mathematics

MDPI

Article

Improved Differential Evolution Algorithm for Slab Allocation and Hot-Rolling Scheduling Integration Problem

Lulu Song [1,2], Ying Meng [1], Qingxin Guo [1,3,4,*] and Xinchang Gong [5]

[1] National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China; 1510389@stu.neu.edu.cn (L.S.); mengying@ise.neu.edu.cn (Y.M.)
[2] Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang 110819, China
[3] Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Shenyang 110819, China
[4] Liaoning Key Laboratory of Manufacturing System and Logistics Optimization, Shenyang 110819, China
[5] Huawei Technologies Company Limited, Beijing 100080, China
* Correspondence: guoqingxin@ise.neu.edu.cn

Abstract: To reduce logistics scheduling costs and energy consumption, this paper studies the slab allocation and hot-rolling scheduling integrated optimization problem that arises in practical iron and steel enterprises. In this problem, slabs are first allocated to orders and then sent to heating furnaces for heating; then, they are sent to a hot-rolling mill for rolling. A 0–1 integer programming model is established to minimize the attribute difference in the allocation cost between slabs and orders, the switching cost of hot-rolling processing, and waiting times after slabs reach rolling mills. Given the problem's characteristics, an improved differential evolution algorithm using a real-number coding method is designed to solve it. Three different heuristic algorithms are proposed to improve the quality of solutions in the initial population. Multiple parent individuals participate in the mutation operation, which increases the population diversity and prevents the algorithm from falling into the local optimum prematurely. Experiments on 14 sets of real production data from a large domestic iron and steel plant show that our improved differential evolution algorithm generates significantly better solutions in a reasonable amount of time compared with CPLEX, the simulated artificial method, and the classical differential evolution algorithm, and it can be used by practitioners.

Keywords: slab allocation; hot-rolling scheduling; integrated optimization problem; differential evolution algorithm; practical problem

MSC: 90-10; 90C90

Citation: Song, L.; Meng, Y.; Guo, Q.; Gong, X. Improved Differential Evolution Algorithm for Slab Allocation and Hot-Rolling Scheduling Integration Problem. *Mathematics* **2023**, *11*, 2050. https://doi.org/10.3390/math11092050

Academic Editor: Frank Werner

Received: 25 March 2023
Revised: 19 April 2023
Accepted: 23 April 2023
Published: 26 April 2023

1. Introduction

The iron and steel industry plays an important role in economic development. With the development of science and technology, competition between iron and steel enterprises is becoming increasingly fierce [1]. The production of iron and steel consists of several steps, including raw materials acquisition, ironmaking, steelmaking [2], continuous casting, hot rolling, cold rolling, electroplating, and so on, and each process is closely connected. Slabs are an important intermediate product in steel production, and hot rolling is a crucial intermediate process. The relationship between the slab allocation problem and the hot-rolling scheduling problem is shown in Figure 1. Establishing reasonable slab allocation schemes and hot-rolling schedules has many benefits, for example, ensuring the continuity of the production process, improving product quality, increasing work efficiency, reducing energy consumption, and saving enterprise costs.

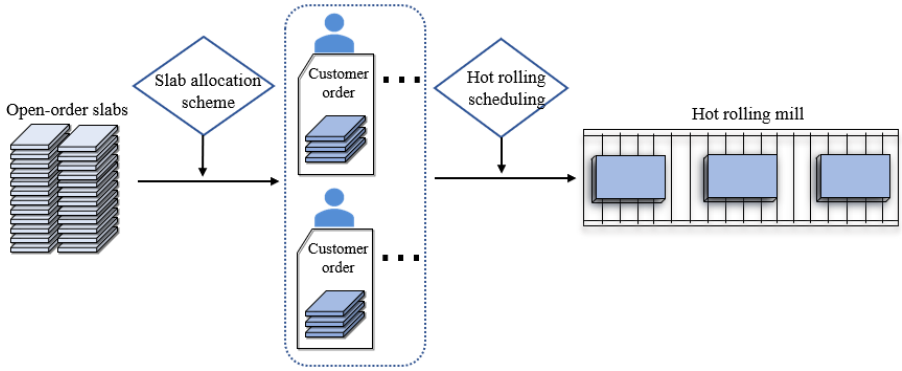


Figure 1. The slab allocation problem and the hot-rolling scheduling problem.

