performance than the compared algorithms. Santucci et al. [31] proposed an algebraic DE for the FSPs with total flowtime criterion. Wu and Che [32] developed a memetic DE (MDE) to solve the biobjective unrelated parallel machine scheduling problem. In their tests, MDE outperforms two famous multiobjective algorithms. However, according to the existing literatures, there are no studies on $F_m/\text{no}-\text{wait}, \text{ST}_{\text{sd}}, r_j/\sum w_j T_j$. Therefore, it is imperative to develop an efficient algorithm for the TWT-NFSP with SSTs and RTs. The developed algorithm can be easily extended for other kinds of flow-shop scheduling problems with no-wait constraint.

This paper proposes an efficient DE-based algorithm with fast evaluating strategies (DE_FES) to minimize the total weighted tardiness for the NFSP with SSTs and RTs. Compared with the existing research, the main contributions of this work are summarized as follows:

(1) The permutation-based model for the TWT-NFSP with SSTs and RTs is formulated for the first time. Moreover, the NP-hardness of this problem is proved via a reduction chain starting from a known NP-hard problem.

(2) Three kinds of properties of the TWT-NFSP with SSTs and RTs are analyzed in detail and are utilized to speed up the search process. It is the first time to analyze the low-bound property (see Subsection 3.4) of the neighbors in the *interchange*-based neighbourhood for the NFSP with RTs.

(3) To improve the exploitation ability of DE_FES, a fast local search combing three property-based strategies is presented to perform the efficient search in the promising regions obtained by the DE-based global search.

The remaining part of this paper is organized as follows. In Section 2, the TWT-NFSP with SSTs and STs is formulated. In Section 3, three fast solution evaluating strategies and some theoretical analyses are presented. In Section 4, DE_FES is proposed after introducing its main components. In Section 5, the test results are provided. Finally, conclusions and future research are given in Section 6.

2. TWT-NFSP with SSTs and RTs

The NFSP with SSTs and RTs is described as follows. There are n jobs to be processed on m machines in the same order. Each machine can only process one job at a time. Each job must be processed without waiting time between consecutive operations. Setup times depend on the sequence of the jobs. If the job has not been released, the machine remains idle until it is released.

2.1. NFSP with SSTs and RTs. Denote $\pi = [j_1, j_2, \ldots, j_n]$ is the permutation or schedule to be processed, $p_{j_i,l}$ is the processing time of job j_i on machine l, sp_{j_i} is the total processing time of job j_i on m machines, $\text{ML}_{j_i,l}$ is the minimum delay time between j_{i-1} and j_i on the machine l, L_{j_{i-1},j_i} is the minimum delay time between j_{i-1} and j_i on the first machine, $s_{j_{i-1},j_i,l}$ is the sequence-dependent setup time between j_{i-1} and j_i on machine l, r_{j_i} is the arrival time of j_i, St_{j_i} is the start processing time of j_i on the first machine, and C_{j_i} is the completion time of j_i on the last machine. Let $p_{j_0,l} = 0$ for $l = [1, \ldots, m]$. Then, $\text{ML}_{j_i,l}$ can be calculated as follows:

$$\text{ML}_{j_i,l} = \begin{cases} \max\left\{s_{j_{i-1},j_i,1} + p_{j_i,1} - p_{j_{i-1},2}, s_{j_{i-1},j_i,2}\right\} + p_{j_i,2}, & l = 2, \\ \max\left\{\text{ML}_{j_i,l-1} - p_{j_{i-1},l}, s_{j_{i-1},j_i,l}\right\} + p_{j_i,l}, & l = 3, \ldots, m. \end{cases} \tag{1}$$

By using (1), L_{j_{i-1},j_i} can be calculated with the following formula:

$$L_{j_{i-1},j_i} = \text{ML}_{j_i,m} + \text{sp}_{j_{i-1}} - \text{sp}_{j_i}. \tag{2}$$

Obviously, St_{j_i} can be written as follows:

$$\text{St}_{j_i} = \begin{cases} \max\left\{\text{ML}_{j_i,m} - \text{sp}_{j_i}, r_{j_i}\right\}, & i = 1, \\ \text{St}_{j_{i-1}} + \max\left\{L_{j_{i-1},j_i}, r_{j_i} - \text{St}_{j_{i-1}}\right\} = \max\left\{\text{St}_{j_{i-1}} + L_{j_{i-1},j_i}, r_{j_i}\right\}, & i = 2, \ldots, n. \end{cases} \tag{3}$$

Thus, C_{j_i} can be calculated as follows:

$$C_{j_i} = \text{St}_{j_i} + \text{sp}_{j_i}, \quad i = 1, \ldots, n. \tag{4}$$

Figure 1(a) gives an example of the NFSP with SSTs when $n = 4$ and $m = 2$. It can be seen that $\text{ML}_{j_i,l}$ and L_{j_{i-1},j_i} are determined by the model of the NFSP with SSTs. Figure 1(b) gives an example of the NFSP with SSTs and RTs when $n = 4$ and $m = 2$. It can be seen that St_{j_i} and C_{j_i} are determined by the model of the NFSP with SSTs and RTs.

2.2. TWT-NFSP with SSTs and RTs. Denote d_{j_i} is the due date of j_i, w_{j_i} is the weighted value of j_i, and Π is the set of all permutations. Then, $\text{TWT}(\pi)$ can be calculated as follows:

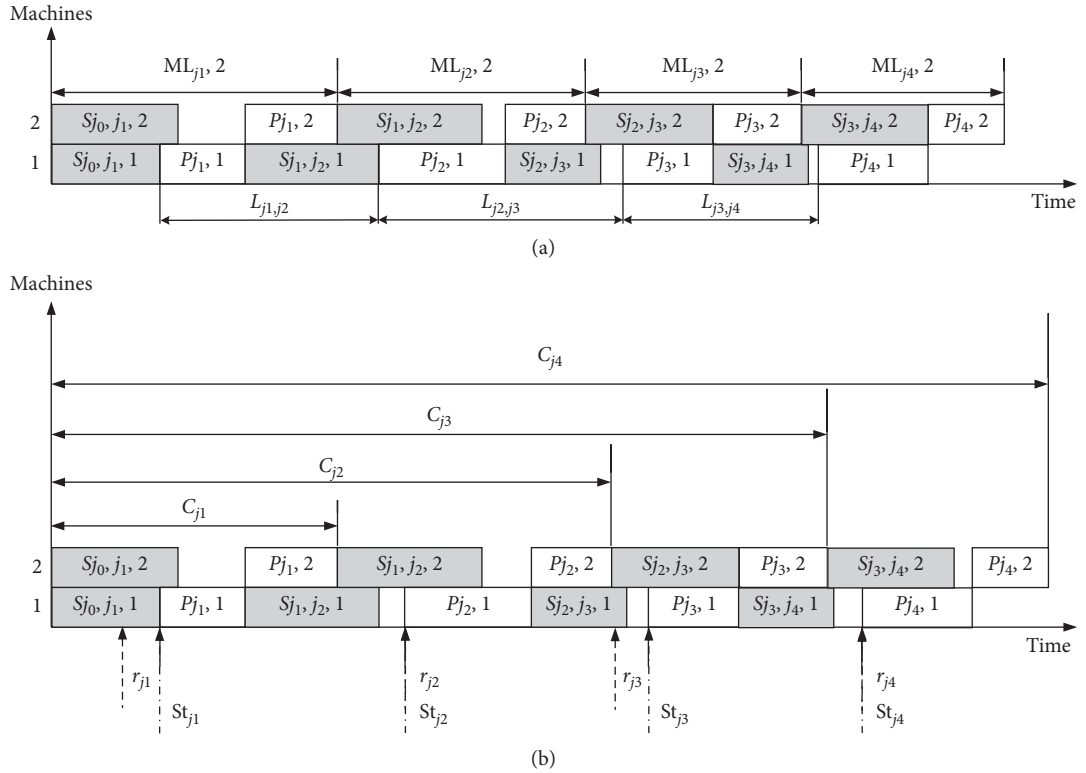

FIGURE 1: Two examples when $n = 4$ and $m = 2$. (a) NFSSP with SDSTs when $n = 4$ and $m = 2$. (b) NFSSP with SDSTs and RDs when $n = 4$ and $m = 2$.

