Hindawi
Complexity
Volume 2018, Article ID 1919438, 19 pages
https://doi.org/10.1155/2018/1919438

*Research Article*

# Integrated Scheduling of Reheating Furnace and Hot Rolling Based on Improved Multiobjective Differential Evolution

**Kun Li**[1] **and Huixin Tian**[2]

[1]*School of Management, Tianjin Polytechnic University, Tianjin, China*
[2]*School of Electrical Engineering & Automation, Tianjin Polytechnic University, Tianjin, China*

Correspondence should be addressed to Huixin Tian; icedewl@163.com

Received 2 May 2018; Accepted 29 August 2018; Published 12 November 2018

Academic Editor: Yimin Zhou



In the hot rolling mill, a cold slab is reheated in reheating furnace and then rolled into thin coils through hot rolling process. Traditional research on hot rolling mill generally takes the two processes as two separate single-objective scheduling problems, though they are closely connected and their optimization objectives are conflicting with each other. In this paper, the integrated scheduling of the reheating furnace and hot rolling is investigated in the view of multiobjective optimization. A mixed-integer programming model with two objectives is formulated for this problem, and a multiobjective differential evolution (MODE) algorithm is developed to solve this model. To make it easy for algorithm design, a hot rolling schedule (i.e., a sequence of slabs) is used to represent a solution and for a given solution, a heuristic with consideration of energy consumption minimization is presented to obtain the schedule of reheating furnace. To achieve the balance of exploration and exploitation, a novel mutation operator with adaptive leader selection, a crossover operator with feasibility consideration, and a guided multiobjective local search are designed. Computational results based on simulated data illustrate that the integrated scheduling is superior to the traditional two-phase scheduling method used in practice and literature. Further results illustrate that the proposed MODE algorithm is effective and superior to some other multiobjective evolutionary algorithms.

## 1. Introduction

Iron and steel industry is one of the pillar industries for the national economy, and the typical production process in an iron and steel enterprise can be divided into three parts (Figure 1): primary steel making, hot rolling (or rough making), and finishing making. In the primary making section, raw materials such as iron ore and coals are firstly transformed into hot metal through blast furnace, subsequently refined into molten steel in melt shop, and finally casted into solid slabs through continuous casting. The slabs are then sent to hot rolling mill in which they are firstly reheated and then rolled into thin hot rolled strips (or coils). Some of the hot rolled coils can be directly sold as products, but most of them will be sent to the finishing making section for further complex processing. The first procedure in the finishing making section is cold rolling, which makes the coils even thinner at normal temperature. Subsequently, some cold rolled coils are sold as products after continuous annealing or further processed in the color coating line, and the others will be consecutively processed through the continuous galvanizing line and the color coating line. According to Figure 1, it can be found that the hot rolling mill, as the rough making process, connects the logistics between the primary steel making and the finishing making. Therefore, the hot rolling scheduling is one of the key important problems for an iron and steel enterprise.

The detailed production process in the hot rolling mill is illustrated in Figure 2. To guarantee the continuous production of hot rolling, there are generally 2–4 reheating furnaces supplying hot slabs for one hot rolling line. Different from traditional batch processing equipment for which jobs enter and leave in batch style, the continuous walking beam reheating furnace can simultaneously heat many slabs; however, the slabs enter and leave it continuously, instead of the batch style. That is, when the furnace is full, one slab can enter

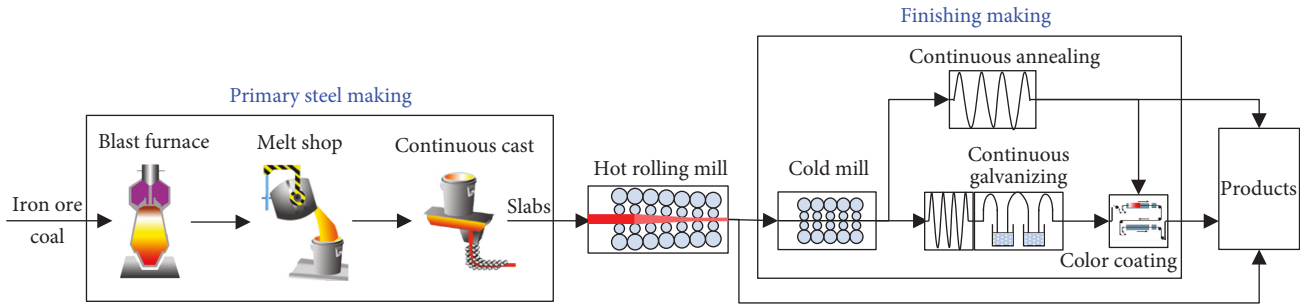


Figure 1: Production process in a hot rolling mill.

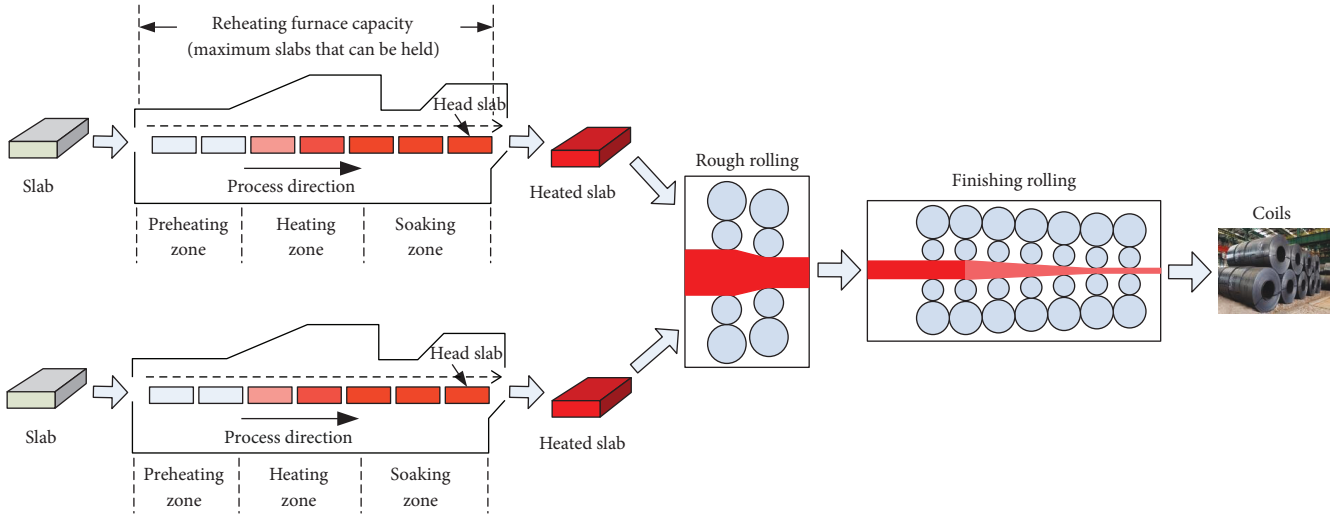


Figure 2: Production process in a hot rolling mill.

the furnace after the head slab leaves the furnace. Before leaving the furnace, each slab must reach its required heating temperature (i.e., it has a minimum heating time in the furnace). On the contrary, a slab cannot leave the furnace even if it has reached its maximum heating time, if there are other slabs before it or the hot rolling line is processing another slab. When such scenario occurs, this slab will be overheated, which results in both energy waste and harm to rolling quality. So the scheduling task of reheating furnace is to allocate slabs to furnaces and determine the heating sequence in each furnace so as to minimize the energy consumption. After pulled out from the reheating furnace, the heated slab further enters the hot rolling process that consists of rough rolling and finishing rolling procedures. The rough rolling machine first transforms the hot slab into a thinner but longer slab (during rolling the width of the slab has little change), and then the finishing rolling machine with 5–7 groups of rollers will consecutively roll the slab into a much thinner strip. Finally, the strip will become a coil at the coiler after cooling. During rolling, if the width of the next coil is smaller than that of its previous coil, then a scar will be left on the next coil. So it is required that the width of coils (slabs) in the hot rolling scheduling should be in a nonascending order. In addition, to guarantee good rolling quality, it is also preferred that the changes in dimension and steel grade between any two slabs should be as less as possible. The reason is that big change in dimension or steel grade will cause significant disturbance to rollers, which in turn deteriorate the rolling stability. That is, the scheduling task of hot rolling is to determine a rolling sequence of slabs so as to minimize the total changeovers of all adjacent slabs.

Based on the above description of production process, it can be found that the scheduling objectives between the reheating furnace and the hot rolling are conflicting with each other, especially in practical production. For example, slabs in a typical iron and steel enterprise generally have many types of dimension (width and thickness), minimum reheating time, and steel grades. The reduction in energy consumption in reheating furnaces may cause significant changeovers of adjacent slabs. On the contrary, good changeovers between adjacent slabs often result in significant increase in energy consumption. Therefore, the integrated scheduling of reheating furnace and hot rolling is a complex multiobjective combinatorial optimization problem.

The rest of the paper is organized as follows. In Section 2, related research results on reheating furnace and hot rolling are reviewed and our research motivation is presented. Section 4 constructs the biobjective mixed-integer programming model for this problem. The multiobjective differential evolution (MODE) algorithm developed for this problem is proposed in Section 5. Section 6 is devoted to analyzing

computational results based on simulated instances. Finally, the paper is summarized in Section 6.

## 2. Literature Review and Motivation

As one of the most important production process related to product quality and energy consumption in iron and steel enterprise, scheduling problems of reheating furnace and hot rolling have attracted considerable attentions in the literature.

For the reheating furnace scheduling, Mui et al. [1] developed a scheduling strategy for the reheating furnace without consideration of the following hot rolling requirements. Broughton et al. [2] investigated the integrated scheduling and control of continuous walking beam reheating furnace and presented a paradigm based on modified genetic algorithm for a practical iron and steel enterprise. However, the mathematical model for the problem was not provided. Mohanty [3] developed an agent-based heuristic to determine the slab sequence in the reheating furnace and hot rolling; however, the overheating and sequence-dependent cost were not considered. Wang and Tang [4] formulated the scheduling of a single continuous walking beam reheating furnace as a mixed-integer linear programming model and presented an improved particle swarm optimization (PSO) algorithm to obtain a near-optimal solution. The case of multiple parallel continuous walking beam reheating furnace was considered in [5], in which the scheduling task was to assign slabs to each reheating furnace, sequence slab sequence for each reheating furnace and determine the feed-in and residence time of each slab so as to reduce unnecessary energy consumption. To solve this problem, a scatter search algorithm was developed. It should be noted that the scheduling problem in [5] was based on a given hot rolling schedule (that is, the drop-out time of each slab has been given by the hot rolling schedule).