Given the complexity of the steel production process, a wide variety of specifications for intermediate products, and frequent fluctuations between actual products and customer orders, a large number of open-order slabs are often produced. For example, the produced slab might not meet the requirements of the customer order or subsequent units; the customer might request a temporary cancellation or ask for changes to the order; there might be manual disengagement in the order caused by unreasonable slab allocation; and manual operation errors might occur. Open-order slab backlogs take up a large amount of space in inventories and can be expensive; thus, they need to be dealt with in a timely fashion. The slab allocation problem refers to matching slabs and orders based on a many-to-one relationship between a slab's specification attributes and an order's requirements, thus improving and optimizing the unreasonable allocation relationship at the same time.

Only the slab is allocated to a deterministic order; we can know how thick the slab will be rolled, so only the allocated slab can be included in the rolling schedule. Furthermore, only the slab programmed into the hot-rolling schedule can be rolled. The problem of hot-rolling production scheduling refers to the reasonable formulation of a slab rolling sequence within a certain period to meet the relevant rolling schedule and rescheduling optimization due to abnormal situations, such as rolling mill failures. Specifically, this is based on the delivery time and priority level of multiple customer orders (slabs in the pre-selection pool) and multiple rolling units; i.e., slab processing sequences are compiled at one time. Each rolling unit consists of many slabs. In the same rolling unit, the jumping values of the adjacent slab's width, thickness, and hardness must meet rolling specification requirements; otherwise, the loss of rolling equipment in the steel enterprise will increase. Many rolling units are arranged in a regular order to form a rolling schedule. Different rolling units correspond to different rolls, so it is necessary to stop the machine and replace the rolls after producing one rolling unit. Because of the high start-up cost of changing rolls, the replacement of rolls should be reduced as much as possible to ensure production quality. In addition, the quality of hot-rolling scheduling directly affects the quality of materials needed by downstream units.

In the hot-rolling process, a slab needs to be heated in the heating furnace first and then rolled after it reaches a certain temperature. If the slab has a high temperature before entering the heating furnace, this process is called 'hot delivery and hot charging'. Increasing the ratio of 'hot delivery and hot charging' is conducive to energy savings and consumption reduction. However, even if a slab has been allocated for an order, if it has not been included in the hot-rolling schedule, it usually is left to wait and cannot be rolled in time. In addition, if the slab allocation scheme is not coordinated with the rolling schedule, the number of slab 'stacking' [3] incidents will increase, thus increasing the extra logistics scheduling costs. At present, planners only consider slab-rolling schedule constraints when compiling hot-rolling schedules. However, slab allocation and slab hot-rolling scheduling are two interrelated and complementary processes. Considering the two processes individually will make the connection between processes unsmooth. Therefore, it is necessary to integrate the slab allocation process with the hot-rolling scheduling

process. In this way, the ratio of 'hot delivery and hot charging' and the material utilization rate can not only be significantly improved but the logistics production cost can also be greatly reduced.

The slab allocation and hot-rolling scheduling integrated optimization problem researched in this paper can be summarized as follows: based on the slab rolling orders of iron and steel enterprises, the allocation relationship between slabs and orders can be optimized to satisfy the constraints of slab allocation and rolling process specifications; thus, the slab rolling schedules of hot-rolling mills can be formulated at the same time. Transforming and integrating these two relatively isolated material arrangement schemes into a unified and dynamic decision-making whole can accurately balance logistics, thus providing effective support to sales departments in ordering work and finally reasonably expanding the production capacity of the whole rolling unit and balancing resource distribution.