$$\begin{aligned} T_{j_i} &= \max\left\{C_{j_i} - d_{j_i}, 0\right\} \\ &= \max\left\{\mathrm{St}_{j_i} + \mathrm{sp}_{j_i} - d_{j_i}, 0\right\}, \quad i = 1, \ldots, n, \end{aligned} \tag{5}$$

$$\begin{aligned} \mathrm{TWT}(\boldsymbol{\pi}) &= \sum_{i=1}^{n} w_{j_i} T_{j_i} = \sum_{i=1}^{n} w_{j_i} \max\left\{C_{j_i} - d_{j_i}, 0\right\} \\ &= \sum_{i=1}^{n} w_{j_i} \max\left\{\mathrm{St}_{j_i} + \mathrm{sp}_{j_i} - d_{j_i}, 0\right\}. \end{aligned} \tag{6}$$

The TWT-NFSP with SSTs and RTs is to find an optimal permutation π^* in Π such that

$$\boldsymbol{\pi}^* = \arg\{\mathrm{TWT}(\boldsymbol{\pi})\} \longrightarrow \min, \quad \forall \boldsymbol{\pi} \in \Pi. \tag{7}$$

2.3. Problem Complexity. The TWT-NFSP with SSTs and RTs can be described by a triplet $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}, r_j/\sum w_j T_j$. The NP-hardness of this problem can be determined by using a reduction chain starting from a known NP-hard problem, i.e., $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ [3, 8]. Problem A reduces to problem B (denoted $A \propto B$) means A is just a special case of B, and B is at least as difficult to solve as A [2].

Theorem 1. *$F_m/no-wait, ST_{sd}, r_j/\sum w_j T_j$ is NP-hard.*

Proof. Obviously, $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ reduces to $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ by setting $r_j = 0$ for all j, and $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ reduces to $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ by setting $w_j = 0$ and $d_j = 0$ for all j. Since each of the above reductions is completed in a polynomial time, a polynomial Turing reduction chain can be established as follows:

$$\begin{aligned} &F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j \propto F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j \propto \\ &\quad F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j, r_j/\sum w_j T_j. \end{aligned} \tag{8}$$

By using (8), it can be concluded that $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}, r_j/\sum w_j T_j$ is NP-hard because $F_m/\mathrm{no-wait}, \mathrm{ST_{sd}}/\sum C_j$ is NP-hard [3, 8]. Theorem 1 holds.

The reductions between objective functions are given in Figure 2. It can be seen from Figure 2 that $\sum w_j T_j$ is one of the most difficult criteria. Based on the reduction concept in [2], all the other types of no-wait flow-shop scheduling problems reduce to the NFSP with SSTs and RTs and almost all the other objective functions reduce to TWT (see Figure 2), which means the TWT-NFSP with SSTs and RTs is a most difficult one and can be utilized to represent a wide range of real-life scheduling problems. Thus, the study on the TWT-NFSP with SSTs and RTs has general and theoretical meanings.

3. Three Fast Evaluating Strategies

The reasonable utilization of problem-dependent properties is very useful for designing effective algorithms [33]. So, this section analyses the properties of the considered problem

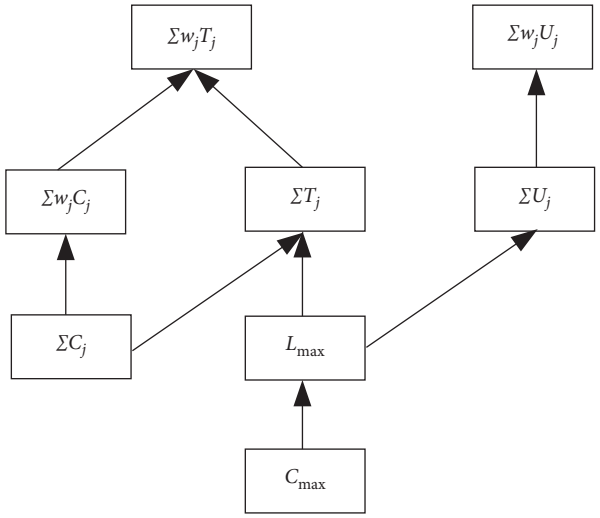

Figure 2: Basic reductions between objective functions in [2].

and presents three fast evaluating strategies. Strategy one is based on the general property of the NFSP. Strategies two and three are deduced from the special properties of *interchange*'s neighbourhood under the NFSP with SSTs and RTs. Strategy one is used to compute the fitness of each individual in DE_FES, and strategies one to three are utilized to design an efficient neighbourhood-based search.

3.1. Strategy One: Fast Computing Strategy. In the model of the NFSP with SSTs and RTs, L_{j_{i-1},j_i} is only decided by j_{i-1} and j_i. By utilizing this property, the computing complexity (CC) of $\mathrm{TWT}(\pi)$ can be reduced. That is, L_{j_{i-1},j_i}, sp_{j_i}, and $\sum_{i=1}^{n} \mathrm{sp}_{j_i}$ are calculated and recorded at DE_FES's initial stage, and then they are treated as constant values at DE_FES's evolution stage. This strategy reduces the CC of $\mathrm{TWT}(\pi)$ from $O(nm)$ to $O(n)$.

3.2. The Interchange-Based Neighbourhood. According to [34], the diameter of *interchange* is $n-1$. That is to say, one solution π can transit to any other solution by utilizing *interchange* at most $n-1$ times. The diameter of *interchange* is one of the shortest among those most-used neighbourhoods. That is, the neighbors or solutions generated by *interchange* are closer to each other. So, this subsection chooses to analyze and utilize the properties of *interchange*'s neighbourhood.

The *interchange*-based neighbourhood of π can be given as

$$N_{\mathrm{interchange}}(\boldsymbol{\pi}) = \left\{\boldsymbol{\pi}^{n,u,v}_{\mathrm{interchange}} = \mathrm{interchange}\ (\boldsymbol{\pi}, u, v) | 1 \le u < v \le n\right\}, \tag{9}$$

where $\mathrm{interchange}(\pi, u, v)$ denotes the interchange of j_u and j_v. The size of $N_{\mathrm{interchange}}(\pi)$ is $n(n-1)/2$. Figure 3 shows a small example of $\pi^{n,u,v}_{\mathrm{interchange}}$ when $n = 10$, $u = 4$, and $v = 7$.

3.3. Strategy Two: Fast Scanning Strategy for $N_{interchange}(\pi)$. A speed-up neighbourhood scanning strategy is presented in this subsection. This strategy is necessary for devising a fast local search.