With respect to the hot rolling scheduling, many research results can be found in the literature. Lopez et al. [6] and Balas and Martin [7] formulated the hot rolling scheduling as a prize-collecting traveling salesman problem whose task was to select a subset of slabs from the candidate slab yard and determine their rolling sequence to minimize the production cost. Such a modeling method can only obtain one single turn (rollers should be replaced by new ones after rolling a consecutive number of slabs and these consecutive slabs between two roller changeovers are called a turn). To obtain multiple rolling turns at the same time, the vehicle routing problem (VRP) model, as well as its variants, was often adopted. Cowling [8] developed a decision support system for hot rolling scheduling based on the prize-collecting VRP (PCVRP) model. Yadollahpour et al. [9] incorporated more realistic constraints and hot charge rolling into the PCVRP model and presented a guided local search algorithm. Chen et al. [10] formulated the hot rolling scheduling as a VRP model and proposed a hybrid PSO algorithm to simultaneously obtain several rolling turns. Zhao et al. [11] proposed a two-stage scheduling method for the hot rolling, in which the first stage established several turns based on VRP model with time windows and the second stage determined the sequence of these turns to achieve higher hot charge ratios. Different from the two stage method of [11], Wang and Tang [12] presented a VRP model that could obtain multiple turns and at the same time determine their sequence of these turns. Chakraborti et al. [13] presented a biobjective genetic algorithm for the hot rolling scheduling to minimize the standard deviation of lower yield strength (or ultimate tensile strength) and standard deviation of width of all adjacent slabs in a turn. Jia et al. [14] formulated the hot rolling scheduling as a multiobjective PCVRP model and proposed a Pareto max-min ant system algorithm. Tan et al. [15] took into account the time-of-use electricity pricing into the hot rolling scheduling and adopted NSGA-II algorithm [16] to minimize two objectives: total jump penalties of adjacent slabs (i.e., optimize product quality) and total electricity cost (i.e., optimize energy consumption).

From the above literature review, it can be found that the scheduling problems in hot rolling mill have always been an active research topic due to the complexity of reheating and hot rolling. However, current researches traditionally took the reheating furnace scheduling and the hot rolling scheduling as two separate optimization problems and often ignored the close connection between reheating and hot rolling. Moreover, most researches centered on the single-objective optimization of hot rolling, though the objectives have conflictions with each other. Tang and Wang [17] once attempted to achieve the simultaneous optimization of the reheating furnace scheduling and the hot rolling scheduling and proposed a two-phase scheduling method. This method tried to obtain a good hot rolling schedule with scatter search in the first phase, and according to this schedule, the rolling time of each slab, as well as the departure time of each slab from a certain reheating furnace, could be determined. Based on them, a heuristic was developed in the second phase to establish the schedules for each reheating furnace. That is, this two-phase method is a sequential scheduling method that cannot achieve the real integrated optimization of both the reheating furnace and the hot rolling. Therefore, in this paper, we investigated the integrated scheduling of reheating furnace and hot rolling by formulating it as multiobjective mixed-integer programming model and developed an improved MODE algorithm for this problem to achieve the simultaneous optimization of reheating furnace and hot rolling.

## 3. Problem Formulation

### 3.1. The Model.

$$\min \quad f_1 = \sum_{i=1}^{N} (d_i - b_i - r_{i,\max}), \tag{1}$$

$$\min \quad f_2 = \sum_{j=1}^{N-1} \sum_{i=1}^{N} \sum_{i'=1, i' \neq i}^{N} y_{ij} y_{i',j+1} C_{ii'}, \tag{2}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} = 1, \quad i = 1, 2, \ldots, N, \tag{3}$$

$$\sum_{i=1}^{N} x_{ijk} \leq 1, \quad j = 1, 2, \ldots, N, k = 1, 2, \ldots, K, \tag{4}$$

$$\sum_{j=1}^{N} y_{ij} = 1, \quad i = 1, 2, \dots, N, \tag{5}$$

$$\sum_{i=1}^{N} y_{ij} = 1, \quad j = 1, 2, \dots, N, \tag{6}$$

$$\sum_{i=1}^{N} x_{i,j+1,k}(b_i + h) + \left(1 - \sum_{i=1}^{N} x_{i,j+1,k}\right) \times M > \sum_{i'=1}^{N} x_{i'jk} b_{i'}, \quad j = 1, \dots, N-1, k = 1, \dots, \tag{7}$$

$$\sum_{i=1}^{N} x_{i,j+1,k} d_i + \left(1 - \sum_{i=1}^{N} x_{i,j+1,k}\right) \times M > \sum_{i'=1}^{N} x_{i'jk} d_{i'}, \quad j = 1, \dots, N-1, k = 1, \dots, K, \tag{8}$$

$$b_{i'} \geq d_i \times \left(\sum_{j=1}^{J} x_{ijk} + \sum_{j=J+Q}^{N} x_{i'jk} - 1\right), \quad i, i' = 1, \dots, \in N, i \neq i', J = 1, \dots, N-Q, k = 1, \dots, K, \tag{9}$$

$$d_i - b_i \geq r_{i,\min}, \quad i = 1, 2, \dots, N, \tag{10}$$

$$\sum_{i=1}^{N} y_{i,j+1} d_i \geq \sum_{i'=1}^{N} y_{i'j}(d_{i'} + p_{i'}), \quad i, i' = 1, 2, \dots, N, j = 1, 2, \dots, N-1, \tag{11}$$

$$\left|\sum_{i=1}^{N} y_{i,j+1} w_i - \sum_{i'=1}^{N} y_{i'j} w_{i'}\right| \leq W_{\max}, \quad j = 1, 2, \dots, N-1, \tag{12}$$

$$\left|\sum_{i=1}^{N} y_{i,j+1} t_i - \sum_{i'=1}^{N} y_{i'j} t_{i'}\right| \leq T_{\max}, \quad j = 1, 2, \dots, N-1, \tag{13}$$

$$x_{ijk}, y_{ij} \in \{0, 1\}, \quad i = 1, 2, \dots, N, j = 1, 2, \dots, N, k = 1, 2, \dots, K, \tag{14}$$

$$b_i \geq 0, \quad i = 1, 2, \dots, N. \tag{15}$$

The objective (1) is to minimize the total unnecessary heating time (or the total overheating time) of all slabs, which in turn can help to reduce the total energy consumption. We choose such an objective based on the following two considerations: firstly, overheating of slabs will deteriorate the product quality; secondly, energy consumption in the reheating furnace is very large in iron and steel industry and the reduction of energy consumption has become one of the key challenging issues in scheduling optimization [18, 19]. The objective (2) is to minimize the total changeovers in width, thickness, steel grade, and so on, between adjacent slabs in the hot rolling schedule so as to improve the product quality. As analyzed before in Section 1, the two objectives are generally conflicting with each other in practical production. Constraint (3) ensures that each slab will be reheated by exactly one reheating furnace in the reheating schedule, and constraint (4) guarantees that each position of the reheating schedule can hold at most one slab. Constraints (5) and (6) ensure that each slab must be processed in the hot rolling, and each position of the hot rolling schedule must hold exactly one slab. Constraints (7) and (8) require that the slab in the $j+1$-th position should be drawn in and drawn out after the slab arranged in the $j$-th position in the reheating schedule. Constraint (9) represents the reheating furnace capacity constraint requiring that a slab cannot be drawn in unless the head slab in the furnace is drawn out. Constraint (10) guarantees that each slab must be reheated for its required heating time. Constraint (11) ensures that a slab in the $j+1$-th positon of the hot rolling schedule cannot be drawn out from the furnace for processing in the hot rolling unless the slab in the $j$-th position of the hot rolling schedule has been completed through the hot rolling production. Constraints (12) and (13) are the maximum changeover constraints for width and thickness of adjacent slabs in the hot rolling schedule. Constraints (14) and (15) determine the value ranges for decision variables (please note that the value range for $d_i$ has been defined in constraint (10)).

## 4. Proposed Discrete Multiobjective Differential Evolution Algorithm

*4.1. Brief Introduction of Multiobjective Differential Evolution.* Different from single-objective optimization, the task of multiobjective optimization is to achieve a set of optimal nondominated solutions uniformly distributed in the objective space. Due to the inherent ability of evolving a swarm of solutions simultaneously at a generation, evolution algorithms have been widely adopted in multiobjective optimization [20, 21]. Among the multiobjective evolutionary algorithms (MOEAs), multiobjective differential evolution (MODE) has achieved a lot of successful applications, especially in practical industries, due to its simple but effective search mechanism [22–29].

The main procedure of classical MODE algorithm is illustrated in Figure 3, where $g$ and $g_{\max}$ are, respectively, the index of generations and the maximum available generations (i.e., the stopping criterion) and $X_{i,g}$ is the $i$-th solution in the population $P$ at the $g$-th generation. There are three main steps in MODE to generate a new solution, namely, mutation, crossover, and selection. In the mutation step, a perturbed vector $V_{i,g} = (v_{i,g}^1, v_{i,g}^2, \dots, v_{i,g}^D)$ is generated through a mutation operator for each solution $X_{i,g} = (x_{i,g}^1, x_{i,g}^2, \dots, x_{i,g}^D)$ with $D$ dimensions in $P$. After mutation, a trial solution $U_{i,g} = (u_{i,g}^1, u_{i,g}^2, \dots, u_{i,g}^D)$ is constructed from $X_{i,g}$ and $V_{i,g}$ according to

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & \text{if rand } (0,1) \leq C_r \text{ or } j = j_{\text{rand}}, \\ x_{i,g}^j, & \text{otherwise}, \end{cases} \tag{16}$$

```
Set g=0, initialize F and C_r, and create the initial population P consisting of n solutions.
while g < g_max do
    for each solution X_{i,g} in P do
        V_{i,g} := Mutation (X_{i,g}, F, mutation operator)      //mutation step
        U_{i,g} := Crossover (X_{i,g}, V_{i,g}, C_r)             //crossover step
        X_{i,g+1} := Selection (X_{i,g}, U_{i,g})                //selection step
    end for
    Set g = g + 1
end while
Output the non-dominated solutions in P.
```

Figure 3: Procedure of classical MODE.

where rand $(0, 1)$ is a random number uniformly generated in [0, 1] and $j_{\text{rand}}$ is a random integer generated in [1, D]. The adoption of $j_{\text{rand}}$ is to avoid the condition that $U_{i,g} = X_{i,g}$. Finally, in the selection step, the new solution for next generation $X_{i,g+1}$ is determined according to

$$X_{i,g+1} = \begin{cases} U_{i,g}, & \text{if } U_{i,g} \text{ dominates } X_{i,g}, \\ X_{i,g}, & \text{otherwise.} \end{cases} \tag{17}$$

For a multiobjective optimization problem, solution $X_1$ is said to dominate solution $X_2$ if all objectives of $X_1$ are better than or equal to those of $X_2$ and there exists at least one objective for which the objective value of $X_1$ is strictly better than that of $X_2$.

*4.2. Proposed MODE Algorithm.* The mathematical model defined by (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14), and (15) is a multiobjective discrete combinatorial optimization problem, and at the same time the integrated scheduling of reheating furnace and hot rolling further make the decision making more complex. This makes it difficult to directly apply MODEs in the literature to solve our problem. Therefore, we prefer to design an improved discrete MODE algorithm for this complex problem based on its characteristics. In the design of this MODE algorithm, several issues should be handled such as solution encoding and decoding that can consider the integrated scheduling characteristics, generation method of new solutions (including mutation, crossover, and selection) that can achieve good balance between exploration and exploitation, and management method of nondominated solutions during evolution that can guarantee a uniform distribution in the objective space.