Slab allocation and hot-rolling scheduling integrated optimization is not only of great significance but is also challenging [4]. Firstly, smooth production of the hot-rolling mill mainly depends on the exact scheduling sequence of slabs, and the hot-rolling mill is operated according to the sequence of one order followed by another, with each order containing many pieces of slabs. Therefore, in addition to coordinating the orders' rolling sequence and the slabs' rolling sequence within the rolling units, it is also necessary to formulate the allocation relationship between the slabs and orders. Any mistake in the middle will affect the overall production and product quality, which will impact the whole process. Secondly, steel production is a complicated process. Faults in production equipment, fluctuations in the composition of raw materials, changes in production operation conditions, errors in the control process, approximations in mechanism models, etc., can lead to deviations in the quality and specifications between the produced slab and the original design. Thus, it is necessary to adjust the allocation relationship between the slab and the order. Thirdly, there is a lot of equipment involved in hot-rolling production lines, and hot-rolling schedules are long and complex. This means that we not only need to consider equipment maintenance and transportation costs but also account for the connection between steelmaking and cold rolling. Therefore, it is quite challenging to work out reasonable slab allocations and hot-rolling production schedules simultaneously. Thus, this is also a technical problem that needs to be urgently solved in the field of iron and steel production; that is, we need to synchronize the data and information flow regarding slab allocation and hot-rolling scheduling to achieve real, integrated production management.

The slab allocation problem and the hot-rolling scheduling problem in iron and steel enterprises have a large amount of data and complex constraints. When considering all aspects of process-planning constraints, we should pursue several goals, such as minimizing differences between slab allocations and orders; minimizing production transition costs caused by specification changes in slabs; minimizing slab waiting times; and minimizing the processing and switching costs of adjacent orders. At present, this problem is generally solved through manual operation by planners, that is, by first manually allocating and then rolling based on a manually arranged hot-rolling schedule. However, it is difficult to achieve global optimization via manual allocation and hot-rolling schedules, which leads to poor production smoothness and hinders production efficiency improvements. More advanced enterprises will use ERP (enterprise resource planning) systems to achieve better information management. However, a common problem with these ERP systems is that they are mainly used to sort and screen existing data, and they lack the ability to change parameters dynamically according to production requirements or provide high-quality allocation schemes and rolling schedules automatically. Therefore, intelligent optimization methods are needed to achieve efficient global optimization.

The differential evolution algorithm is an intelligent optimization algorithm first proposed by Kenneth Price and Rainer Storn in 1995 [5]. It ensures the survival of the fittest among individuals by simulating the natural selection laws of competition and cooperation in the evolutionary process [6] of biological populations [7]. It has been used to solve many practical problems and has good global search abilities and robustness. The

slab allocation and hot-rolling scheduling integrated optimization problem is complex, large-scale, and has many constraints. While the differential evolution algorithm has fewer parameters, it can effectively reduce the complexity of algorithms and is suitable for solving optimization problems in large-scale and complex environments. There are many-to-one correspondences between slabs and orders and between slabs and rolling units, which shows that this problem requires a complex algorithm. The differential evolution algorithm has flexible coding modes, is easy to implement, and has strong operability and feasibility. Therefore, the differential evolution algorithm is suitable for solving this problem.

To sum up, research on the integrated optimization of slab allocations and hot-rolling scheduling has the following benefits:

(1) By integrating slab allocation with the hot-rolling scheduling process, the ratio of 'hot charging and hot delivery' can be improved, energy consumption and logistics costs can be reduced, and enterprise profit margins can be improved.
(2) By considering hot-rolling scheduling in the process of slab allocation, the allocation and hot-rolling production costs can be comprehensively coordinated, and the hot-rolling switching costs caused by unreasonable feeding can be reduced.
(3) This can quickly generate high-quality, large-scale slab allocation schemes and effective hot-rolling schedules, thus reducing inventory costs and workers' work intensity and improving resource allocation, work efficiency, product quality, timely delivery, production management levels, and enterprise benefits.
(4) The differential evolution algorithm can solve the slab allocation and hot-rolling scheduling integrated optimization problem, which not only provides a theoretical reference for solving similar large-scale practical problems but also expands the differential evolution algorithm's application field.

The rest of this paper is organized as follows. Section 2 summarizes the related literature. The integrated optimization problem is described, and the model is constructed in Section 3. Our proposed improved differential evolution algorithm is introduced in Section 4. Section 5 shows the experimental results and analytics for the practical problem. Finally, Section 6 elaborates on the conclusion and discusses future work.