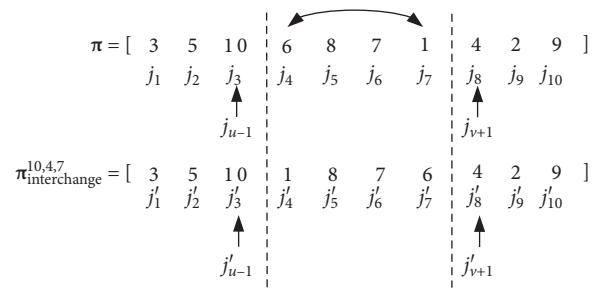

Figure 3: A small example of $\pi^{n,u,v}_{\mathrm{interchange}}$ when $n = 10$, $u = 4$, and $v = 7$.

Denote Find Best $N_{\mathrm{interchange}}(\pi)$ is the method of obtaining the best neighbor with TWT criterion in $N_{\mathrm{interchange}}(\pi)$ (i.e., the $N_{\mathrm{interchange}}(\pi)$-based neighbourhood scanning method). When $u = 1,\ldots,n-1$ and $i = 1,\ldots,u-1$, it has $j_i = j'_i$, $w_{j_{u-1}} T_{j_{u-1}} = w_{j_{u-1}'} T_{j_{u-1}'}$, and $\sum_{i=1}^{u-1} w_{j'_i} T_{j'_i} = \sum_{i=1}^{u-1} w_{j_i} T_{j_i}$. Then, according to (5) and (6), it has

$$\begin{aligned} \mathrm{TWT}\left(\boldsymbol{\pi}^{n,u,v}_{\mathrm{interchange}}\right) &= \sum_{i=1}^{u-1} w_{j'_i} T_{j'_i} + \sum_{i=u}^{n} w_{j'_i} T_{j'_i} \\ &= \sum_{i=1}^{u-1} w_{j_i} T_{j_i} + \sum_{i=u}^{n} w_{j'_i} T_{j'_i}. \end{aligned} \tag{10}$$

It is clear from (10) that in Find Best $N_{\mathrm{interchange}}(\pi)$, St_{j_i} and $\sum_{ll=1}^{i} w_{j_{ll}} T_{j_{ll}}$ $(i = 1,\ldots,n-1)$ can be computed and recorded before evaluating or scanning each neighbor in $N_{\mathrm{interchange}}(\pi)$, and then they can be treated as constant values when evaluating $\mathrm{TWT}(\pi^{n,u,v}_{\mathrm{interchange}})$. Specifically, if $u > 1$, $\mathrm{St}_{j_{u-1}'}$ and $\sum_{i=1}^{u-1} w_{j'_i} T_{j'_i}$ can be directly replaced by $\mathrm{St}_{j_{u-1}}$ and $\sum_{i=1}^{u-1} w_{j_i} T_{j_i}$, respectively, and $\mathrm{St}_{j'_u}$ can be calculated from $\mathrm{St}_{j_{u-1}}$. When scanning each neighbor in $N_{\mathrm{interchange}}(\pi)$, only $\sum_{i=u}^{n} w_{j'_i} T_{j'_i}$ needs to be calculated. So, by using strategies one and two, the CC of Find Best $N_{\mathrm{interchange}}(\pi)$ can be decreased to some extent.

3.4. Strategy Three: Fast Nonimproving Neighbor Identification Strategy for $N_{interchange}(\pi)$. According to the $N_{\mathrm{interchange}}(\pi)$-based neighbourhood properties, two lemmas are stated and then are used in the proof of Theorem 2. By utilizing Theorem 2, the nonimproving neighbors in $N_{\mathrm{interchange}}(\pi)$ can be identified with lower CC, which means the CC of Find Best $N_{\mathrm{interchange}}(\pi)$ can be further reduced.

Denote $\Delta\mathrm{St}_{j_y}$ $(y = 1, 2, \ldots, n-1)$ and $\Delta\mathrm{St}_{j_y} \ge 0)$ is a start processing time delay of j_y on the first machine, St'_{j_y} is the start processing time of j_y on the first machine when $\Delta\mathrm{St}_{j_y}$ is added to St_{j_y} (i.e., $\mathrm{St}'_{j_y} = \mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} \ge \mathrm{St}_{j_y}$), $\mathrm{St}'_{j_{y+ll}}$ $(ll = 1, \ldots, n-y)$ is the start processing time of job j_{y+ll} on the first machine when St_{j_y} is set to St'_{j_y} and $T'_{j_{y+ll}}$ $(ll = 1, \ldots, n-y)$ is the tardiness of job j_{y+ll} on the first machine when St_{j_y} is set to St'_{j_y}.

Lemma 1. *For a fixed job permutation $\pi = (j_1, j_2, \ldots, j_n)$ of an NFSP with SSTs and RTs, if St_{j_y} is set to St'_{j_y}, $(St'_{j_y} \ge St_{j_y})$, then it has $St'_{j_{y+ll}} \ge St_{j_{y+ll}}$ for $ll = 1, \ldots, n-y$.*

Proof. Let $\Delta_{y+ll} = \mathrm{St}'_{j_{y+ll}} - \mathrm{St}_{j_{y+ll}} (ll = 1, \ldots, n-y)$ denote the start processing time delay of j_{y+ll}. Lemma 1 can be proved by using mathematical induction.

Firstly, when $ll = 1$, with (3), it has

$$\begin{aligned}\mathrm{St}'_{j_{y+1}} &= \mathrm{St}'_{j_y} + \max\left\{L_{j_y,j_{y+1}}, r_{j_{y+1}} - \mathrm{St}'_{j_y}\right\} \\ &= \max\left\{\mathrm{St}'_{j_y} + L_{j_y,j_{y+1}}, r_{j_{y+1}}\right\} \\ &= \max\left\{\mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} + L_{j_y,j_{y+1}}, r_{j_{y+1}}\right\}.\end{aligned} \tag{11}$$

Obviously, when $\Delta\mathrm{St}_{j_y} = 0$, it has $\mathrm{St}'_{j_{y+1}} = \mathrm{St}_{j_{y+1}}$.

When $\Delta\mathrm{St}_{j_y} > 0$, based on the value of $r_{j_{y+1}}$, three cases are discussed as follows:

Case 1. $r_{j_{y+1}} \le St_{j_y} + L_{j_y,j_{y+1}}$.

Under this case, with (11) and (3), it has

$$\mathrm{St}'_{j_{y+1}} = \mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} + L_{j_y,j_{y+1}} = \mathrm{St}_{j_{y+1}} + \Delta\mathrm{St}_{j_y} > \mathrm{St}_{j_{y+1}}. \tag{12}$$

Case 2. $\mathrm{St}_{j_y} + L_{j_y,j_{y+1}} < r_{j_{y+1}} < \mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} + L_{j_y,j_{y+1}}$.

Under this case, with (11) and (3), it has

$$\begin{aligned}\mathrm{St}'_{j_{y+1}} &= \mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} + L_{j_y,j_{y+1}} > r_{j_{y+1}} \\ &= \max\left\{\mathrm{St}_{j_y} + L_{j_y,j_{y+1}}, r_{j_{y+1}}\right\} = \mathrm{St}_{j_{y+1}}.\end{aligned} \tag{13}$$

Case 3. $r_{j_{y+1}} \ge \mathrm{St}_{j_y} + \Delta\mathrm{St}_{j_y} + L_{j_y,j_{y+1}}$.

Under this case, with (11) and (3), it has

$$\mathrm{St}'_{j_{y+1}} = r_{j_{y+1}} = \max\left\{\mathrm{St}_{j_y} + L_{j_y,j_{y+1}}, r_{j_{y+1}}\right\} = \mathrm{St}_{j_{y+1}}, \tag{14}$$

and with (12)–(14), it has $\mathrm{St}'_{j_{y+1}} \ge \mathrm{St}_{j_{y+1}}$.