*4.2.1. Solution Encoding and Decoding.* The decision making of reheating furnace scheduling is to allocate slabs to each furnace and at the same time determine their heating sequence, which makes it very hard for the solution encoding and decoding, as well as the neighborhood search designing. So we prefer to adopt the hot rolling schedule (i.e., a sequence of slabs $S = (s(1), s(2), \ldots, s(N))$ in which $s_i$ is the slab arranged at the $i$-th position) to encode a solution, and for a given solution, a heuristic with consideration of energy consumption minimization is presented to obtain the corresponding schedule of the reheating furnace. The main idea of this heuristic is that the current slab should be allocated to one of the first available furnaces that can result in the minimum energy consumption (i.e., the minimum overheating time). Let $E_k$ represent the earliest available time of furnace $k(k = 1, 2, \ldots, K)$ and $H(k)$ denote the current head slab (i.e., the slab holding the first position in furnace $k$, and please see Figure 2) in furnace $k$. Then, the detailed procedure of the decoding heuristic can be described in Figure 4.

*4.2.2. Population Initialization Method.* To get an initial population $P$ with good distribution in the objective space, a three-step initialization method is adopted: the classical NEH method [30], a modified multiobjective NEH method, and a random method. In the NEH, the initial slab sequence $S$ is determined based on practical processing constraints: first, sequence all the $N$ slabs in the nonascending order of width because it is preferred that the slab width changes from wide to narrow; second, for slabs with the same width, use the swap (swap two slabs) neighborhood search to improve the total changeover cost. In the following, we first give the population initialization method and then present the modified multiobjective NEH method. The population initialization method can be described as follows.

*Step 1.* Use the classical NEH to generate the first two solutions with the objectives (1) and (2), respectively, to obtain two extreme solutions, and then add them to $P$.

*Step 2.* Generate some solutions with the modified multiobjective NEH method and then add them to $P$. If the total number of initial solutions is larger than the population size $n_{\text{pop}}$, repeatedly remove the most crowded solution in $P$ until $|P| = n_{\text{pop}}$ and then terminate; otherwise, go to Step 3.

*Step 3.* Randomly select two slabs with similar width and thickness in a solution $S$ randomly selected from $P$ and perform a swap move on them to generate a new random solution (please note that two similar slabs mean that the swap of them will not result in violation for constraints (12) and

Input $S = (s(1), s(2), \ldots, s(K), \ldots, s(N))$. Set the first available time of each furnace to be zero (i.e., it is assumed that all the slabs have arrived at time zero).

Allocate the first $K$ slabs $s(1), s(2), \ldots, s(K)$ to furnace 1, 2, …, $K$, respectively, and set the drawing-in time of each slab to be $b_{s(i)} = 0$.

Update the earliest available time of each furnace $k$ ($k$=1, …, $K$) to be $E_k = h$.

Update the drawing-out time of each slab to be $d_{s(1)} = r_{s(1)}$ and $d_{s(k)} = \max\{d_{s(k-1)} + p_{s(k-1)}, r_{s(k)}\}$, $k$=2, …, $K$.

**for** $i := K+1$ **to** $N$ **do**

  Determine the set of furnaces (denoted as $U$) having the minimum first available time, i.e., each furnace $m$ in $U$satisfying $m = \text{argmin}\{E_k, k=1, 2, \ldots, K\}$.

  Calculate the overheating time of slab $s(i)$ when it is allocated to each furnace $m$ in $U$ (denoted as $O_{s(i)}$) according to $O_{s(i)} = \max\{d_{s(i)} - b_{s(i)} - r_{s(i)}, 0\}$.

  Select the furnace $k$ that can result in the least overheating time of slab $s(i)$, allocate slab $s(i)$ to the selected furnace, and update its drawing-in time $b_{s(i)} = E_k$.

  Update the first available time of the selected furnace $k$ (i.e., $E_k$) according to equation $E_k = \begin{cases} E_k + h, \text{ if furnace } k \text{ is not full} \\ d_{H(k)} + h, \text{ otherwise} \end{cases}$, in which $d_{H(k)}$ can be easily obtained because the drawing-out time of each slab is updated at each iteration.

  Update the drawing-out time of slab $s(i)$ to be $d_{s(i)} = \max\{d_{s(i-1)} + p_{s(i-1)}, b_{s(i)} + r_{s(i)}\}$.

**end for**

Return the re-heating furnace schedule.

Figure 4: Procedure of the decoding heuristic.

(13), i.e., the new random solution is still feasible). Repeat this procedure for $n_{\text{pop}} - |P|$ times.

Since the NEH method is very popular in scheduling problems, we only present the modified multiobjective NEH in the paper. Let $S = (s(1), s(2), \ldots, s(K), \ldots, s(N))$ be a given slab sequence, and $\Psi_i$ denote the nondominated partial solutions (a partial solution is defined as a sequence of $n(n < N)$ slabs) obtained in the $i$-th iteration. Since at each iteration a new slab in $S$ will be inserted into the nondominated partial solutions, each solution of $\Psi_j$ is in fact a sequence of $j$ slabs. The multiobjective NEH method can be given in Figure 5.

*4.2.3. Improved Discrete Mutation, Crossover, and Selection Operators.* For the permutation-based combinatorial optimization problems such as permutation flowshop scheduling, several kinds of discrete DE operators have been proposed [31–33] by defining novel discrete mutation and crossover operators. In [33], the computational results showed that the P-DDE [31] was superior to the W-DDE [32] and that the H-DDE [33], which could be viewed as an improvement version of W-DDE, was the best. However, in our problem, these discrete mutation and crossover operators cannot be directly adopted. The main reasons are as follows. On one hand, the new solution generated by the W-DDE and H-DDE may often be much different with comparison to the target solution. This may be good for the permutation flowshop scheduling problem since it can help to avoid being trapped in local optimal regions. But for the problem with constraints like ours, the new solution generated by them may often be unfeasible and it is hard to restore feasibility. On the other hand, the P-DDE used the discrete crossover operators of genetic algorithm such as the partially mapped crossover (PMX) [34]. When such crossover operator is used in our problem, we have to handle not only the infeasibility of duplicated jobs in the new solution but also the violations with constraints (12) and (13). Therefore, we prefer to develop some new mutation and crossover operators with consideration of our problem's characteristics.

*(1) Mutation Operator with Leader Selection*. In the proposed mutation operator, the adaptive leader selection mechanism is inspired by the hybrid MOEA proposed in [35], in which the concept of personal best and global best of particle swarm optimization was incorporated into MOEA. Its main idea is to adaptively select a nondominated solution in the external archive to generate the perturbed solution $V_{i,g}$. For a target solution $S_{i,g}$, the definition of the mutation operator with this mechanism is as follows:

$$V_{i,g} = \begin{cases} \text{insertion}\left(S_{P,g-1}\right), & \text{if } \text{rand}_1 < p_m \text{ and } \text{rand}_2 < p_s, \\ \text{insertion}\left(S_{G,g-1}\right), & \text{if } \text{rand}_1 < p_m \text{ and } \text{rand}_2 \geq p_s, \\ \text{insertion}\left(S_{r,g-1}\right), & \text{otherwise}, \end{cases} \tag{18}$$

where $S_{P,g-1}$ is the solution randomly selected from the three nondominated solutions in the external archive that are

```
Set Ψ_1 = {S = (s(1))}.
for i := 2 to N do
    Set Ψ_i = Φ.
    for each solution S in Ψ_{i-1} do
        for j := 1 to i do
            if inserting slab s(i) to the position j of S does not violate constraints (12)-(13) do
                Insert slab s(i) to the position j of S and generate a new solution S'.
                Add S' to Ψ_i.
            end if
        end for
    end for
    Refine Ψ_i so that only the non-dominated partial solutions persist.
end for
Add the non-dominated solutions in Ψ_N to P.
```

Figure 5: Procedure of the multiobjective NEH.

nearest to $S_{i,g}$, $S_{G,g-1}$ is the solution randomly selected from the whole external archive, and $S_{r,g-1}$ is the solution randomly selected from the population $P(r \neq i)$. $\text{rand}_1$ and $\text{rand}_2$ are two random number uniformly generated in [0, 1]. $p_m$ is called the mutation probability, and $p_s$ is the leader selection parameter. Equation (18) means that if $\text{rand}_1$ is less than $p_m$, we have two choices: perform a random insertion move (delete a slab on its current position and then insert it to another random position) on $S_{P,g-1}$ if $\text{rand}_2$ is less than $p_s$; otherwise, perform a random insertion move on $S_{G,g-1}$. If $\text{rand}_1$ is equal to or larger than $p_m$, then a random insertion move will be performed on another random solution in the current population to generate the perturbed solution. According to (18), it can be found that the first two items have good ability of accelerating the convergence speed since the new perturbed solution is generated around the external archive. Furthermore, it should be noted that the first item is focused on the local search because $S_{P,g-1}$ is one of the most nearest solutions to $S_{i,g}$ while the second item pays more attention to search diversity. Finally, the third item takes the advantage of search diversity and can help to avoid being trapped in local optimum. To make the mutation operator adaptively change the focus from exploitation to exploration, the value of parameter $p_s$ is updated according to

$$p_s = \frac{g_{\max} - g}{g_{\max}}, \tag{19}$$

where $g_{\max}$ denotes the maximum number of generations. That is, the first item will have more chance to be selected during the early stage of evolution while the second item will be selected with high probability during the later stage of evolution.

*(2) Crossover Operator with Feasibility Consideration.* The crossover operator follows the main idea of traditional DE and incorporates the consideration of feasibility maintenance. As mentioned above, there are two main feasibility requirements: one is defined by constraints (5) and (6) and another is defined by constraints (12) and (13). According to the two requirements, the crossover operator to generate the trial solution $U_{i,g} = (u_{i,1,g}, u_{i,2,g}, \ldots, u_{i,N,g})$ is defined in Figure 6, in which $s_{i,j,g}$ and $v_{i,j,g}$ denote the slab arranged at the $j$-th position of the target solution $S_{i,g}$ and the perturbed solution $V_{i,g}$, respectively. The main idea of this crossover operator is to determine the slab $u_{i,j,g}$ adaptively according to different scenarios. Please note that the generated trial solution $U_{i,g}$ will have no duplication of slabs, but may also have violations for constraints (12) and (13), which will be further fixed by swap or insertion moves. For example, if a slab in a certain position $j$ has violations for constraint (12) or (13) with its adjacent slabs, we will first try to swap it with another slab in this solution to restore feasibility. If the swap does not work, then we further try to use the insertion move: try to insert this slab to another position, and if this does not work, try to insert another slab to the position before or after this slab. Because the set of slabs for hot rolling scheduling are carefully selected by the scheduler from a large number of candidate slabs, the violation of constraints (12) and (13) can be easily fixed with the swap and insertion moves.

