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

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

Copyright © 2020 Rong Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 develop an effective algorithm to address the total weighted tardiness objective (TWT) for the NFSP with SSTs and RTs (i.e., \( T_{\text{m}/\text{no-wait}}^{\text{TWT}} \)).

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 unrelated parallel machine scheduling problem. In their tests, MDE outperforms two famous multimutational algorithms. However, according to the existing literatures, there are no studies on algorithms. In their 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 neighborhood 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.

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

$$C_{j_i} = S_{j_i} + sp_{j_i}, \quad i = 1, \ldots, n.$$  (4)

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, $p_{j_i}$ is the processing time of job $j_i$, on machine $l$, $s_{j_i,l}$ is the total processing time of job $j_i$ on $m$ machines, $ML_{j_i,l}$ is the minimum delay time between $j_i$ and $j_i$ on the machine $l$, $L_{j_i-1,l}$ 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 $p_{j_i,l} = 0$ for $l = [1, \ldots, m]$. Then, $ML_{j_i,l}$ can be calculated as follows:

$$ML_{j_i,l} = \begin{cases} \max\{s_{j_i-1,j_i}^l + p_{j_i-1} - p_{j_i}, s_{j_i-1,j_i}^l\}, & l = 2, \\ \max\{ML_{j_i-1,l} - p_{j_i-1} + s_{j_i-1,j_i}^l\} - p_{j_i,l}, & l = 3, \ldots, m. \end{cases}$$  (1)

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

$$L_{j_i-1,l} = ML_{j_i,m} + sp_{j_i-1} - sp_{j_i}.$$  (2)

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

$$St_{j_i} = \begin{cases} \max\{ML_{j_i,m} - sp_{j_i}, r_{j_i}\}, & i = 1, \\ St_{j_i-1} + \max\{L_{j_i-1,l}, r_{j_i} - St_{j_i-1}\} = \max\{St_{j_i-1} + L_{j_i-1,l}, r_{j_i}\}, & i = 2, \ldots, n. \end{cases}$$  (3)

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 $\Pi$ is the set of all permutations. Then, TWT($\pi$) can be calculated as follows:
$T_j = \max\{C_j - d_j, 0\}$

$= \max\{S_j + s_j - d_j, 0\}, \quad i = 1, \ldots, n,$

$\text{TWT}(\pi) = \sum_{i=1}^{n} w_i T_j = \sum_{i=1}^{n} w_i \max\{C_j - d_j, 0\}$

$= \sum_{i=1}^{n} w_i \max\{S_j + s_j - d_j, 0\}.$

(6)

The TWT-NFSP with SSTs and RTs is to find an optimal permutation $\pi^*$ in $\Pi$ such that

$\pi^* = \arg\{\text{TWT}(\pi)\} \rightarrow \min, \quad \forall \pi \in \Pi.$

(7)

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

Theorem 1. $F_m/\text{no} - \text{wait}, ST_{sd}/\sum w_jT_j$ is NP-hard.

Proof. Obviously, $F_m/\text{no} - \text{wait}, ST_{sd}/\sum C_j$ reduces to $F_m/\text{no} - \text{wait}, ST_{sd}/\sum C_j$ by setting $r_j = 0$ for all $j$, and $F_m/\text{no} - \text{wait}, ST_{sd}/\sum C_j$ is NP-hard [3, 8]. Problem 1 holds.

The reductions between objective functions are given in Figure 2. It can be seen from Figure 2 that $\sum w_jT_j$ 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 the 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...
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\rightarrow j'} is only decided by j_{i-1} and j_i. By utilizing this property, the computing complexity (CC) of TWT(\pi) can be reduced. That is, L_{j\rightarrow j'}, sp_{j_i}, and \sum_{i=1}^{n} 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 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 \pi 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 \pi can be given as

\[ N_{\text{interchange}}(\pi) = \{ \pi'_{\text{interchange}} = \text{interchange}(\pi, u, v) | 1 \leq u < v \leq n \}, \]