Secondly, suppose $\mathrm{St}'_{j_{y+ll}} \ge \mathrm{St}_{j_{y+ll}}$ holds when $1 \le ll \le k$. Thus, $\Delta_{y+k} = \mathrm{St}'_{j_{y+k}} - \mathrm{St}_{j_{y+k}} \ge 0$. With (3) and (11), it has

$$\begin{aligned}\mathrm{St}'_{j_{y+k+1}} &= \max\left\{\mathrm{St}'_{j_{y+k}} + L_{j_{y+k},j_{y+k+1}}, r_{j_{y+k+1}}\right\} \\ &= \max\left\{\mathrm{St}_{j_{y+k}} + \Delta_{y+k} + L_{j_{y+k},j_{y+k+1}}, r_{j_{y+k+1}}\right\}.\end{aligned} \tag{15}$$

Similar to the analysis when $ll = 1$, it has $\mathrm{St}'_{j_{y+k+1}} \ge \mathrm{St}_{j_{y+k+1}}$.

Thus, based on mathematical induction, $\mathrm{St}'_{j_{y+ll}} \ge \mathrm{St}_{j_{y+ll}}$ holds when $1 \le ll \le n-y$.

Lemma 2. *For a fixed job permutation $\pi = [j_1, j_2, \ldots, j_n]$ of an NFSP with SSTs and RTs, if St_{j_y} is set to St'_{j_y} $(St'_{j_y} \ge St_{j_y})$, then it has $T'_{j_{y+ll}} \ge T_{j_{y+ll}}$ for $ll = 1, \ldots, n-y$.*

Proof. According to Lemma 1, when $\mathrm{St}'_{j_y} \ge \mathrm{St}_{j_y}$, it has $\mathrm{St}'_{j_{y+ll}} \ge \mathrm{St}_{j_{y+ll}}$ for $ll = 1, \ldots, n-y$. Then, with (5), it has

$$T'_{j_{y+ll}} = \max\left\{\mathrm{St}'_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}}, 0\right\}, \tag{16}$$

$$T_{j_{y+ll}} = \max\left\{\mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}}, 0\right\}. \tag{17}$$

Obviously, if $\mathrm{St}'_{j_y} = \mathrm{St}_{j_y}$, it has $T'_{j_{y+ll}} = T_{j_{y+ll}}$ for $ll = 1, \ldots, n-y$.

If $\mathrm{St}'_{j_y} > \mathrm{St}_{j_y}$, according to the value of $d_{j_{y+ll}}$, three cases are discussed as follows:

Case 1. $d_{j_{y+ll}} \le \mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}}$.

Under this case, with (16) and (17), it has

$$\begin{aligned}T'_{j_{y+ll}} &= \mathrm{St}'_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}} > \mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}} \\ &= \max\left\{\mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}}, 0\right\} = T_{j_{y+ll}}.\end{aligned} \tag{18}$$

Case 2. $\mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} < d_{j_{y+ll}} < \mathrm{St}'_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}}$.

Under this case, with (16) and (17), it has

$$\begin{aligned}T'_{j_{y+ll}} &= \mathrm{St}'_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}} > 0 \\ &= \max\left\{\mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}}, 0\right\} = T_{j_{y+ll}}.\end{aligned} \tag{19}$$

Case 3. $d_{j_{y+ll}} \ge \mathrm{St}'_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}}$.

Under this case, with (16) and (17), it has

$$T'_{j_{y+ll}} = 0 = \max\left\{\mathrm{St}_{j_{y+ll}} + \mathrm{sp}_{j_{y+ll}} - d_{j_{y+ll}}, 0\right\} = T_{j_{y+ll}}. \tag{20}$$

Based on the above analysis, Lemma 2 holds.

Theorem 2. *If $v < n$ and $\sum_{i=u}^{v} w_{j_i} T'_{j_i} \ge \sum_{i=u}^{v} w_{j_i} T_{j_i}$ and $St_{j_{v+1}} \ge St_{j_{v+1}}$, then $TWT(\pi^{n,u,v}) \ge TWT(\pi)$.*

Proof. According to the values of u, two cases are discussed as follows:

Case 1. $u > 1$.

From (6) and (10), it has

$$\mathrm{TWT}(\boldsymbol{\pi}) = \sum_{i=1}^{u-1} w_{j_i} T_{j_i} + \sum_{i=u}^{v} w_{j_i} T_{j_i} + \sum_{i=v+1}^{n} w_{j_i} T_{j_i}, \tag{21}$$

$$\mathrm{TWT}(\boldsymbol{\pi}^{n,u,v}) = \sum_{i=1}^{u-1} w_{j'_i} T_{j'_i} + \sum_{i=u}^{v} w_{j'_i} T_{j'_i} + \sum_{i=v+1}^{n} w_{j'_i} T_{j'_i}. \tag{22}$$

Since $j'_i = j_i$ for $i = 1, \ldots, u-1$, it has

$$\sum_{i=1}^{u-1} w_{j_i'} T_{j_i'} = \sum_{i=1}^{u-1} w_{j_i} T_{j_i}. \tag{23}$$

As $j_i' = j_i$ for $i = v+1, \ldots, n$ and $\mathrm{St}_{j_{v+1}'} \geq \mathrm{St}_{j_{v+1}}$, according to Lemma 2, it has $T_{j_i'} \geq T_{j_i}$ for $i = v+1, \ldots, n$. Then, it follows that

$$\sum_{i=v+1}^{n} w_{j_i'} T_{j_i'} \geq \sum_{i=v+1}^{n} w_{j_i} T_{j_i}. \tag{24}$$

Finally, because $\sum_{i=u}^{v} w_{j_i'} T_{j_i'} \geq \sum_{i=u}^{v} w_{j_i} T_{j_i}$, from (21)–(24), it has

$$\begin{aligned} \mathrm{TWT}\left(\boldsymbol{\pi}^{n,u,v}\right) - \mathrm{TWT}\left(\boldsymbol{\pi}\right) = {} & \sum_{i=u}^{v} w_{j_i'} T_{j_i'} - \sum_{i=u}^{v} w_{j_i} T_{j_i} + \sum_{i=v+1}^{n} w_{j_i'} T_{j_i'} \\ & - \sum_{i=v+1}^{n} w_{j_i} T_{j_i} \geq 0. \end{aligned} \tag{25}$$

Case 2. $u = 1$.

Under this case, from (6) and (10), it has

$$\mathrm{TWT}\left(\boldsymbol{\pi}\right) = \sum_{i=u}^{v} w_{j_i} T_{j_i} + \sum_{i=v+1}^{n} w_{j_i} T_{j_i}, \tag{26}$$

$$\mathrm{TWT}\left(\boldsymbol{\pi}^{n,u,v}\right) = \sum_{i=u}^{v} w_{j_i'} T_{j_i'} + \sum_{i=v+1}^{n} w_{j_i'} T_{j_i'}. \tag{27}$$

Since $\sum_{i=u}^{v} w_{j_i'} T_{j_i'} \geq \sum_{i=u}^{v} w_{j_i} T_{j_i}$, with (24), (26), and (27), it has $\mathrm{TWT}(\pi^{n,u,v}) - \mathrm{TWT}(\pi) \geq 0$.

Based on the above analysis, Theorem 2 holds.

By utilizing Theorem 2, after $\sum_{i=u}^{v} w_{j_i'} T_{j_i'}$ and $\mathrm{St}_{j_{v+1}'}$ have been computed, the nonimproving neighbors can be detected without calculating $\sum_{i=v+1}^{n} w_{j_i'} T_{j_i'}$ if the conditions in Theorem 2 are satisfied. So, Theorem 2 is helpful in reducing the CC of Find Best $N_{\mathrm{interchange}}(\pi)$. Let $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$ denote Find Best $N_{\mathrm{interchange}}(\pi)$ with the above three strategies. Note that $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$ is a key component in the local search of our proposed DE_FES.