2. Literature Review

Our literature review on the problem of integrating the slab allocation problem and the hot-rolling scheduling problem will focus on the following two aspects: the related problems and the related methods.

2.1. Research on Related Problems

2.1.1. Slab Allocation Problem

The relevant research on the slab allocation problem is limited and outdated. Based on the different types of slabs and orders, slab allocation problems can be divided into three categories: open-order slab allocation problems [8], slab re-allocation problems [9], and self-drawn order-grouping problems [10]. Among them, the open-order slab allocation problem refers to allocating surplus slabs that have not been allocated to orders to customer orders to improve the utilization of slabs. Lv et al. [11] solved a stochastic surplus open-order slab allocation problem using a scenario-based modeling approach and the scatter search algorithm. The slab re-allocation problem is also a full-order slab allocation problem; this involves reallocating a full-order slab that has been allocated to one order to a new customer order to further optimize and adjust the relationship between the full-order slab and customer order, shortening the completion time of the customer orders and improving the level of customer satisfaction. Tang et al. [9] transformed an integer nonlinear programming model into a mixed-integer linear programming (MILP) model and proposed a heuristic algorithm based on a tabu search to obtain satisfactory solutions efficiently for the slab reallocation problem in the steel industry. The self-drawn order-grouping problem refers to a virtual order created by iron and steel enterprises when an inventory is overstocked; that is, open-order slabs with overlapping upper and lower rolling width limits are divided

into groups to quickly 'digest' the inventory. Lv and Wang [10] adopted a method based on data analysis to study whether it is better to group a self-drawn order immediately or wait for a potential customer order to arrive when there is no suitable customer order for the surplus slab.

2.1.2. Hot-Rolling Scheduling Problem

Compared with the slab allocation problem, the relevant research on the hot-rolling scheduling problem is relatively rich and diversified. However, most research on the hot-rolling scheduling problem defaults to the conclusion that the allocation relationship between the slab and order has already been determined. In essence, the hot-rolling scheduling problem can be defined as a multiple traveling salesman problem [12] or a vehicle routing problem [13]. Tang et al. [12] proposed a multiple traveling salesman problem model for hot-rolling scheduling in iron and steel complexes and solved it using a modified genetic algorithm. Tang and Wang [13] presented the prize-collecting vehicle routing problem (PCVRP), a variant of the classical vehicle routing problem, which is derived from hot-rolling production in the iron and steel industry. One major characteristic of the PCVRP is that the customers need not be visited compulsorily; instead, a prize can be collected from each customer when visiting them. In addition to a capacity constraint, the model also introduced a task completion constraint, which requires the total demand of visited customers to be no less than a predetermined amount. Li and Tian [14] also investigated a PCVRP extracted from an actual hot-rolling production process. They developed a two-level self-adaptive variable neighborhood search algorithm based on two decision levels in the PCVRP, namely, the selection of customers to visit and vehicle scheduling by selected customers. To obtain the lower bound of the PCVRP, the mixed-integer programming model of the proposed PCVRP was transformed into an equivalent traveling salesman problem model by using the graph expansion method. Özgür et al. [15] reviewed the planning and scheduling methods of hot-rolling mills in iron and steel production around the following aspects: constraints included in the analysis, objective functions (or optimization criteria) employed, the optimization algorithm used, and the experimental data included.

2.1.3. Integration Optimization Problem

As can be seen from the above literature review, the biggest gap of the integration optimization problem lies in the fact that, in the previous literature, the slab allocation problem and the hot-rolling scheduling problem are generally researched separately. To make up for this gap, this paper integrates the two sub-problems, so as to achieve the global optimal decision from a broader perspective. Fortunately, the research on integration optimization problem from other related industries can provide us with some context. A knowledge network system was developed by Yang et al. [16] based on the needs of the modern iron and steel enterprise integrated production management. They proposed a model knowledge representation and intelligent matching mechanism. Li et al. [17] researched the integrated problem of soaking pit heating and hot rolling scheduling in steel plants. They provided a fast heuristic algorithm with worst-case bound to generate near-optimal solutions, and a branch and bound algorithm to solve the problem to optimality. Tan et al. [18] designed a hybrid algorithm that combines mixed-integer programming and constraint programming to solve a scheduling problem in a steel plant, referring to continuous casting, reheating furnace, and hot-rolling processes.