*(3) Selection Operator.* With the generated trial solution $U_{i,g}$, the selection operator is applied according to traditional MODE.

$$S_{i,g+1} = \begin{cases} U_{i,g}, & \text{if } U_{i,g} \text{ is not dominated by } S_{i,g}, \\ S_{i,g}, & \text{otherwise.} \end{cases} \tag{20}$$

*4.2.4. Guided Multiobjective Local Search.* In the multiobjective memetic algorithm proposed by Wang and Tang [36], a machine learning-based multiobjective local search method was developed. Its main idea was to first find the representative solutions in the external archive using clustering method

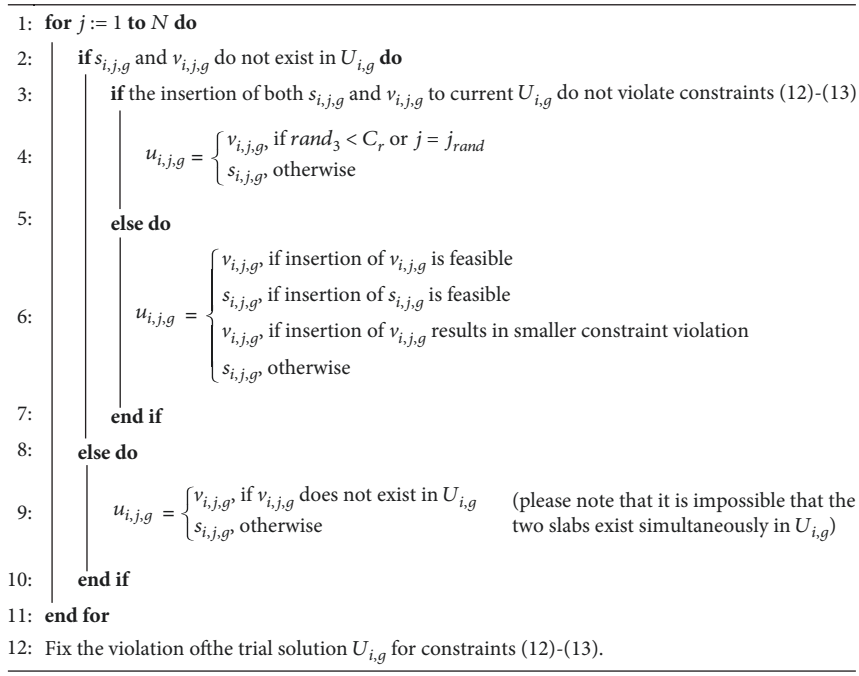


Figure 6: Procedure of the crossover operator with feasibility consideration.

and then perform a two-phase local search on the selected representative solutions. Considering that there are two objectives and many constraints in our problem, we propose another kind of multiobjective local search, which follows the main idea of the method in [36] but has less complexity. There are two main steps of our method: the first one is to select a solution with better diversity in the objective space from the external archive, and the second one is to apply a Pareto-based variable neighborhood descent search on the selected solution. For a given external archive, the detailed procedure of the guided multiobjective local search is given in Figure 7, in which the flag $\text{flag}_i = 1$ means that the $i$-th solution in the external archive has been selected for local search in previous generations (please note that we will memorize this flag for each solution in the external archive during evolution and that the flag will be set to be zero for a new solution added to the external archive).

In the above guided multiobjective local search, only the solution that has good diversity and has not been selected in previous generations can be selected for local search, which can help to improve the spread of nondominated solutions in the external archive. For the selected solution, there are two kinds of neighborhoods that are adopted, i.e., insertion (delete a slab in its current position and then insert it in another different position) and swap (swap two different slabs). Although the neighborhood sizes of them are $N(N-1)$ and $N(N-1)/2$, respectively, the number of practical moves is much smaller because only the feasible moves are permitted (line 8 and line 17). In addition, for each new generated solution $S_{\text{new}}$ in the local search, the decoding procedure in Section 4.2.1 will be used to construct the corresponding reheating furnace schedule before the evaluation.

*4.2.5. Update of External Archive.* The update of the external archive follows traditional MOEAs, that is, for a new solution $S_{\text{new}}$, it can be added into the external archive if and only if there is no solution in the external archive that can dominate $S_{\text{new}}$. After its addition, we will remove the solutions that are dominated by $S_{\text{new}}$ and may further remove the most crowded solution if the size of the external archive exceeds it maximum value $n_{\text{EA}}$.

*4.2.6. Whole Procedure of the Proposed Discrete Multiobjective Differential Evolution.* Based on the above sections, the procedure of the proposed discrete multiobjective differential evolution can be given in Figure 8, in which $g_{\max}$ denotes the maximum available generations.

## 5. Computational Results

*5.1. Test Problems and Parameter Setting.* To test the performance of the proposed algorithm, there are two kinds of test problems that are adopted in the experiments: a set of 90 randomly generated problems and a set of 4 practical production problems.

The random problems are generated according to practical data of heavy plate hot rolling as follows. The width of a slab is uniformly generated in [1500 mm, 2100 mm], and the thickness of a slab is uniformly generated in [200 mm, 300 mm]. The least residence time of a slab $r_{i,\min}$ in the reheating furnace is uniformly generated in [150 min, 250 min], and the corresponding maximum residence time of this slab $r_{i,\max}$ is set as the sum of $r_{i,\min}$ and a random number uniformly generated in [50 min, 100 min]. The rolling time of a slab is uniformly generated in [2.0 min, 4.0 min], the setup time between two adjacent slabs in the same

```
Calculate the crowding distance (as did in NSGA-II) of each solution in the external archive.
Sequence the solutions in the external archive in the non-ascending order of the crowding distance (a larger value of the crowding distance means better diversity).
From the first one of the sequence, select the one whose flag_i = 0 (denoted as S), and set its flag to be flag_i = 1 (i.e., this selected solution will not be selected in next local search).
Set Ψ = Φ, and for the selected solution perform the following two neighborhood searches.
for i := 1 to N do                                  //perform the insertion neighborhood search
    Delete the slab arranged in position i (denote this slab as s(i)) of S.
    for j := 1 to N do
        if j ≠ i and the insertion of s(i) in position j does not violate constraints (12)-(13) do
            Insert s(i) in position j and generate a new solution S_new.
            Evaluate the new solution S_new, set its flag to be zero, and add it into Ψ.
        end if
    end for
    Restore solution S.
end for
for i := 1 to N do                                  //perform the swap neighborhood search
    for j := 1 to N do
        if j ≠ i and the swap of s(i) and s(j) does not violate constraints (12)-(13) do
            Swap s(i) and s(j) of S and generate a new solution S_new.
            Evaluate the new solution S_new, set its flag to be zero, and add it into Ψ.
            Restore solution S.
        end if
    end for
end for
Update the external archive with Ψ.
```

Figure 7: Procedure of the guided multiobjective local search.

```
Set g=0, initialize parameters, and create the initial population P.
Update the external archive withthe initial population P.
while g < g_max is not reached do
    for each solution S_{i,g} in P do
        V_{i,g} := Mutation (S_{i,g}, p_m)          //apply mutation described in section 4.2.3.1
        U_{i,g} := Crossover (S_{i,g}, V_{i,g}, C_r)   //apply crossover described in section 4.2.3.2
        S_{i,g+1} := Selection (S_{i,g}, U_{i,g})   //apply selection described in section 4.2.3.3
    end for
    Update the external archive with the population P.
    Apply the multi-objective local search (section 4.2.4) on the external archive.
    Set g = g + 1.
end while
Output the non-dominated solutions in P.
```

Figure 8: Procedure of the discrete multiobjective differential evolution algorithm.

reheating furnace is set to 1 min, and the capacity of a reheating furnace is set to 30 slabs. The maximum permitted changes of width and thickness are set to 500 mm and 50 mm, respectively. The changeover cost between two slabs $i$ and $i'$ is calculated as $C_{ii'} = |w_i - w_{i'}| + 10|t_i - t_{i'}|$. There are three values for the number of slabs {100, 150, 200} and the number of reheating furnaces {3, 4, 5}, which can result in 9 different problem groups. For each problem group $N \times K$, 10 instances are randomly generated, and there is a total of 90 random instances that are used in the experiments.

The proposed discrete multiobjective differential evolution algorithm (represented as DMODE) is implemented in C++ and tested on a personal computer with Intel Core i7-6700 (3.4GHz) and 8 GB memory. Its parameters are set as

Table 1: Performance metric (GD, IGD, and HV) analysis of the mutation with adaptive leader selection.

| Problem | | GD | | | IGD | | | HV | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $N$ | $\text{DMODE}_{\text{no}}$ | $\text{DMODE}_{\text{leader}}$ | Sig | $\text{DMODE}_{\text{no}}$ | $\text{DMODE}_{\text{leader}}$ | Sig | $\text{DMODE}_{\text{no}}$ | $\text{DMODE}_{\text{leader}}$ | Sig |
| | 100 | 0.0046 | **0.0030** | + | 0.0391 | **0.0176** | + | 0.8103 | **0.8670** | + |
| 3 | 150 | 0.0042 | **0.0024** | + | 0.0439 | **0.0165** | + | 0.8117 | **0.8662** | + |
| | 200 | 0.0039 | **0.0015** | + | 0.0422 | **0.0233** | + | 0.8276 | **0.8712** | + |
| | 100 | 0.0038 | **0.0024** | + | 0.0303 | **0.0186** | + | 0.8249 | **0.8822** | + |
| 4 | 150 | 0.0066 | **0.0041** | + | 0.0432 | **0.0287** | + | 0.8254 | **0.8650** | + |
| | 200 | 0.0041 | **0.0009** | + | 0.0467 | **0.0306** | + | 0.8349 | **0.8808** | + |
| | 100 | 0.0053 | **0.0020** | + | 0.0315 | **0.0152** | + | 0.8418 | **0.9018** | + |
| 5 | 150 | 0.0046 | **0.0031** | + | 0.0398 | **0.0333** | + | 0.8292 | **0.8690** | + |
| | 200 | 0.0025 | **0.0011** | + | 0.0670 | **0.0302** | + | 0.8154 | **0.8877** | + |
| Average | | 0.0044 | **0.0023** | | 0.0426 | **0.0238** | | 0.8246 | **0.8768** | |

follows according to the experiments: the size of population is set to $n_{\text{pop}} = 100$, the size of the external archive is set to $n_{\text{EA}} = 100$, the mutation probability is set to $p_{\text{m}} = 0.8$, the crossover probability is set to $C_{\text{r}} = N(0.5, 0.1)$, and the maximum CPU time is set to $T_{\max} = 300$ seconds for instances with $N = 100$, $T_{\max} = 500$ seconds for instances with $N = 150$, and $T_{\max} = 600$ seconds for instances with $N = 200$. For each problem instance, a testing algorithm will be performed for 30 independent runs to collect statistical results. The parameters such as $p_{\text{m}}$ and $C_{\text{r}}$ are set based on a preliminary computation experiment. For example, the value of $p_{\text{m}}$ can be selected from three levels {0.7, 0.8, 0.9} while the value of $C_{\text{r}}$ can be selected from three levels {0.3, 0.5, 0.7}. Then for each level of $p_{\text{m}}$, the average IGD obtained by all levels of $C_{\text{r}}$ is taken for comparison to determine a good value of $p_{\text{m}}$. Similar method is used to determine the value of $C_{\text{r}}$.