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

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

| Complexity |
|------------|

Denote \( \text{FindBest} N_{\text{interchange}}(\pi) \) is the method of obtaining the best neighbor with TWT criterion in \( N_{\text{interchange}}(\pi) \) (i.e., the \( N_{\text{interchange}}(\pi) \)-based neighbourhood scanning method). When \( u = 1, \ldots, n - 1 \) and \( i = 1, \ldots, n - 1 \), it has \( \Delta = T_{ji} - T_{ji} = w_{ji}T_j + w_{ji}T_{ji} \), and \( \Delta' = w_{ji}T_j + w_{ji}T_{ji} \). Then, according to (5) and (6), it has

\[
\text{TWT}(\pi_{\text{interchange}}) = \sum_{i=1}^{n-1} w_{ji}T_j + \sum_{i=1}^{n} w_{ji}T_{ji}
\]

(10)

It is clear from (10) that in Find Best \( N_{\text{interchange}}(\pi) \), \( St_{ji} \) and \( \sum_{i=1}^{n-1} w_{ji}T_j \) can be directly replaced by \( St_{ji} \) and \( \sum_{i=1}^{n-1} w_{ji}T_{ji} \), respectively, and \( St_{ji} \) can be calculated from \( St_{ji} \). When scanning each neighbor in \( N_{\text{interchange}}(\pi) \), only \( \sum_{i=1}^{n-1} w_{ji}T_{ji} \) needs to be calculated. So, by using strategies one and two, the CC of \( \text{FindBest} N_{\text{interchange}}(\pi) \) can be decreased to some extent.

3.4. Strategy Three: Fast Nonimproving Neighbor Identification Strategy for \( N_{\text{interchange}}(\pi) \). According to the \( N_{\text{interchange}}(\pi) \)-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 \( N_{\text{interchange}}(\pi) \) can be identified with lower CC, which means the CC of Find Best \( N_{\text{interchange}}(\pi) \) can be further reduced.

Denote \( \Delta St_{ji} (y = 1, 2, \ldots, n - 1) \) and \( \Delta St_{ji} \geq 0 \) is a start processing time delay of \( j_y \) on the first machine, \( St_{ji} \) is the start processing time of \( j_y \) on the first machine when \( \Delta St_{ji} \) is added to \( St_{ji} \) (i.e., \( St_{ji}' = St_{ji} + \Delta St_{ji} \), \( St_{ji}' \geq 0 \)) is the start processing time of job \( j_y \) on the first machine when \( St_{ji} \) is set to \( St_{ji}' \), and \( T_{ji} \) is the tardiness of job \( j_y \) on the first machine when \( St_{ji} \) is set to \( St_{ji}' \).

**Lemma 1.** For a fixed job permutation \( \pi = (j_1, j_2, \ldots, j_n) \) of an NFSP with SSTs and RTs, if \( St_{ji} \) is set to \( St_{ji}' \) (\( St_{ji}' \geq St_{ji} \)), then it has \( St_{ji}' \geq St_{ji} \) for all \( i = 1, \ldots, n - y \).

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

**Figure 3:** A small example of \( \pi_{\text{interchange}} \) when \( n = 10, u = 4 \), and \( v = 7 \).
Proof. Let $\Delta_{y;il} = \text{ST}_{jy;il} - \text{ST}_{jy;il}$ ($ll = 1, \ldots, n - y$) denote the start processing time delay of $j_{y;il}$. Lemma 1 can be proved by using mathematical induction.

Firstly, when $ll = 1$, with (3), it has
\[
\text{ST}'_{jy;il} = \text{ST}_{jy} + \max \left\{ \text{LT}_{jy,jy;il} \right\}
\]
\[
= \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il} \text{r}_{jy;il} \
\]
\[
= \max \left\{ \text{ST}_{jy} + L_{jy,jy;il} \text{r}_{jy;il}, \text{ST}_{jy} + \Delta \text{ST}_{jy} \right\}.
\]

(11)

Obviously, when $\Delta \text{ST}_{jy} = 0$, it has $\text{ST}'_{jy;il} = \text{ST}_{jy;il}$.

When $\Delta \text{ST}_{jy} > 0$, based on the value of $\text{r}_{jy;il}$, three cases are discussed as follows:

Case 1. $\text{r}_{jy;il} \leq \text{ST}_{jy} + L_{jy,jy;il}$.