2.2. Research on Related Methods

2.2.1. Heuristic Algorithm

Regarding the methods for solving hot-rolling scheduling problems in previous studies, the heuristic algorithm is clearly the first choice. Pan et al. [19] proposed a mathematical model and two-stage heuristic for hot-rolling scheduling in compact strip production. Puttkammer et al. [20] used a greedy randomized adaptive search procedure heuristic for

the hot strip mill scheduling problem under consideration of energy consumption. The heuristic algorithm is generally simple, short in solving time, and strong in stability, but its shortcomings are also obvious. On the one hand, the solution obtained by the heuristic algorithm can only guarantee feasibility, not optimality. On the other hand, the heuristic algorithm is generally designed according to the characteristics of a specific problem and has no generalization. As the intelligent optimization algorithm can make up for these disadvantages, the heuristic algorithm is also often combined with intelligent optimization algorithm in the literature. Hu et al. [21] developed a heuristic method combining an improved ant colony extended algorithm with local search procedures dedicated to the hot strip mill production scheduling problem. To research a multi-objective hot-rolling scheduling problem in the compact strip production, Pan et al. [22] adopted an improved hot-rolling scheduling heuristic to solve the sheet-strip assignment problem and a multi-objective evolutionary algorithm to find the Pareto optimal or near-optimal solutions for the sheet-strip sequencing problem.

2.2.2. Differential Evolution Algorithm

Compared with the heuristic algorithm, the differential evolution algorithm has stronger global search ability and more flexible application. Generally speaking, improving a differential evolution algorithm [23] involves improving the control parameters, the mutation strategy [24], the crossover strategy, the selection strategy, and gene population recombination. Some scholars have used differential evolution algorithms to study problems related to scheduling. Li and Fang [25] proposed a differential evolution algorithm based on an evolutionary direction to solve the robust multi-objective optimization problem of rolling schedules in tandem with cold rolling. Their algorithm calculated the horizontal angle of the vector, which was used to choose the mutation vector. The chosen vector contained the convergence direction, which changed the random mutation operation in the differential evolution algorithm. Chakraborti and Kumar [26] addressed the problem of minimizing the hot-rolling time for an ingot from a given initial thickness to a prescribed final one, subject to a number of system constraints, by using a multi-population genetic algorithm and a differential evolution algorithm. To solve the problem of dynamic scheduling in steelmaking–continuous casting production, Tang et al. [27] proposed an improved differential evolution algorithm with a real-coded matrix representation for each individual in the population, a two-step method for generating the initial population, and a new mutation strategy. Zhao et al. [28] proposed a hybrid differential evolution algorithm to estimate the distribution of that algorithm based on a neighborhood search for a job shop scheduling problem. To strengthen the searchability of their algorithm, a chaotic strategy was introduced to update the parameters of the DE. Two mutation operators were used. A neighborhood search algorithm based on blocks on a critical path was adopted to further improve the solution quality.

3. Problem Description and Model Establishment

3.1. Problem Description

The allocation difference cost between a slab and an order has five aspects, namely, steel grade, weight, width, length, and deformation. Based on the premise of satisfying allocation specification constraints, order weight constraints, and allocation mode constraints, solving the slab allocation problem involves maximizing the number of open-order slabs allocated to an order, minimizing the quality and specification differences between the slab and order, satisfying the priority requirements of the slab and order to the greatest extent, and improving the integrity of the order as much as possible. When the order processing sequence is found, the outbound sequence of the slabs can be obtained based on the allocation relationship between the slab and order.