*5.2. Performance Metrics.* In the experiment, three kinds of popular performance metrics are adopted [21], i.e., *General Distance (GD)*, *Inverted Generational Distance* (*IGD*), and *Hypervolume (HV)*.

The GD metric can be used to evaluate how close the Pareto front obtained by a certain algorithm (e.g., $A$) is to the true (or reference) Pareto optimal front (e.g., $A_{\text{opt}}$). Its definition is given in

$$\text{GD} = \frac{\sqrt{\sum_{i=1}^{|A|} d_i^2}}{|A|}, \tag{21}$$

in which $|A|$ is the sum of nondominated solutions in $A$ and $d_i$ represents the Euclidean distance in the objective space between the $i$-th solution in $A$ and its nearest solution in $A_{\text{opt}}$.

On the contrary, the definition of IGD is presented in

$$\text{IGD} = \frac{\sqrt{\sum_{i=1}^{|A_{\text{opt}}|} d_i^2}}{|A_{\text{opt}}|}, \tag{22}$$

in which $|A_{\text{opt}}|$ represents the sum of nondominated solutions in the true (or reference) Pareto front $A_{\text{opt}}$ and $d_i$ denotes the Euclidean distance in the objective space between the $i$-th solution in $A_{\text{opt}}$ and its nearest solution in $A$. The IGD metric can be used to evaluate not only the distance between $A$ and $A_{\text{opt}}$ but also the distribution of $A$. For example, if all solutions in $A$ are trapped in a local region that is very close to $A_{\text{opt}}$ in the objective space, then $A$ will have a very small (or good) value of *GD* metric, but it will have a very large (or bad) value of IGD metric because it has bad distribution.

Similar to IGD, the HV metric can also evaluate both the closeness and distribution of $A$ by accumulating the area covered by each solution in $A$ with comparison to a reference point (often set as [1.0, 1.0] after normalization).

Since the true Pareto optimal front of each problem cannot be obtained, we prefer to use the reference Pareto optimal front in our experiment. The reference Pareto optimal front is obtained by selecting the true nondominated solutions from the union of $A$ obtained by all the testing algorithms in the experiment (i.e., all the algorithms tested in Section 6). Moreover, we will normalize the objective values when calculating the performance metrics.

*5.3. Performance Analysis of the Mutation with Adaptive Leader Selection.* In this section, we first carried out experiments using the random problems to evaluate the performance of the adaptive leader selection method used in the mutation procedure.

The computational results between our algorithm using the mutation operator with leader selection (denoted as $\text{DMODE}_{\text{leader}}$) and the one using the tradition mutation operator without leader selection (denoted as $\text{DMODE}_{\text{no}}$) are given in Table 1. In $\text{DMODE}_{\text{no}}$, the mutation operator only uses the third item in (18). In Table 1, the performance value of each performance metric (GD, IGD, and HV) for each problem group is the average of the ten problems in this group and the better result is shown in a bold style. In addition, the pair-wise $T$-test with a confidence level of 95% is used to test whether the difference of performance

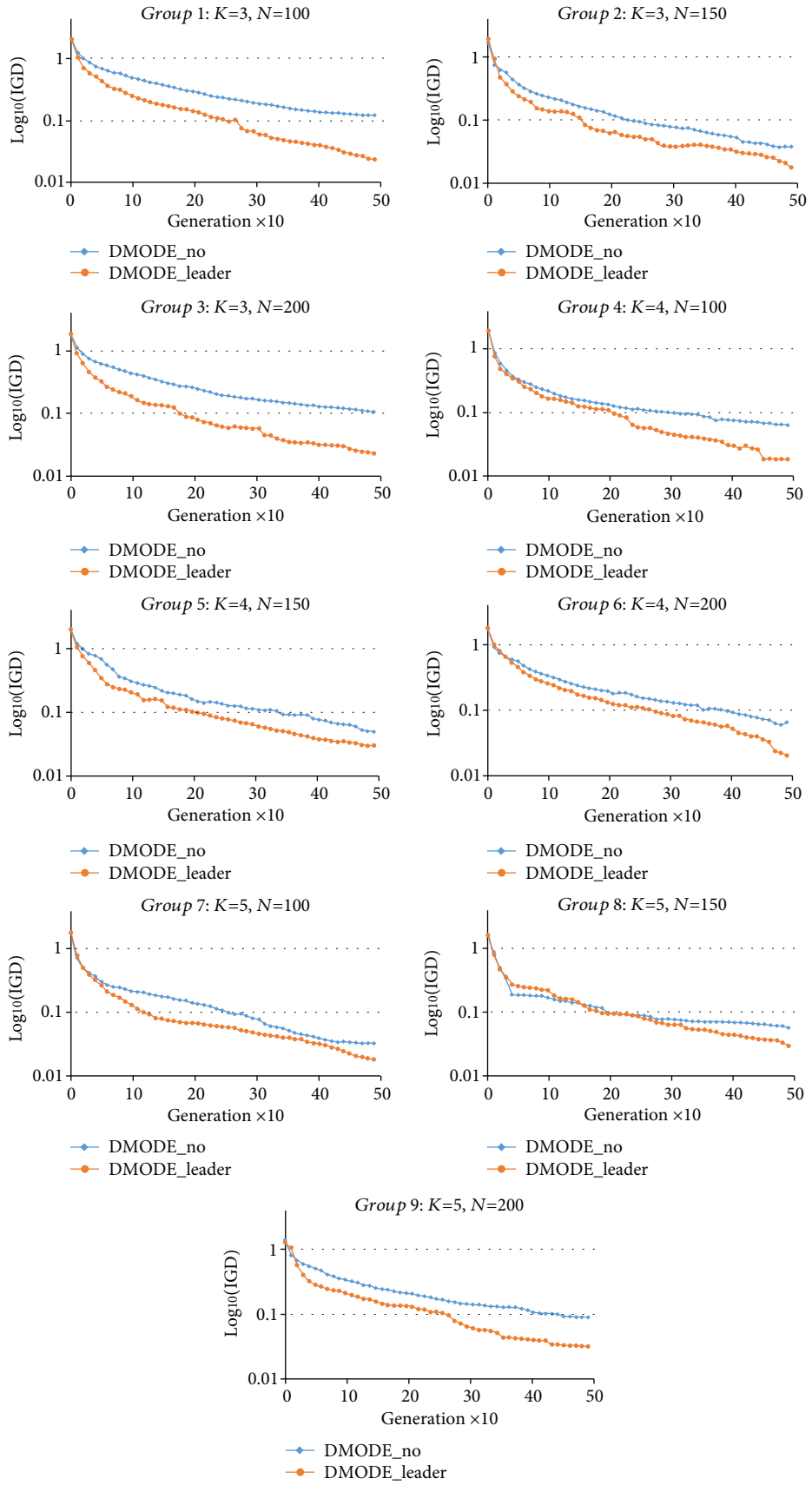


Figure 9: Evolution process of the IGD metric for the two algorithms.

Table 2: Performance metric (GD, IGD, and HV) analysis of the multiobjective local search.

| Problem | | GD | | | IGD | | | HV | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $N$ | DMODE$_{_noLS}$ | DMODE$_{_LS}$ | Sig | DMODE$_{_noLS}$ | DMODE$_{_LS}$ | Sig | DMODE$_{_noLS}$ | DMODE$_{_LS}$ | Sig |
| | 100 | 0.0110 | **0.0030** | + | 0.1085 | **0.0176** | + | 0.7092 | **0.8670** | + |
| 3 | 150 | 0.0061 | **0.0024** | + | 0.0714 | **0.0165** | + | 0.7768 | **0.8662** | + |
| | 200 | 0.0062 | **0.0015** | + | 0.0808 | **0.0233** | + | 0.7692 | **0.8712** | + |
| | 100 | 0.0091 | **0.0024** | + | 0.1062 | **0.0186** | + | 0.7213 | **0.8822** | + |
| 4 | 150 | 0.0080 | **0.0041** | + | 0.0902 | **0.0287** | + | 0.7658 | **0.8650** | + |
| | 200 | 0.0072 | **0.0009** | + | 0.1048 | **0.0306** | + | 0.7473 | **0.8808** | + |
| | 100 | 0.0073 | **0.0020** | + | 0.0801 | **0.0152** | + | 0.7810 | **0.9018** | + |
| 5 | 150 | 0.0095 | **0.0031** | + | 0.0982 | **0.0333** | + | 0.7242 | **0.8690** | + |
| | 200 | 0.0109 | **0.0011** | + | 0.1506 | **0.0302** | + | 0.6630 | **0.8877** | + |
| Average | | 0.0084 | **0.0023** | | 0.0990 | **0.0238** | | 0.7398 | **0.8768** | |

metrics between two testing algorithms are significant or not. The symbol "+" in the column *Sig* means that the performance difference is significant while the symbol "–" means that there is no significant difference between their performance metrics.

From Table 1, it can be seen that the DMODE using the mutation with leader selection obtains significantly better results of every metric for all the problem groups. In particular, as the problem size ($K$ and $N$) increases, the performance difference tends to increase. For example, for the largest problem group with $K = 5$ and $N = 200$, the GD and IGD metric values obtained by DMODE$_{\text{leader}}$ are less than a half of those obtained by DMODE$_{\text{no}}$.

Figure 9 gives the evolution process of IGD obtained by the two algorithms (for simplicity, we use only an instance randomly selected from the ten instances in each problem group). It is clear that the DMODE can achieve a better convergence speed and maintain good distribution during evolution process with the help of the mutation with adaptive leader selection. Moreover, we can see that the IGD obtained by DMODEleader can still decrease much faster that obtained by the DMODEno. The main reasons can be analyzed as follows. The first item of (18) focuses on the local search of promising regions, the second item of (18) integrates the local search and diversity maintenance, and the last item of (18) focuses on diversity maintenance. Overall, their adaptive selection can help to achieve a better balance of exploitation and exploration.

*5.4. Performance Analysis of the Multiobjective Local Search.* In this section, we further test the efficiency of the multiobjective local search for the DMODE algorithm. The computational results between our algorithm with the local search (denoted as DMODE$_{_LS}$) and the one without the local search (denoted as DMODE$_{_noLS}$) are shown in Table 2, in which the performance metrics GD, IGD, and HV are given, and the evolution process of IGD metric obtained by the two algorithms is illustrated in Figure 10.

From Table 2, similar observations can be obtained. The multiobjective local search can help DMODE to achieve significantly better results for all problem groups with respect to each performance metric. According to Figure 10, it is clear that the IGD value obtained by DMODE$_{_noLS}$ decreases very slowly at the middle and last evolution process and the difference between the IGD values between DMODE$_{_noLS}$ and DMODE$_{_LS}$ tends to increase during the evolution process, which shows the efficiency of multiobjective local search.