4. DE_FES for TWT-NFSP with SSTs and RTs

In this section, DE_FES's main components designed for addressing the TWT-NFSP with SSTs and RTs are explained as follows.

4.1. Solution Representation. As mentioned in Section 1, the original DE cannot be directly used to solve the TWT-NFSP with SSTs and RTs. Hence, a largest-order-value (LOV) rule in [35] is adopted to transform the ith individual $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ of DE to the job permutation vector $\pi_i = \{j_{i,1}, j_{i,2}, \ldots, j_{i,n}\}$. Based on LOV rule, $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ are firstly ranked by descending order to obtain the temporary sequence $\varphi_i = [\varphi_{i,1}, \varphi_{i,2}, \ldots, \varphi_{i,n}]$. Then, π_i can be obtained by using the following formula:

$$j_{i,\varphi_{i,k}} = k. \tag{28}$$

It is worth mentioning that the LOV rule achieved better results than the famous random key rule in our previous tests.

4.2. DE-Based Global Search. Since DE exhibited strong global search ability in many previous studies, DE_FES's global search is devised according to the *DE/rand-to-best/1/exp* scheme [28, 29, 35, 36], in which the base vector is set to the best of all individuals. This means the valuable information of the best individual can be shared among all individuals during the evolution process.

4.3. Fast Local Search. Denote insert(π, u, v) is the operator of inserting j_u in the v*th* dimension of π. The pseudocode of our fast local search is provided as follows (Algorithm 1).

In Algorithm 1, Step 2 is utilized to avoid plunging into local optima and it leads the search to a quite different region. In Step 3, three proposed strategies in Section 3 are adopted in $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$, which can help the local search reach more regions in the same running time. Hence, Step 3 performs a deep and efficient exploitation from the region found by Step 2. It is worth noting that $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$, designed according to the properties of the problem, is the key part that distinguishes the above local search from the local searches in the existing hybrid DE-based algorithms. This is the first time to analyze the low-bound property of the neighbors in $N_{\mathrm{interchange}}(\pi)$ (see Theorem 2 in Subsection 3.4) and utilize this property to reduce the CC of local search. By utilizing the low-bound property (corresponding to strategy three) to further reduce the CC of neighbor evaluation, all neighbors in $N_{\mathrm{interchange}}(\pi)$ can be evaluated in a very short time. Obviously, searching the whole neighbourhood with low CC is an efficient and ideal search mode. Therefore, $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$ adopts the "comprehensive neighbourhood search scheme" similar to that in [29], instead of the "first-improvement-move neighbourhood search scheme" in [28] and the "random variable neighbourhood search scheme" in [35]. Meanwhile, due to the use of the low-bound property, the CC of $S123$_Find Best $N_{\mathrm{interchange}}(\pi)$ is smaller than that of the neighbourhood search in [29], which allows DE_FES to have a more efficient local search engine.

4.4. DE_FES. Based on Subsections 4.1–4.3, the pseudocode of DE_FES is given as follows (Algorithm 2).

It can be seen from Algorithm 2 that DE_FES remains the basic characteristic of the original DE. In DE_FES, DE's standard crossover and mutation are utilized to generate candidate individuals, and DE's competition scheme is still used to get new individuals. Meanwhile, DE_FES also adopts the LOV rule, fast computing strategy, and fast local search with three proposed strategies to make DE suitable for

Step 1: transform individual $X_i(t)$ to π_{i_0} via the LOV rule.
Step 2: perturbation phase.
  Set $\pi_{i\_t} = \pi_{i\_0}$.
  For $tt = 1$ to KK
    Randomly select u and v, where $|u - v| > n/3$;
    π_i = insert (π_{i_t}, u, v);
    $\pi_{i\_t} = \pi_i$;
  End.
Step 3: exploitation phase.
  Set $ll = 0$;
  Do
    π_{i_1} = $S123$_Find Best $N_{\text{interchange}}(\pi_i)$;
    If $f(\pi_{i\_1}) < f(\pi_i)$, then
      $\pi_i = \pi_{i\_1}$;
    else
      ll++;
    end;
  While $ll < 1$.
Step 4: if $f(\pi_i) \le f(\pi_{i\_0})$, then $\pi_{i\_0} = \pi_i$.
Step 5: transform π_{i_0} back to $X_i(t)$.

Algorithm 1: Fast local search.

Step 0: denote CR is the crossover probability, random (0, 1) is the random value in the interval (0, 1), $t_$max is the maximum generation, t is a generation, Pop(t) is the population with size N_p at t, $X_i(t)$ is the ith individual with dimension $N(N = n)$ inPop(t), $x_{i,l}(t)$ is the lth variable of individual $X_i(t)$, and tmp_l is the lth variable of tmp. Each $X_i(t)$ is evaluated or calculated via strategy one.
Step 1: input N, $N_p \ge 3$, CR $\in [0, 1]$, and let bounds be lower$(x_{i,l}) = 0$ and upper$(x_{i,l}) = 4$, $l = 1, \ldots, N$.
Step 2: calculate and record sp_{j_i}, $\sum_{i=1}^{n} \text{sp}_{j_i}$ and L_{j_{i-1}, j_i} $(j_{i-1}, j_i \in \{1, \ldots, n\})$. //prepare for//utilizing strategy one
Step 3: population initialization. $x_{i,l}(0) = \text{lower}(x_{i,l}) + \text{random}(0, 1) * (\text{upper}(x_{i,l}) - \text{lower}(x_{i,l}))$, 1/2, $l = 1, \ldots, N$ for $i = 1, \ldots, N_p$.
Step 4: set $t = 1$ and select an individual $X_{\text{best}}(0)$ with the minimum objective value from Pop(0) as best.
Step 5: evolution stage (Step 5 to Step 11). Set i=1.
Step 6: set *the trial vector*tmp $= X_i(t-1)$ and $L' = 0$. Randomly select $r1, r2 \in \{1, \ldots, N_p\}$ $(r1 \ne r2 \ne i)$ and randomly select $l \in \{1, \ldots, N\}$.
Step 7: execute **DE's Mutation and Crossover**.
  Step 7.1: let $\text{tmp}_l = \text{tmp}_l + F^*(\text{best}_l - \text{tmp}_l) + F * (x_{r1,l}(t-1) - x_{r2,l}(t-1))$.
    If $\text{tmp}_l < \text{lower}(x_{i,l})$, then let $\text{tmp}_l = 2^*\text{lower}(x_{i,l}) - \text{tmp}_l$.
    If $\text{tmp}_l > \text{upper}(x_{i,l})$, then let $\text{tmp}_l = 2^*\text{upper}(x_{i,l}) - \text{tmp}_l$.
  Step 7.2: set $l = (l \bmod N) + 1$ and $L' = L' + 1$.
  Step 7.3: if $(\text{random}(0, 1) < CR)$ and $(L' < N)$, go to Step 7.1.
Step 8: execute **DE's Selection**.
  If $f(\text{tmp}) \le f(X_i(t-1))$, then set $X_i(t) = \text{tmp}$;
  else set $X_i(t) = X_i(t-1)$.
Step 9: if $f(\text{tmp}) < f(\text{best})$, then set best = tmp.
Step 10: set $i = i + 1$. If $i \le N_p$, then go to Step 6.
Step 11: execute **fast local search** on best.
Step 12: set $t = t + 1$. If $t \le t_$max, then go to Step 5.
Step 13: output the current best with its objective value.