Under this case, with (11) and (3), it has
\[
\text{ST}'_{jy} = \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il} \text{r}_{jy;il} > \text{ST}_{jy;il}.
\]

(12)

Case 2. $\text{ST}_{jy} + L_{jy,jy;il} < \text{r}_{jy;il} < \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il}$.

Under this case, with (11) and (3), it has
\[
\text{ST}'_{jy;il} = \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il} \text{r}_{jy;il} > \text{ST}_{jy;il}.
\]

(13)

Case 3. $\text{r}_{jy;il} \geq \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il}$.

Under this case, with (11) and (3), it has
\[
\text{ST}'_{jy;il} = \text{ST}_{jy} + \Delta \text{ST}_{jy} + L_{jy,jy;il} \text{r}_{jy;il} = \text{ST}_{jy;il}.
\]

(14)

and with (12)–(14), it has $\text{ST}'_{jy;il} \geq \text{ST}_{jy;il}$.

Secondly, suppose $\text{ST}'_{jy;il} \geq \text{ST}_{jy;il}$ holds when $1 \leq ll < k$. Thus, $\Delta_{y+k} = \text{ST}'_{jy+k} - \text{ST}_{jy+k} \geq 0$. With (3) and (11), it has
\[
\text{ST}'_{jy+k;il} = \max \left\{ \text{ST}_{jy+k} + \Delta \text{ST}_{jy+k} + L_{jy+k,jy+k;il} \text{r}_{jy+k;il}, \text{ST}_{jy+k;il} \right\}.
\]

(15)

Similar to the analysis when $ll = 1$, it has $\text{ST}'_{jy+k;il} \geq \text{ST}_{jy+k;il}$.

Thus, based on mathematical induction, $\text{ST}'_{jy;il} \geq \text{ST}_{jy;il}$ holds when $1 \leq ll \leq n - y$.

Lemma 2. For a fixed job permutation $\pi = [j_1, j_2, \ldots, j_n]$ of an NFS with SSTs and RTSs, if $\text{ST}_{jy}$ is set to $\text{ST}_{jy}$ ($\text{ST}'_{jy} \geq \text{ST}_{jy}$), then it has $T'_{jy;il} \geq T_{jy;il}$ for $ll = 1, \ldots, n - y$.

Proof. According to Lemma 1, when $\text{ST}'_{jy} \geq \text{ST}_{jy}$, it has $\text{ST}'_{jy;il} \geq \text{ST}_{jy;il}$ for $ll = 1, \ldots, n - y$. Then, with (5), it has
\[
T'_{jy;il} = \max \left\{ \text{ST}'_{jy;il} + \text{SP}_{jy;il} - d_{jy;il}, 0 \right\}.
\]

(16)

\[
T_{jy;il} = \max \left\{ \text{ST}_{jy;il} + \text{SP}_{jy;il} - d_{jy;il}, 0 \right\}.
\]

(17)

Obviously, if $\text{ST}'_{jy} = \text{ST}_{jy}$, it has $T'_{jy;il} = T_{jy;il}$ for $ll = 1, \ldots, n - y$.

If $\text{ST}'_{jy} > \text{ST}_{jy}$, according to the value of $d_{jy;il}$, three cases are discussed as follows:

Case 1. $d_{jy;il} \leq \text{ST}_{jy;il} + \text{SP}_{jy;il}$.

Under this case, with (16) and (17), it has
\[
T'_{jy;il} = \text{ST}_{jy;il} + \text{SP}_{jy;il} - d_{jy;il} \geq \text{ST}_{jy;il} + \text{SP}_{jy;il} - d_{jy;il} = T_{jy;il}.
\]

(18)

Case 2. $\text{ST}_{jy;il} + \text{SP}_{jy;il} < d_{jy;il} < \text{ST}'_{jy;il} + \text{SP}_{jy;il}$.

Under this case, with (16) and (17), it has
\[
T'_{jy;il} = \text{ST}_{jy;il} + \text{SP}_{jy;il} - d_{jy;il} \geq 0
\]

(19)

Case 3. $d_{jy;il} \geq \text{ST}'_{jy;il} + \text{SP}_{jy;il}$.