*5.5. Comparison with the Other MOEAs on Random Problems.* To further evaluate the performance of the proposed DMODE algorithm, in this section, we compare it with the other powerful MOEAs in the literature. Since NSGA-II was often adopted in practical optimization problems such as the hot rolling scheduling [15], it is selected as the comparison algorithm in this experiment. To make NSGA-II applicable in our problem, the following modifications are made: first, the NSGA-II adopts the encoding and decoding method described in Section 4.2.1; second, the PMX crossover is used to generate new offsprings and the random *insertion* move is taken as the mutation operator; third, since the new offsprings generated by the crossover and mutation operator may be infeasible, a feasibility restoring method based on swap moves is used to guarantee feasibility; fourth, the guided multiobjective local search in Section 4.2.4 is used at each generation to improve a nondominated solution randomly selected from the current population. The population size of NSGA-II is set to 100, and the stopping criterion is the same one used in our DMODE.

Besides the NSGA-II, we also take the multiobjective iterated local search (MOILS) proposed by Xu et al. [37] as the comparison algorithm because this algorithm shows better results than the others for the biobjective permutation flowshop scheduling problem. The MOILS adopts the block-based multiobjective local search in which a block of jobs (slabs) will be removed first and then reinserted into the partial schedule, which may result in many infeasible solutions for our problem. So we use the guided multiobjective local search in Section 4.2.4 to replace the block-based multiobjective local search in MOILS. Similarly, the MOILS also adopts the same encoding and decoding method in Section 4.2.1. Since at each generation of MOILS there is only one solution selected for local search, we also use the crossover and mutation operators to evolve its external

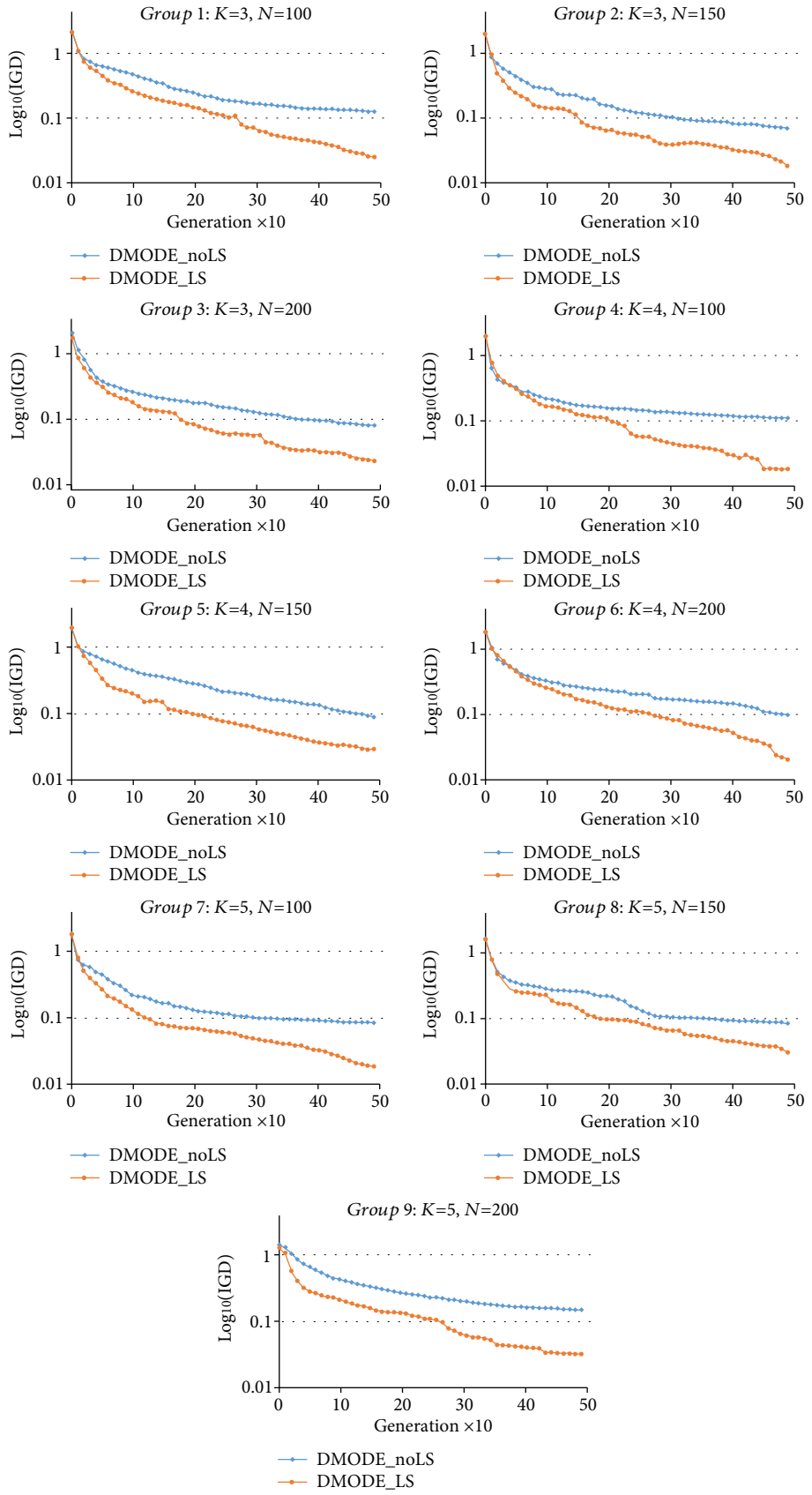


Figure 10: Evolution process of the IGD metric for the two algorithms with and without local search.

Table 3: Performance metrics obtained by NSGA-II, MOILS, and DMODE for random problems.

| Problem | | GD | | | IGD | | | HV | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $N$ | NSGA-II | MOILS | DMODE | NSGA-II | MOILS | DMODE | NSGA-II | MOILS | DMODE |
| 3 | 100 | $0.0057^{+}$ | $0.0036^{=}$ | **0.0030** | $0.1090^{+}$ | $0.0195^{=}$ | **0.0176** | $0.6541^{+}$ | **0.8681**$^{=}$ | 0.8670 |
| | 150 | $0.0044^{+}$ | $0.0038^{+}$ | **0.0024** | $0.0822^{+}$ | $0.0431^{+}$ | **0.0165** | $0.6868^{+}$ | $0.8361^{+}$ | **0.8662** |
| | 200 | $0.0041^{+}$ | $0.0058^{+}$ | **0.0015** | $0.1169^{+}$ | $0.0403^{+}$ | **0.0233** | $0.6781^{+}$ | $0.8254^{+}$ | **0.8712** |
| 4 | 100 | $0.0068^{+}$ | $0.0044^{+}$ | **0.0024** | $0.1250^{+}$ | $0.0289^{+}$ | **0.0186** | $0.7195^{+}$ | $0.8660^{=}$ | **0.8822** |
| | 150 | $0.0068^{+}$ | **0.0039**$^{=}$ | 0.0041 | $0.1044^{+}$ | **0.0276**$^{=}$ | 0.0287 | $0.7522^{+}$ | **0.8677**$^{=}$ | 0.8650 |
| | 200 | $0.0042^{+}$ | $0.0040^{+}$ | **0.0009** | $0.1318^{+}$ | $0.0473^{+}$ | **0.0306** | $0.7003^{+}$ | $0.8358^{+}$ | **0.8808** |
| 5 | 100 | $0.0092^{+}$ | $0.0035^{+}$ | **0.0020** | $0.1050^{+}$ | $0.0329^{+}$ | **0.0152** | $0.7281^{+}$ | $0.8737^{+}$ | **0.9018** |
| | 150 | **0.0030**$^{=}$ | $0.0048^{+}$ | 0.0031 | $0.1168^{+}$ | $0.0464^{+}$ | **0.0333** | $0.6951^{+}$ | $0.8135^{+}$ | **0.8690** |
| | 200 | $0.0034^{+}$ | $0.0062^{+}$ | **0.0011** | $0.1658^{+}$ | $0.0807^{+}$ | **0.0302** | $0.6267^{+}$ | $0.8203^{+}$ | **0.8877** |
| Average | | 0.0053 | 0.0044 | **0.0023** | 0.1174 | 0.0407 | **0.0238** | 0.6934 | 0.8452 | **0.8768** |

archive at each generation to ensure that MOILS have similar computational effort with our DMODE.

The computational results of the three algorithms on the random problem groups are shown in Table 3, in which the symbols "+," "=," and "–" mean that DMODE is significantly better than, equal to, and worse than a comparison algorithm for a problem group, respectively. The best result for each problem group is shown in a bold style.

From Table 3, it can be observed that the proposed DMODE algorithm achieves much better results of average GD, IGD, and HV metrics than NSGA-II and MOILS, which means that the proposed DMODE algorithm is effective for the problem considered in this paper. More specifically, for the GD metric, the DMODE algorithm obtains significantly better results for 8 problem groups than NSGA-II and for 7 problem groups than MOILS. Although NSGA-II obtains the best GD result for problem group $K = 5$ and $N = 150$, it can be seen that its IGD and HV metrics are quite worse than ours. The reason is that most solutions in the Pareto front obtained by NSGA-II are trapped in local optimal regions that are close to the reference Pareto front and the distribution of its Pareto front is not good (please see group 8 in Figure 11). For the IGD metric, the proposed DMODE obtains significantly better results for all the 9 problem groups than NSGA-II and for 7 problem groups than MOILS. For the HV metric, the DMODE can achieve significantly better results for all the 9 problem groups than NSGA-II and for 6 problem groups than MOILS.

The graphical results of the Pareto fronts obtained by NSGA-II, MOILS, and DMODE for each problem group are illustrated in Figure 11 (for simplicity, only one problem instance is randomly selected for comparison). From this figure, it is clear that the Pareto fronts obtained by our DMODE have both good convergence and distribution for most problems. In addition, the MOILS can obtain Pareto fronts with better convergence and distribution than the NSGA-II for most problems. The evolution process of IGD metric for each algorithm is also given in Figure 12, from which it can be seen that our DMODE can achieve much faster IGD decrease for all the problem groups except group 5. The MOILS can succeed to obtain the best IGD performance for problem group 5.

*5.6. Comparison with the Other MOEAs on Practical Problems.* In this section, we further test the performance of NSGA-II, MOILS, and our DMODE on the four practical problem instances with $K = 3$ and $N = 104$, 145, 202, and 225. The computational results of the GD, IGD, and HV metrics are shown in Table 4, and the Pareto fronts obtained by them are illustrated in Figure 13.

From Table 4, it can be observed that the DMODE obtains that best average value for each performance metric and it succeeds to achieve the best results for all the performance metrics for the last three problem instances with larger size. Although MOILS obtains the best result of GD metric for the first problem case, from the IGD and HV metrics, it can be concluded that the reason is that the Pareto front obtained by MOILS does not have even distribution in the objective space, which can be supported by the first figure in Figure 13. From Figure 13, it is quite clear that the proposed DMODE algorithm can achieve the Pareto fronts with better convergence and distribution, which means that the DMODE algorithm can be used effectively to solve the practical problems.

After the Pareto near-optimal solutions are obtained by our algorithm, the scheduler can select an appropriate one from them according to the following rule. First, he can determine a threshold value for the changeover cost in hot rolling, which means that the solutions whose changeover costs are less than this threshold value are acceptable. Second, he selects the solution with the minimum overheating time to act as the application schedule.