Algorithm 2: Pseudocode of DE_FES.

addressing the considered problem efficiently. Not only does DE_FES adopt DE's parallel searching scheme to find promising regions, but it also utilizes an efficient local search based on three problem-dependent strategies to execute deep exploitation in these promising regions. Because the global and local searches are well balanced, DE_FES is expected to acquire satisfactory solutions in a reasonable time.

5. Simulation Results and Comparisons

5.1. Experimental Setup. To test the performance of DE_FES, numerical tests are carried out by using a series of randomly generated instances. The $n \times m$ combinations are set as {20, 30, 50, 70} × {5, 10, 20}. Both the setup time $s_{j_{i-1}, j_i, l}$ and the processing time $p_{j_i, l}$ are generated from a uniform

Table 1: Comparison results of four algorithms.

Instance ($N \times m$)	IGA				MSA_FMS				DE_FES_V1				DE_FES			
	BIP	AIP	SD	CT	BIP	AIP	SD	CT	BIP	AIP	SD	CT	BIP	AIP	SD	CT
20×5	45.863	45.370	0.439	6575	43.291	39.323	2.619	6005	45.571	43.084	2.128	6900	**45.888**	**45.571**	**0.291**	**45238**
20×10	42.690	41.696	0.665	6316	40.467	36.884	2.720	6005	42.458	39.704	2.665	7018	**42.822**	**42.425**	**0.353**	**45589**
20×20	42.665	42.108	0.536	6402	40.083	35.405	3.005	6005	42.619	40.608	2.031	7841	**42.755**	**42.601**	**0.228**	**55974**
30×5	44.297	43.272	0.837	15075	42.212	39.739	1.519	13507	43.085	36.879	4.037	17537	**44.308**	**43.800**	**0.356**	**189400**
30×10	47.471	46.236	1.016	14930	45.653	42.579	1.782	13507	45.164	40.483	3.403	19197	**47.862**	**47.199**	**0.475**	**220552**
30×20	55.064	54.423	0.583	14999	53.148	50.177	1.692	13508	54.178	49.051	3.273	21623	**55.579**	**55.018**	**0.504**	**251780**
50×5	57.289	56.567	0.542	41091	56.747	55.748	0.623	37513	54.868	50.124	3.512	67412	**57.305**	**56.914**	**0.276**	**793132**
50×10	**51.363**	50.579	0.605	41406	50.444	49.333	0.625	37511	47.598	43.275	3.205	69226	51.357	**50.808**	**0.387**	**930641**
50×20	56.076	54.799	0.959	40504	55.216	53.939	0.877	37511	53.295	48.721	3.139	78760	**56.310**	**55.739**	**0.374**	**1194146**
70×5	**50.054**	48.962	0.648	80750	49.823	49.032	**0.466**	73516	45.302	38.740	3.867	205045	49.960	**49.374**	0.496	**1730785**
70×10	56.837	55.892	0.565	81593	56.729	55.848	0.544	73515	51.306	44.605	4.055	200543	**57.002**	**56.459**	**0.367**	**2143718**
70×20	60.330	59.422	0.552	80767	60.064	59.234	0.465	73517	55.589	50.537	3.470	205202	**60.518**	**59.934**	**0.364**	**2798744**
Average	50.833	49.944	0.662	35867	49.490	47.270	1.411	32635	48.420	43.817	3.232	75525	**50.972**	**50.487**	**0.373**	**866642**

Table 2: Comparison results of four algorithms on 21 instances with different α.

Instance (N, m)	IGA				MSA_FMS				DE_FES_V1				DE_FES			
	BIP	AIP	SD	CT	BIP	AIP	SD	CT	BIP	AIP	SD	CT	BIP	AIP	SD	CT
α								20×10								
0	**49.165**	**48.909**	**0.314**	7146	47.357	44.146	2.328	6004	48.751	46.811	1.133	7661	49.076	48.700	0.405	**45476**
0.2	59.238	58.215	0.976	6116	56.765	52.936	2.179	6003	58.340	57.330	0.922	6915	**59.239**	**58.480**	**0.604**	**40832**
0.4	62.253	59.144	1.544	6255	60.119	57.394	2.148	6006	62.056	60.216	2.909	6812	**62.275**	**61.991**	**0.298**	**44966**
0.6	65.597	64.908	**0.273**	6367	62.235	57.418	4.363	6005	65.156	61.101	3.838	7170	**65.600**	**65.155**	0.336	**50647**
0.8	26.985	25.304	1.098	6255	23.702	19.129	2.779	6004	27.135	23.334	4.309	6953	**27.135**	**26.733**	**0.390**	**49608**
1.0	25.083	25.066	**0.053**	5967	22.584	18.083	3.553	6005	25.259	18.684	5.355	6897	**25.918**	**25.408**	0.435	**51369**
1.5	10.508	10.324	0.395	6106	**10.509**	9.083	1.691	6006	10.509	10.450	0.186	6717	**10.509**	**10.509**	**0.000**	**36227**
								30×10								
0	**66.162**	**64.722**	1.506	15459	63.185	59.259	1.766	13507	63.986	62.055	1.098	18164	66.071	64.405	**0.973**	**200428**
0.2	70.866	70.137	0.550	14947	68.842	66.700	1.775	13508	70.637	68.835	1.045	19179	**70.896**	**70.315**	**0.375**	**215644**
0.4	64.583	62.566	2.042	14849	63.198	59.134	1.679	13507	64.408	58.946	3.890	18629	**65.434**	**64.432**	**0.686**	**215138**
0.6	**48.556**	47.642	1.043	15040	46.795	44.215	1.597	13507	46.247	42.306	2.874	19589	48.555	**48.069**	**0.525**	**225717**
0.8	42.160	40.394	1.049	14800	41.493	38.176	2.593	13506	38.882	31.539	4.942	20657	**43.570**	**43.338**	**0.195**	**232461**
1.0	31.322	29.788	0.777	14569	27.703	24.564	1.523	13510	23.546	15.498	6.419	18893	**31.856**	**31.194**	**0.557**	**225553**
1.5	8.647	8.405	0.144	14847	8.355	6.008	1.540	13507	8.441	4.203	3.552	19269	**8.648**	**8.642**	**0.018**	**228924**
								70×10								
0	75.659	75.132	**0.381**	82677	76.688	75.216	0.723	73516	74.364	72.604	1.124	154623	**76.235**	**75.342**	0.548	**2092456**
0.2	**78.687**	77.439	0.925	87863	78.231	77.190	0.769	73512	75.786	73.427	1.745	176600	78.556	**77.903**	**0.384**	**2130306**
0.4	82.864	82.009	0.541	84500	82.612	81.788	0.754	73516	79.440	75.092	3.023	197006	**83.275**	**82.800**	**0.356**	**2130650**
0.6	**60.425**	58.181	0.836	83435	59.472	57.783	0.858	73517	51.743	37.813	7.991	184690	59.867	**59.158**	**0.600**	**2207943**
0.8	52.211	51.370	0.759	77457	52.533	52.100	**0.238**	73515	46.681	38.042	4.138	202198	**52.762**	**52.261**	0.305	**2159777**
1.0	41.860	41.084	0.431	77498	41.452	40.849	0.380	73515	32.752	22.887	6.615	219949	**42.191**	**41.715**	**0.280**	**2169624**
1.5	**6.150**	6.031	0.084	77722	6.115	6.011	**0.082**	73515	−1.620	−7.631	3.751	268732	6.130	**6.035**	0.093	**2115272**
Average	48.999	47.941	0.749	34280	47.616	45.104	1.682	31009	46.309	41.597	3.374	75586	**49.228**	**48.695**	**0.398**	**803287**