Under this case, with (16) and (17), it has
\[
T'_{jy;il} = 0 = \max \left\{ \text{ST}_{jy;il} + \text{SP}_{jy;il} - d_{jy;il}, 0 \right\} = T_{jy;il}.
\]

(20)

Based on the above analysis, Lemma 2 holds.

Theorem 2. If $\nu < n$ and $\sum_{i=1}^{n} w_i \text{T}'_{ji} \geq \sum_{i=1}^{n} w_i \text{T}_{ji}$ and $\text{ST}_{jy;il} \geq \text{ST}_{jy;il}$, then $\text{TWT}(\pi^{\nu;\nu}) \geq \text{TWT}(\pi)$.

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

Case 1. $u > 1$.

From (6) and (10), it has
\[
\text{TWT}(\pi) = \sum_{i=1}^{u-1} w_i \text{T}_{ji} + \sum_{i=1}^{\nu} w_i \text{T}_{ji} + \sum_{i=1}^{n} w_i \text{T}_{ji},
\]

(21)

\[
\text{TWT}(\pi^{\nu;\nu}) = \sum_{i=1}^{u-1} w_i \text{T}_{ji} + \sum_{i=1}^{\nu} w_i \text{T}_{ji} + \sum_{i=1}^{n} w_i \text{T}_{ji}.
\]

(22)

Since $j'_i = j_i$ for $i = 1, \ldots, u - 1$, it has
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 4th individual $X_i=\{x_{i1},x_{i2},\ldots,x_{in}\}$ of DE to the job permutation vector $\pi_i=\{j_{i1},j_{i2},\ldots,j_{in}\}$. Based on LOV rule, $X_i=\{x_{i1},x_{i2},\ldots,x_{in}\}$ are firstly ranked by descending order to obtain the temporary sequence $\phi_i=\{\phi_{i1},\phi_{i2},\ldots,\phi_{in}\}$. Then, $\pi_i$ can be obtained by using the following formula:

$$j_{i,\phi_{i,k}} = k.$$  

(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 ($\pi, u, v$) is the operator of inserting $j_u$ in the $v$th dimension of $\pi$. 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 $\text{S123}_{\text{Find Best}} N_{\text{interchange}}(\pi)$, 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 $\text{S123}_{\text{Find Best}} N_{\text{interchange}}(\pi)$, 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 $N_{\text{interchange}}(\pi)$ (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 $N_{\text{interchange}}(\pi)$ can be evaluated in a very short time. Obviously, searching the whole neighborhood with low CC is an efficient and ideal search mode. Therefore, $\text{S123}_{\text{Find Best}} N_{\text{interchange}}(\pi)$ adopts the “comprehensive neighbourhood search scheme” similar to that in [29], instead of the “first-improvement-move neighbour- 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 $\text{S123}_{\text{Find Best}} N_{\text{interchange}}(\pi)$ 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 \( \pi_{i,0} \) via the LOV rule.
Step 2: perturbation phase.
   Set \( \pi_{i,t} = \pi_{i,0} \).
   For \( t = 1 \) to \( KK \)
     Randomly select \( u \) and \( v \), where \( |u - v| > n/3 \);
     \( \pi_t = \text{insert} (\pi_{i,t}, u, v) \);
     \( \pi_{i,t} = \pi_t \).
   End.
Step 3: exploitation phase.
   Set \( l = 0 \);
   Do
     \( \pi_{t-1} = \text{S123_Fast Best} \).
     If \( f(\pi_{t-1}) < f(\pi_t) \), then
       \( \pi_t = \pi_{t-1} \);
     else
       \( l++ \);
   end;
   While \( l < 1 \).
Step 4: if \( f(\pi_t) \leq f(\pi_{t-1}) \), then \( \pi_{t,0} = \pi_t \).
Step 5: transform \( \pi_{t,0} \) 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 \((0, 1)\), \( t_{\text{max}} \) is the maximum generation, \( t \) is a generation, \( \text{Pop}(t) \) is the population with size \( N_p \) at \( t \), \( X_i(t) \) is the \( i \)-th individual with dimension \( N(N = n) \) in \( \text{Pop}(t) \), \( x_{ij}(t) \) is the \( j \)-th variable of individual \( X_i(t) \), and \( \text{tmp} \) is the \( j \)-th variable of \( \text{tmp} \). Each \( X_i(t) \) is evaluated or calculated via strategy one.