## 6. Conclusions

This paper investigates a new production scheduling problem in the hot rolling mill, namely, the integrated scheduling of reheating furnace and hot rolling, with the motivation to achieve the optimization of reheating scheduling and hot rolling scheduling simultaneously. Two important but conflicting objectives are considered in this problem: minimization of total changeover cost between adjacent slabs in the hot rolling to improve coil quality and minimization of total overheating times of slabs in the reheating furnace to reduce energy consumption. We formulate this problem as a

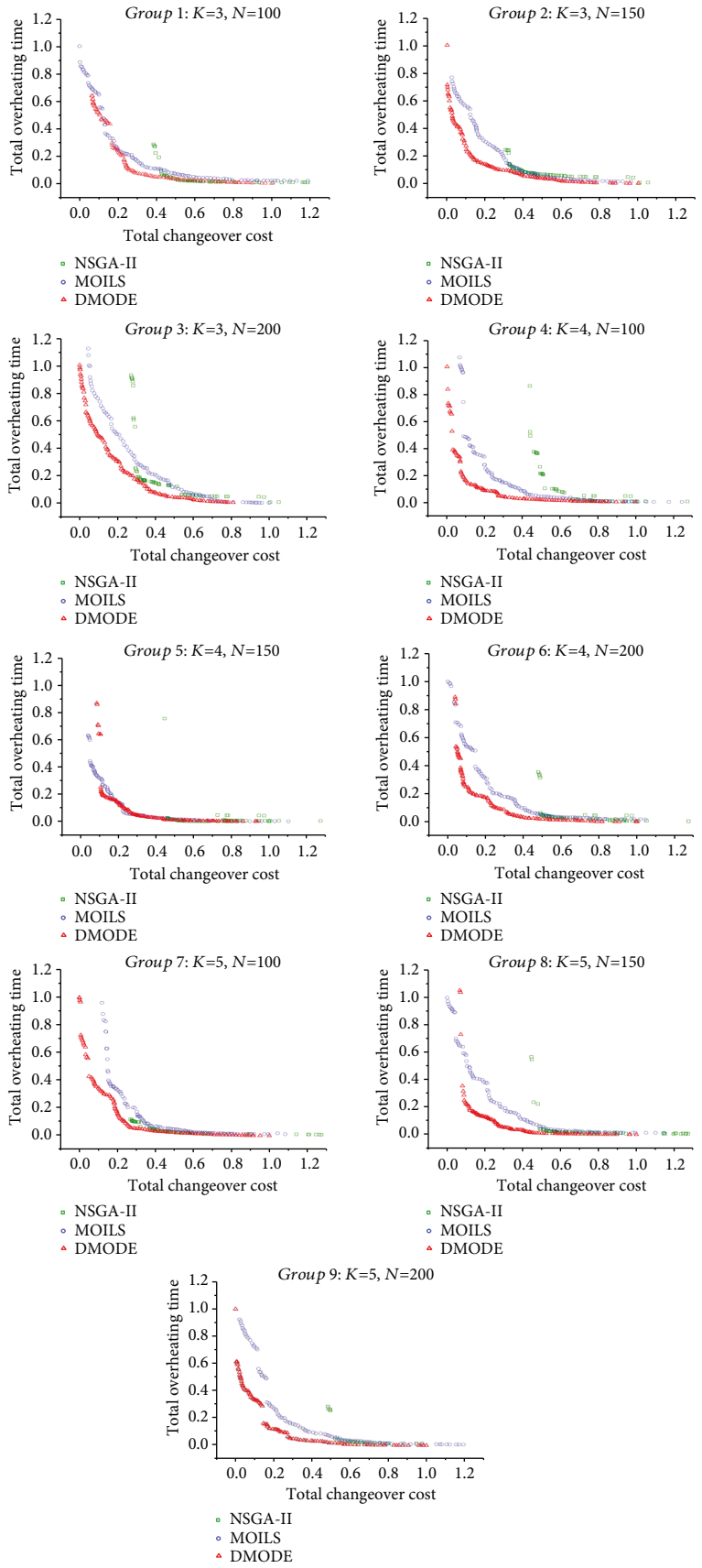


Figure 11: Pareto front obtained by NSGA-II, MOILS, and DMODE.

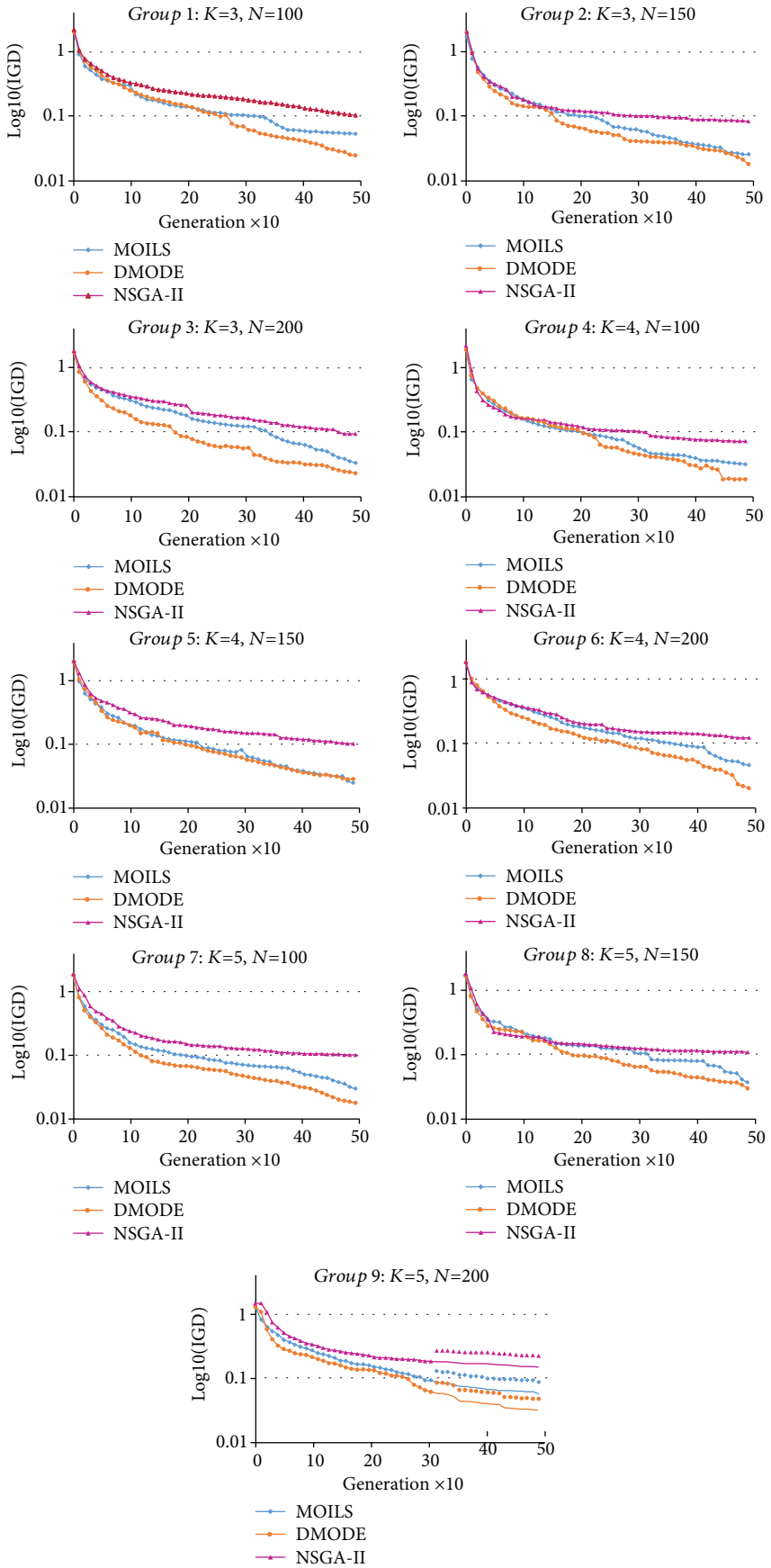


Figure 12: Evolution process IGD metric obtained by NSGA-II, MOILS, and DMODE.

Table 4: Performance metrics obtained by NSGA-II, MOILS, and DMODE for practical problems.

| Problem | | GD | | | IGD | | | HV | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $N$ | NSGA-II | MOILS | DMODE | NSGA-II | MOILS | DMODE | NSGA-II | MOILS | DMODE |
| 3 | 104 | 0.0045 | **0.0007** | 0.0032 | 0.145 | 0.0543 | **0.0189** | 0.7697 | 0.8599 | **0.8636** |
| | 145 | 0.0014 | 0.0020 | **0.0009** | 0.0666 | 0.0484 | **0.0324** | 0.8194 | 0.9327 | **0.9611** |
| | 202 | 0.003 | 0.0155 | **0.0019** | 0.0869 | 0.126 | **0.0382** | 0.8354 | 0.754 | **0.8833** |
| | 225 | 0.0067 | 0.0049 | **0.0007** | 0.0908 | 0.0652 | **0.0248** | 0.8476 | 0.913 | **0.9331** |
| Average | | 0.0039 | 0.0056 | **0.0017** | 0.0973 | 0.0735 | **0.0286** | 0.8180 | **0.8649** | **0.9103** |