distribution [1, 100]. The release time r_{j_i} is randomly generated from $(0, 150n\alpha)$, in which α is a control parameter. Obviously, the value of r_{j_i} increases with the value of α. If $r_{j_i} \geq \mathrm{St}_{j_{i-1}} + L_{j_{i-1},j_i}$ and $r_{j_i} \geq d_{j_i}$ for $i = 1, \ldots, n$, the optimal solution can be obtained when each job j_i is processed at its release time. This means that a larger value of α may reduce the difficulty of solving the problem. Hence, the values of α are set to 0, 0.2, 0.4, 0.6, 0.8, 1, and 1.5, respectively. The weight value w_{j_i} is randomly generated in (0, 1). Let d_{p,j_i} be the due date of the job j_i on the instance p, C_{p,j_i} the completion time of the job j_i on the instance p, and random $(-C_{p,j_i}, 0)$ a random value in $(-C_{p,j_i}, 0)$. Then, d_{p,j_i} is set as follows:

Step 1: randomly generate a sequence $(j_1, j_2, \ldots, j_n)$ for each instance p.

Step 2: calculate C_{j_i} for $i = 1, \ldots, n$.

Step 3: specify d_{p,j_i} by

$$d_{p,j_i} = C_{p,j_i} + \mathrm{random}\left(-C_{p,j_i}, 0\right). \tag{29}$$

Obviously, the due date d_{p,j_i} is relatively tight, which helps maintain the company's competitiveness. The total number of test instance is $4 \times 3 \times 7 = 84$. These instances can be downloaded from https://pan.baidu.com/s/1mCdtt3MisBGf16W19xP2aA (password: cmka).

DE_FES's three main parameters are set as follows: the scaling factor $F = 0.7$, the crossover parameter CR = 0.1, and the population size popsize = 30. Furthermore, KK in DE_FES's local search is set to 3. In order to make a fair comparison, the maximum generation of a modified simulated annealing algorithm with first move strategy

(MSA_FMS) [37] is set to $15 \times n^2$, and the other algorithms run the same time as MSA_FMS. Each algorithm runs 20 times independently on each instance. All procedures are implemented with Delphi 2010 and the comparisons are executed on a 2.33 GHz CPU with 3 GB memory.

5.2. Comparisons of DE_FES and Two Effective Algorithms. Let $\pi_{\text{ini}}(\alpha)$ be the permutation or schedule in which all jobs are arranged in ascending order of release time at α, $\pi(\alpha)$ the permutation π at α, $\text{TWT}(\pi(\alpha))$ the value of TWT when $\pi = \pi(\alpha)$, $\text{avg\_TWT}(\pi(\alpha))$ the average value of $\text{TWT}(\pi(\alpha))$, $\text{best\_TWT}(\pi(\alpha))$ the best value of $\text{TWT}(\pi(\alpha))$, $\text{worst\_TWT}(\pi(\alpha))$ the worst value of $\text{TWT}(\pi(\alpha))$, $\text{AIP}(\pi(\alpha)) = (\text{TWT}(\pi_{\text{ini}}(\alpha)) - \text{avg\_TWT}(\pi(\alpha)))/\text{TWT}(\pi_{\text{ini}}(\alpha)) \times 100\%$ the relative improvement percentage of $\text{avg\_TWT}(\pi(\alpha))$ to $\text{TWT}(\pi_{\text{ini}}(\alpha))$, $\text{BIP}(\pi(\alpha)) = (\text{TWT}(\pi_{\text{ini}}(\alpha)) - \text{best\_TWT}(\pi(\alpha)))/\text{TWT}(\pi_{\text{ini}}(\alpha)) \times 100\%$ the relative improvement percentage of $\text{best\_TWT}(\pi(\alpha))$ to $\text{TWT}(\pi_{\text{ini}}(\alpha))$, $\text{CT}(\alpha)$ the calculation or evaluation times of $\text{TWT}(\pi_{\text{ini}}(\alpha))$, $\text{SD}(\alpha)$ the standard deviation of $\text{TWT}(\pi(\alpha))$ at α, S_α the set of all values of α, and $|S_\alpha|$ the number of different values in S_α. Then, four following performance measures are defined to evaluate the performances of the compared algorithms. These performance measures are $\text{AIP} = \sum_{\alpha \in S_\alpha} \text{AIP}(\alpha)/|S_\alpha|$, $\text{BIP} = \sum_{\alpha \in S_\alpha} \text{BIP}(\alpha)/|S_\alpha|$, $\text{CT} = \sum_{\alpha \in S_\alpha} \text{CT}(\alpha)/|S_\alpha|$, and $\text{SD} = \sum_{\alpha \in S_\alpha} \text{SD}(\alpha)/|S_\alpha|$.

To demonstrate the effectiveness of DE_FES for the TWT-NFSP with SSTs and RTs, DE_FES is tested against two effective scheduling algorithms, i.e., a modified simulated annealing algorithm with first move strategy (MSA_FMS) [37] and an iterated greedy algorithm (IGA) [38]. MSA_FMS outperforms a well-known simulated annealing algorithm presented by Osman and Potts [37, 39]. Based on our previous tests, MSA_FMS also performs better than a hybrid genetic algorithm [40]. IGA is one of the best algorithms for solving the FSPs with SSTs [38, 41]. In addition, to show the effectiveness of the proposed strategies two and three, DE_FES is also compared with its variant DE_FES_V1, which removes these two strategies from DE_FES.

The test results are given in Tables 1 and 2. Table 1 shows the values of BIP, AIP, SD, and CT of each compared algorithm. Table 2 provides the details of each compared algorithm for addressing the instances 20×10, 30×10, and 70×10 under different α.

It is clear from Tables 1 and 2 that, in most instances, the values of AIP and BIP obtained by DE_FES are larger than those obtained by IGA and MSA_FMS and, in all instances, are larger than those obtained by DE_FES_V1. This confirms the superiority of DE_FES and clarifies the effect of utilizing the proposed strategies in DE. In addition, it can be seen that the CT values of DE_FES are obviously larger than those of IGA and MSA_FMS, and it increases quickly as the problem's scale increases. This means that the proposed speed-up strategies can significantly reduce the CC of evaluating solutions and neighbors. More precisely, with the help of the proposed strategies, DE_FES can search more regions or solutions under the same running time. This greatly increases its probability of obtaining high-quality solutions. Besides, the SD values of DE_FES are relatively small, which indicates the robustness of DE_FES. In summary, DE_FES has powerful search ability to solve the considered problem.

6. Conclusions

As far as we know, this is the first paper on differential evolution (DE) for dealing with the TWT-NFSP with SSTs and RTs. In view of the complexity of the problem, a differential evolution algorithm with fast evaluating strategies (DE_FES) is developed to find satisfactory solutions of the considered problem. First, the LOV rule is adopted to ensure that DE is suitable for solving the flow-shop scheduling problems. Second, the DE-based global search is used to guide the search to enough promising regions distributed in solution space. Third, by investigating the problem's model structure and the neighbourhood properties, three fast evaluating strategies are proposed and then applied to design a fast local search, which is used to execute thorough and fast exploitation from the promising regions obtained via DE-based search. Since the DE-based parallel search and the problem-dependent local search in DE_FES are well balanced, it can effectively solve the TWT-NFSP with SSTs and RTs. Test results manifest the efficiency and robustness of the proposed DE_FES. Future research direction is to find more valuable properties and neighbourhoods for scheduling problems and extend the DE-based algorithm to uncertain scheduling problems.