Step 1: input \( N, N_p \geq 3, \text{CR} \in [0, 1] \), and let bounds be lower \( (x_{ij}) = 0 \) and upper \( (x_{ij}) = 4 \), \( l = 1, \ldots, N \).
Step 2: calculate and record \( s_{i, l} \), \( L_{i, l}, (j, i, l) \in [1, \ldots, n] \). //prepare for utilizing strategy one
Step 3: population initialization. \( x_{ij}(0) = \text{lower}(x_{ij}) + \text{random}(0, 1) \times (\text{upper}(x_{ij}) - \text{lower}(x_{ij})), 1/2, l = 1, \ldots, N \) for \( i = 1, \ldots, N_p \).
Step 4: set \( t = 1 \) and select an individual \( X_{\text{out}}(0) \) with the minimum objective value from \( \text{Pop}(0) \) as best.
Step 5: evolution stage (Step 5 to Step 11). Set \( i = 1 \).
Step 6: set the trial vector \( \text{tmp} = X_i(t - 1) \) and \( L' = 0 \). Randomly select \( r_1, r_2 \in [1, \ldots, N_p] \) \((r_1 \neq r_2 \neq i)\) and randomly select \( l \in [1, \ldots, N] \).
Step 7: execute DE’s Mutation and Crossover.
   Step 7.1: let \( \text{tmp}_l = \text{tmp}_l + F^*(\text{best} - \text{tmp}_l) + F \times (x_{r_1, l}(t - 1) - x_{r_2, l}(t - 1)) \).
   If \( \text{tmp}_l \text{ lower}(x_{ij}) \), then let \( \text{tmp}_l = 2^* \text{lower}(x_{ij}) - \text{tmp}_l \).
   If \( \text{tmp}_l \text{ upper}(x_{ij}) \), then let \( \text{tmp}_l = 2^* \text{upper}(x_{ij}) - \text{tmp}_l \).
   Step 7.2: set \( l = (l \mod N) + 1 \) and \( L' = L' + 1 \).
   Step 7.3: if \( \text{random}(0, 1) < \text{CR} \) and \( (L' < N) \), go to Step 7.1.
Step 8: execute DE’s Selection.
   If \( f(\text{tmp}) \leq f(X_i(t - 1)) \), 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 \( \text{best} = \text{tmp} \).
Step 10: set \( i = i + 1 \). If \( i \leq N_p \), then go to Step 6.
Step 11: execute fast local search on best.
Step 12: set \( t = t + 1 \). If \( t \leq t_{\text{max}} \), 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 exploration 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 \([20, 30, 50, 70] \times [5, 10, 20] \). Both the setup time \( s_{j,h,j} \) and the processing time \( p_{j,i} \) are generated from a uniform
### Table 1: Comparison results of four algorithms.

| Instance (N x m) | IGA | AIP | CT | BIP | AIP | CT | BIP | AIP | CT | BIP | AIP | CT | BIP | AIP | CT |
|------------------|-----|-----|----|-----|-----|----|-----|-----|----|-----|-----|----|-----|-----|----|
| 20 × 5           | 45.863 | 45.370 | 0.439 | 6575 | 46.291 | 39.323 | 2.619 | 6005 | 45.571 | 43.084 | 2.128 | 6900 | 45.888 | 45.571 | 0.291 |
| 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 |
| 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 |
| 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 |
| 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 |
| 50 × 5           | 55.064 | 54.423 | 0.583 | 14999 | 53.148 | 50.177 | 1.692 | 13508 | 54.198 | 49.051 | 3.273 | 21623 | 55.579 | 55.018 | 0.504 |
| 50 × 10          | 57.289 | 56.367 | 0.942 | 41091 | 56.747 | 54.868 | 1.024 | 37513 | 56.887 | 50.124 | 3.212 | 67412 | 57.305 | 56.914 | 0.276 |
| 100 × 5          | 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 |
| 100 × 10         | 50.076 | 49.799 | 0.959 | 40504 | 55.216 | 53.939 | 0.877 | 37513 | 53.295 | 48.721 | 3.139 | 78760 | 56.310 | 55.739 | 0.374 |