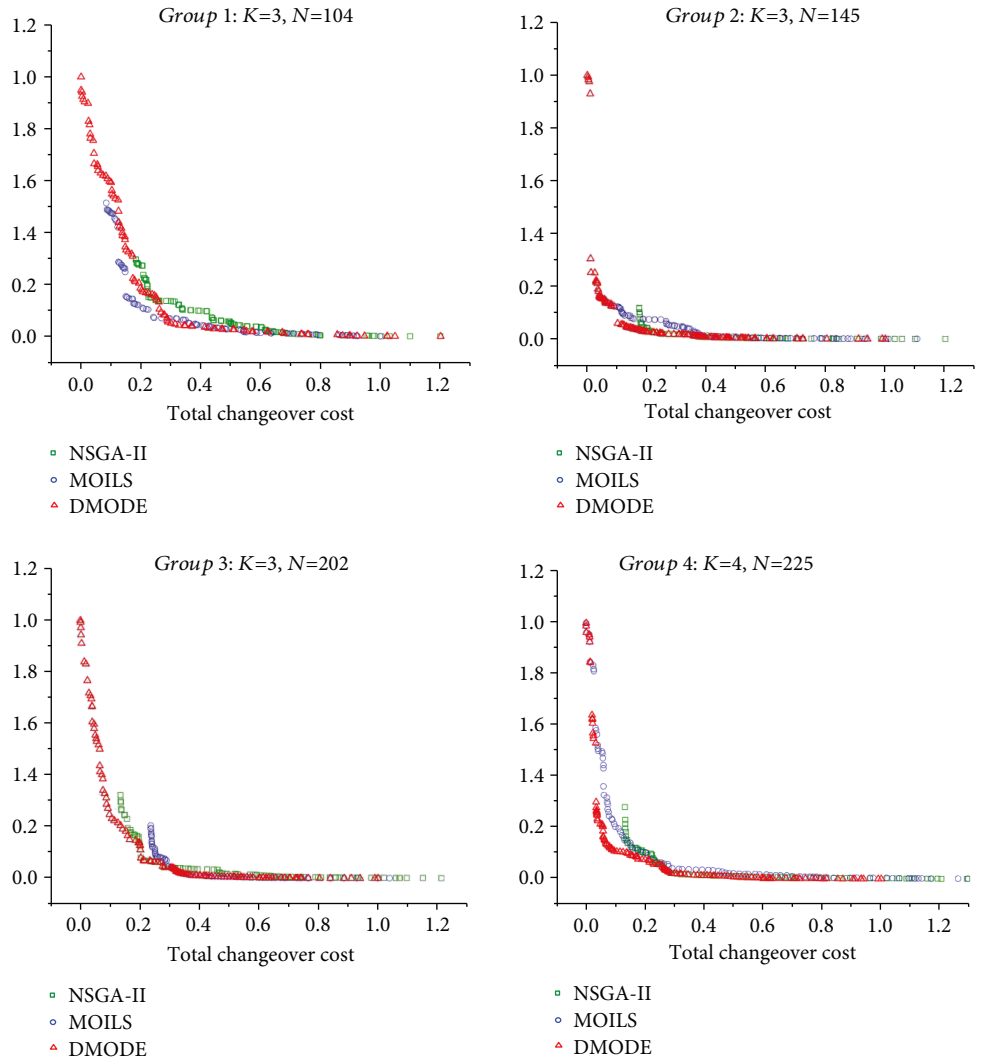


Figure 13: Evolution process IGD metric obtained by NSGA-II, MOILS, and DMODE.

biobjective mixed-integer programming model and proposed a discrete MODE algorithm to solve it. To make the MODE applicable, a novel encoding and decoding method is developed, in which the energy consumption reduction is taken into account in the decoding process. To handle the complex constraints, new discrete mutation and crossover operators are proposed, in which the balance between exploitation and exploration is tackled in the discrete mutation operator and the feasibility maintenance is integrated in the crossover operator. A guided multiobjective local search is also used to further improve the search efficiency of the DMODE. In the experiment, two kinds of problem instances are used, i.e., random instances and practical instances. The computational results illustrate the efficiency of the mutation and crossover operators and the guided local search. Further results show that the proposed DMODE algorithm is superior to some other powerful algorithms such as NSGA-II and MOILS for both random and practical problems. Future research may be the extension and application of the DMODE algorithm in practical hot rolling mill.

## Parameters

$i, i'$: Index of slabs
$j$: Index of positions
$k$: Index of furnaces
$N$: Sum of slabs for reheating and hot rolling
$K$: Sum of available reheating furnaces
$r_{i,\min}$: Least residence time of slab $i$ in the reheating furnace
$r_{i,\max}$: Maximum residence time of slab $i$ in the reheating furnace
$h$: Setup time between consecutive fed-in slabs in the same furnace
$p_i$: Processing time of slab $i$ in the hot rolling
$w_i$: Width of slab $i$
$W_{\max}$: Maximum change of width between adjacent slabs
$t_i$: Thickness of slab $i$
$T_{\max}$: Maximum change of thickness between adjacent slabs
$Q$: Maximum number of slabs that can be held in a furnace (i.e., furnace capacity)
$C_{ii'}$: Changeover cost between two adjacent slabs $i$ and $i'$ in the hot rolling
$M$: A very large number.

*Decision Variables*

$$x_{ijk} = \begin{cases} 1, & \text{if slab } i \text{ is arranged at the } j\text{-th position in furnace } k, \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \dots, N, k = 1, \dots, K,$$

$$y_{ij} = \begin{cases} 1, & \text{if slab } i \text{ is arranged at the } j\text{-th position in hot rolling schedule,} \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \dots, N,$$

$b_i$: Drawing-in time of slab $i$ in a certain reheating furnace
$d_i$: Drawing-out (or departure) time of slab $i$ from a certain reheating furnace, which is also the starting time for hot rolling.

## Data Availability

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

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This research is supported by the National Natural Science Foundation of China (Nos. 71602143, 61403277, and 61573086), the Program for Innovative Research Team of the University of Tianjin (No. TD13-5038), the Tianjin Natural Science Foundation (18JCYBJC22000), and the Tianjin Science and Technology Committee Correspondent Project (18JCTPJC62600).

## References

[1] C. Mui, E. Osinski, J. A. Meech, and P. V. Barr, "A SCADA-based expert system to provide delay strategies for a steel billet reheat furnace," in *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99 (Cat. No.99EX296)*, pp. 111–118, Honolulu, HI, USA, 1999.

[2] J. S. Broughton, M. Mahfouf, and D. A. Linkens, "A paradigm for the scheduling of a continuous walking beam reheat furnace using a modified genetic algorithm," *Materials and Manufacturing Processes*, vol. 22, no. 5, pp. 607–614, 2007.

[3] P. P. Mohanty, "An agent-oriented approach to resolve the production planning complexities for a modern steel manufacturing system," *International Journal of Advanced Manufacturing Technology*, vol. 24, no. 3-4, pp. 199–205, 2004.

[4] X. Wang and L. Tang, "Scheduling a single machine with multiple job processing ability to minimize makespan," *Journal of the Operational Research Society*, vol. 62, no. 8, pp. 1555–1565, 2011.

[5] L. Tang, H. Ren, and Y. Yang, "Reheat furnace scheduling with energy consideration," *International Journal of Production Research*, vol. 53, no. 6, pp. 1642–1660, 2015.

[6] L. Lopez, M. W. Carter, and M. Gendreau, "The hot strip mill production scheduling problem: a tabu search approach," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 317–335, 1998.

[7] E. Balas and C. H. Martin, *ROLL-A-ROUND: Software Package for Scheduling the Rounds of a Rolling Mill*, Copyright Balas and Martin Associates, Pittsburg, PA, USA, 1985.

[8] P. Cowling, "A flexible decision support system for steel hot rolling mill scheduling," *Computers & Industrial Engineering*, vol. 45, no. 2, pp. 307–321, 2003.

[9] M. R. Yadollahpour, M. Bijari, S. Kavosh, and M. Mahnam, "Guided local search algorithm for hot strip mill scheduling problem with considering hot charge rolling," *The International Journal of Advanced Manufacturing Technology*, vol. 45, no. 11-12, pp. 1215–1231, 2009.

[10] A. l. Chen, G. K. Yang, and Z. M. Wu, "Production scheduling optimization algorithm for the hot rolling processes,"

*International Journal of Production Research*, vol. 46, no. 7, pp. 1955–1973, 2008.

[11] J. Zhao, W. Wang, Q. Liu, Z. Wang, and P. Shi, "A two-stage scheduling method for hot rolling and Its application," *Control Engineering Practice*, vol. 17, no. 6, pp. 629–641, 2009.

[12] X. Wang and L. Tang, "Integration of batching and scheduling for hot rolling production in the steel industry," *International Journal of Advanced Manufacturing Technology*, vol. 36, no. 5-6, pp. 431–441, 2008.

[13] N. Chakraborti, B. Siva Kumar, V. Satish Babu, S. Moitra, and A. Mukhopadhyay, "A new multi-objective genetic algorithm applied to hot-rolling process," *Applied Mathematical Modelling*, vol. 32, no. 9, pp. 1781–1789, 2008.

[14] S. J. Jia, J. Yi, G. K. Yang, B. Du, and J. Zhu, "A multi-objective optimisation algorithm for the hot rolling batch scheduling problem," *International Journal of Production Research*, vol. 51, no. 3, pp. 667–681, 2013.

[15] M. Tan, H. L. Yang, B. Duan, Y. X. Su, and F. He, "Optimizing production scheduling of steel plate hot rolling for economic load dispatch under time-of-use electricity pricing," *Mathematical Problems in Engineering*, vol. 2017, Article ID 1048081, 13 pages, 2017.

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

[17] L. Tang and X. Wang, "A two-phase heuristic for the production scheduling of heavy plates in steel industry," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 104–117, 2010.

[18] M. Rager, C. Gahm, and F. Denz, "Energy-oriented scheduling based on evolutionary algorithms," *Computers & Operations Research*, vol. 54, pp. 218–231, 2015.

[19] L. Merkert, I. Harjunkoski, A. Isaksson, S. Säynevirta, A. Saarela, and G. Sand, "Scheduling and energy – industrial challenges and opportunities," *Computers & Chemical Engineering*, vol. 72, pp. 183–198, 2015.

[20] C. A. Coello Coello, "Multi-objective evolutionary algorithms in real-world applications: some recent results and current challenges," in *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, D. Greiner, B. Galván, J. Périaux, N. Gauger, K. Giannakoglou, and G. Winter, Eds., pp. 3–18, Springer International Publishing, Cham, 2015.

[21] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[22] E. Mezura-Montes, M. Reyes-Sierra, and C. A. Coello Coello, "Multi-objective optimization using differential evolution: a survey of the state-of-the-art," in *Advances in Differential Evolution*, U. K. Chakraborty, Ed., pp. 173–196, Springer, Berlin Heidelberg, 2008.

[23] J. Cheng, G. G. Yen, and G. Zhang, "A grid-based adaptive multi-objective differential evolution algorithm," *Information Sciences*, vol. 367-368, pp. 890–908, 2016.

[24] S. Ramesh, S. Kannan, and S. Baskar, "An improved generalized differential evolution algorithm for multi-objective reactive power dispatch," *Engineering Optimization*, vol. 44, no. 4, pp. 391–405, 2012.

[25] X. Wang and L. Tang, "Multiobjective operation optimization of naphtha pyrolysis process using parallel differential evolution," *Industrial & Engineering Chemistry Research*, vol. 52, no. 40, pp. 14415–14428, 2013.

[26] S. Sharma and G. P. Rangaiah, "An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes," *Computers & Chemical Engineering*, vol. 56, pp. 155–173, 2013.

[27] Y. Shen and Y. Wang, "Operating point optimization of auxiliary power unit using adaptive multi-objective differential evolution algorithm," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 115–124, 2017.

[28] X. Wang and L. Tang, "An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization," *Information Sciences*, vol. 348, pp. 124–141, 2016.

[29] L. Tang, X. Wang, and Z. Dong, "Adaptive multiobjective differential evolution with reference axis vicinity mechanism," *IEEE Transactions on Cybernetics*, pp. 1–15, 2018.

[30] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[31] 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.

[32] 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.

[33] Q. K. Pan, L. Wang, L. Gao, and W. D. Li, "An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers," *Information Sciences*, vol. 181, no. 3, pp. 668–685, 2011.

[34] C. L. Chen, R. V. Neppalli, and N. Aljaber, "Genetic algorithms applied to the continuous flow shop problem," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 919–929, 1996.

[35] L. Tang and X. Wang, "A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 20–45, 2013.

[36] X. Wang and L. Tang, "A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 79, pp. 60–77, 2017.

[37] J. Xu, C.-C. Wu, Y. Yin, and W.-C. Lin, "An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 52, pp. 39–47, 2017.