Data Availability

Data were curated by the authors and are available upon request.

Conflicts of Interest

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

Acknowledgments

This research was partially supported by National Natural Science Foundation of China (Grant nos. 51665025, 61963022, and 60904081).

References

[1] A. Allahverdi, "A survey of scheduling problems with no-wait in process," *European Journal of Operational Research*, vol. 255, no. 3, pp. 665–686, 2016.

[2] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer Press, New York, NY, USA, 5th edition, 2016.

[3] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.

[4] F. Jin, S. Song, and C. Wu, "A simulated annealing algorithm for single machine scheduling problems with family setups,"

Computers & Operations Research, vol. 36, no. 7, pp. 2133–2138, 2009.

[5] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," *The Journal of the Operational Research Society*, vol. 45, no. 4, pp. 472–478, 1994.

[6] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations Research*, vol. 44, no. 3, pp. 510–525, 1996.

[7] W. H. M. Raaymakers and J. A. Hoogeveen, "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing," *European Journal of Operational Research*, vol. 126, no. 1, pp. 131–151, 2000.

[8] R. Ruiz and A. Allahverdi, "Some effective heuristics for no-wait flowshops with setup times to minimize total completion time," *Annals of Operations Research*, vol. 156, no. 1, pp. 143–171, 2007.

[9] S. Wang, J. Na, and Y. Xing, "Adaptive optimal parameter estimation and control of servo mechanisms: theory and experiments," *Institute of Electrical and Electronics Engineers Transactions on Industrial Electronics*, vol. 68, no. 1, p. 598, 2021.

[10] S. Wang and J. Na, "Parameter estimation and adaptive control for servo mechanisms with friction compensation," *Institute of Electrical and Electronics EngineersTransactions on Industrial Informatics*, vol. 16, no. 11, pp. 6816–6825, 2020.

[11] S. Wang, L. Tao, Q. Chen, J. Na, and X. Ren, "USDE-based sliding mode control for servo mechanisms with unknown system dynamics," *Institute of Electrical and Electronics Engineers/ASME Transactions on Mechatronics*, vol. 25, no. 2, pp. 1056–1066, 2020.

[12] Y.-N. Guo, X. Zhang, D.-W. Gong, Z. Zhang, and J.-J. Yang, "Novel interactive preference-based multiobjective evolutionary optimization for bolt supporting networks," *Institute of Electrical and Electronics Engineers Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 750–764, 2020.

[13] Y.-N. Guo, J. Cheng, S. Luo, D. Gong, and Y. Xue, "Robust dynamic multi-objective vehicle routing optimization method," *Institute of Electrical and Electronics Engineers/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1891–1903, 2018.

[14] Y. X. Yang, P. Li, S. Y. Wang, B. Liu, and Y. L. Luo, "Scatter search for distributed assembly flowshop scheduling to minimize total tardiness," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 861–868, Wellington, New Zealand, June 2017.

[15] B. Qian, Z.-C. Li, and R. Hu, "A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem," *Applied Soft Computing*, vol. 61, pp. 921–934, 2017.

[16] W. Z. Duan, Z. Y. Li, Y. X. Yang, B. Liu, and K. Y. Wang, "EDA based probabilistic memetic algorithm for distributed blocking permutation flowshop scheduling with sequence dependent setup time," in *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 992–999, Wellington, New Zealand, June 2017.

[17] H.-Y. Sang, Q.-K. Pan, P.-Y. Duan, and J.-Q. Li, "An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 29, no. 6, pp. 1337–1349, 2018.

[18] Z. C. Li, B. Qian, R. Hu, L. L. Chang, and J. B. Yang, "An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups," *Knowledge-Based Systems*, vol. 173, pp. 83–112, 2019.

[19] P. Wang, H. Sang, Q. Tao et al., "Improved migrating birds optimization algorithm to solve hybrid flowshop scheduling problem with lot-streaming," *Institute of Electrical and Electronics Engineers Access*, vol. 8, pp. 89782–89792, 2020.

[20] 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.

[21] R. Hu, Q. Zhang, B. Qian, L. Chang, and Z. Zhou, "An effective soft-Sensor method based on belief-rule-base and differential evolution for tipping paper permeability measurement," *Complexity*, vol. 2018, pp. 1–14, Article ID 4378701, 2018.

[22] B. Qian, Q.-Q. Wang, R. Hu, Z.-J. Zhou, C.-Q. Yu, and Z.-G. Zhou, "An effective soft computing technology based on belief-rule-base and particle swarm optimization for tipping paper permeability measurement," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 3, pp. 841–850, 2019.

[23] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "Differential evolution algorithm for permutation flowshop sequencing problem with makespan criterion," in *Proceedings of 4th International Symposium on Intelligent Manufacturing Systems*, pp. 442–452, Sakarya, Turkey, 2004.

[24] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, 2006.

[25] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 55, no. 4, pp. 795–816, 2008.

[26] B. Qian, L. Wang, R. Hu, D. X. Huang, and X. Wang, "A DE-based approach to no-wait flow-shop scheduling," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 787–805, 2009.

[27] L. Wang, Q.-K. Pan, P. N. Suganthan, W.-H. Wang, and Y.-M. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems," *Computers & Operations Research*, vol. 37, no. 3, pp. 509–520, 2010.

[28] R. Hu, X. Meng, B. Qian, and K. Li, "A differential evolution approach for NTJ-NFSSP with SDSTs and RDs," *Lecture Notes in Computer Science*, vol. 7390, pp. 288–299, 2012.

[29] B. Qian, P. Z. Du, R. Hu, and G. L. Che, "A differential evolution algorithm with two speed-up methods for NFSSP with SDSTs and RDs," in *The 10th World Congress on Intelligent Control and Automation*, pp. 490–495, London, UK, July 2012.

[30] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *Institute of Electrical and Electronics Engineers Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 209–225, 2014.

[31] V. Santucci, M. Baioletti, and A. Milani, "Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion," *Institute of Electrical and Electronics Engineers Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 682–694, 2016.

[32] X. Wu and A. Che, "A memetic differential evolution algorithm for energy-efficient parallel machine scheduling," *Omega*, vol. 82, pp. 155–165, 2019.

[33] Y. C. Ho, Q. C. Zhao, and D. Pepyne, "The no free lunch theorem: complexity, and computer security," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 48, no. 5, pp. 783–793, 2003.

[34] T. Schiavinotto and T. Stützle, "A review of metrics on permutations for search landscape analysis," *Computers & Operations Research*, vol. 34, no. 10, pp. 3143–3153, 2007.

[35] B. Qian, L. Wang, R. Hu et al., "A hybrid differential evolution for permutation flow-shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 757–777, 2008.

[36] B. Qian, L. Wang, D.-x. Huang, W.-l. Wang, and X. Wang, "An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers," *Computers & Operations Research*, vol. 36, no. 1, pp. 209–233, 2009.

[37] H. Ishibuchi, S. Misaki, and H. Tanaka, "Modified simulated annealing algorithms for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 81, no. 2, pp. 388–398, 1995.

[38] R. Ruiz and T. Stützle, "An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1143–1159, 2008.

[39] I. Osman and C. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[40] L. Wang and D. Z. Zheng, "An effective hybrid heuristic for flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 21, no. 1, pp. 38–44, 2003.

[41] X. Li, Z. Yang, R. Ruiz, T. Chen, and S. Sui, "An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects," *Information Sciences*, vol. 453, pp. 408–425, 2018.