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

| Instance (N, m) | IGA | AIP | CT | BIP | AIP | CT | BIP | AIP | CT | BIP | AIP | CT | BIP | AIP | CT |
|------------------|-----|-----|----|-----|-----|----|-----|-----|----|-----|-----|----|-----|-----|----|
| α = 0.1          | 49.165 | 48.999 | 0.314 | 7146 | 47.357 | 44.146 | 2.328 | 6004 | 48.751 | 46.811 | 1.133 | 7661 | 49.076 | 47.800 | 0.405 |
| α = 0.2          | 59.238 | 58.215 | 0.976 | 6116 | 56.765 | 53.296 | 2.176 | 6003 | 58.340 | 57.330 | 0.922 | 6915 | 59.239 | 58.480 | 0.604 |
| α = 0.4          | 62.253 | 59.144 | 1.544 | 6255 | 60.119 | 57.394 | 2.184 | 6006 | 62.056 | 60.216 | 2.909 | 6812 | 62.275 | 61.991 | 0.298 |
| α = 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 |
| α = 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 |
| α = 1.0          | 25.083 | 25.066 | 0.553 | 5967 | 22.584 | 18.083 | 3.553 | 6005 | 25.299 | 26.848 | 5.355 | 6812 | 25.918 | 25.408 | 0.435 |
| α = 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 |

### Step 3: specify $d_{p,j}$ by

$$d_{p,j} = C_{p,j} + \text{random}(-C_{p,j}, 0).$$  \hspace{1cm} (29)

### Step 6: obviously, the due date $d_{p,j}$ 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/1Mcdt3MisBGF16W9xP2aA (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 $\pi_{ini}(a)$ be the permutation or schedule in which all jobs are arranged in ascending order of release time at $a$, $\pi(a)$ the permutation $\pi$ at $a$, $TWT(\pi(a))$ the value of TWT when $\pi = \pi(a)$, $\text{avg}_T\text{WTT}(\pi(a))$ the average value of $TWT(\pi(a))$, $\text{best}_T\text{WTT}(\pi(a))$ the best value of $TWT(\pi(a))$, $\text{worst}_T\text{WTT}(\pi(a))$ the worst value of $TWT(\pi(a))$, $\text{AIP}(\pi(a)) = (TWT(\pi_{ini}(a)) - \text{avg}_T\text{WTT}(\pi(a))) / TWT(\pi_{ini}(a)) \times 100\%$ the relative improvement percentage of $\text{avg}_T\text{WTT}(\pi(a))$ to $TWT(\pi_{ini}(a))$, $\text{BIP}(\pi(a)) = (TWT(\pi_{ini}(a)) - \text{best}_T\text{WTT}(\pi(a))) / TWT(\pi_{ini}(a)) \times 100\%$ the relative improvement percentage of $\text{best}_T\text{WTT}(\pi(a))$ to $TWT(\pi_{ini}(a))$, $\text{CT}(\pi(a))$ the calculation or evaluation times of $TWT(\pi_{ini}(a))$, $\text{SD}(\pi(a))$ the standard deviation of $TWT(\pi(a))$ at $a$, $S_a$ the set of all values of $a$, and $|S_a|$ the number of different values in $S_a$. Then, four following performance measures are defined to evaluate the performances of the compared algorithms. These performance measures are $\text{AIP} = \sum_{\alpha \in S_a} \text{AIP}(\pi(a))/|S_a|$, $\text{BIP} = \sum_{\alpha \in S_a} \text{BIP}(\pi(a))/|S_a|$, $\text{CT} = \sum_{\alpha \in S_a} \text{CT}(\pi(a))/|S_a|$, and $\text{SD} = \sum_{\alpha \in S_a} \text{SD}(\pi(a))/|S_a|$.

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 $20 \times 10, 30 \times 10,$ and $70 \times 10$ under different $a$.

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 Engineers Transactions 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.