A hot-rolling schedule consists of several rolling units, as shown in Figure 2. Every time a rolling unit is rolled, a roll-changing operation must be carried out. A rolling unit is composed of many slabs, and this characterizes the rolling sequence of a group of slabs. A

complete rolling unit consists of two parts: the hot-roll material and the main body material. The slab sequence in the rolling unit has a 'double trapezoidal' structure in terms of width, which is similar to a 'tortoise shell'. A positive trapezoid is part of the hot-roll material, and its slab width changes incrementally. The main purpose of hot-roll material is to heat rolls during the initial rolling, make the roll's thermal expansion uniform, and ensure the quality of the subsequent strip steel. Generally, their quantity is small, and they are easy to make. The reverse trapezoid is part of the main body material. The slab width of the main body material is decreased to ensure the quality of hot-rolled products and operational stability. The main body material is the main component of the rolling unit, and it comes in large quantities; thus, it needs to meet many program constraints when making a schedule.

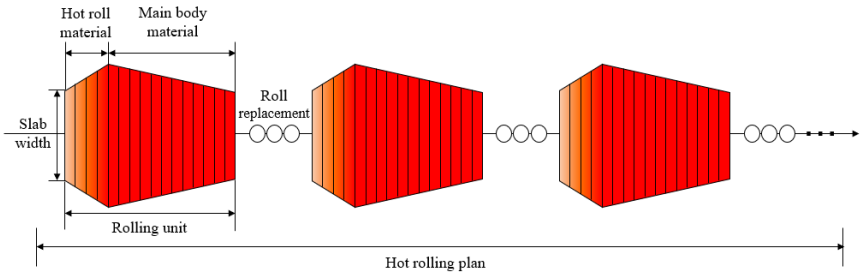


Figure 2. Structure diagram of hot-rolling scheduling.

The hot-rolling scheduling process is divided into the following steps: determining the planning quantity, establishing the preselection pool, determining the preparation schedule, and adjusting the schedule. When making the planning sequence, the hot delivery slab should be considered first to ensure the continuity of production. Secondly, hot-charging slabs should be used to shorten the slab's heating time in the heating furnace. Finally, cold slabs should be used in the warehouse, which can avoid energy waste. When making hot-rolling schedules, the following requirements should be met: (1) The hardness and thickness of the slab need to change smoothly. (2) The strip width of the main body material should be decreased in general, and a rolling length of the same width cannot exceed a certain upper limit. (3) To reduce the number of roll changes, the total rolling length of the whole rolling unit should be less than the maximum allowable rolling length of the roll and should not be less than the minimum rolling length as much as possible.

Based on the above analysis, the integration problem of slab allocation and hot-rolling scheduling studied in this paper can be described as follows: given a slab set, an order set, a rolling unit set, and a position set in the rolling unit; under the conditions of satisfying the allocation mode constraint, the order excess constraint, and the rolling scheduling constraint; optimizing the allocation relationship between the slab and order; and making the best slab-rolling schedule, we can minimize the allocation costs of attribute differences between the slab and order, the switching cost of adjacent slab processing, the waiting time after the slab arrives at the rolling unit, and the switching cost of adjacent order processing.

3.2. Model Establishment

3.2.1. Parameters

i, h index of slabs participating in the allocation;
j, s index of orders participating in the allocation;
l, l_1, l_2 index of rolling units;
k index of positions in the rolling unit;
I set of slabs participating in the allocation;
J set of orders participating in the allocation;
L set of rolling units;
K set of positions in the rolling unit;
c_{ij} allocation difference cost for allocating slab i to order j;
d_{ih} switching cost of slab i produced immediately before slab h;

f_{js} switching cost of order j produced immediately before order s;
F_1 weight coefficient of the allocation specification difference penalty in the objective function;
F_2 weight coefficient of the switching cost of the adjacent slab in the objective function;
F_3 weight coefficient of the slab waiting time in the objective function;
F_4 weight coefficient of the switching cost of the adjacent order in the objective function;
p_i treatment time of slab i;
r_i arrival time of slab i;
D_j the demand of order j;
τ_j the delivery date of order j in the hot-rolling process;
w_i weight of slab i;
T_g roll changing time;
T_0 treatment time of the hot-roll material;
M a large positive number.

3.2.2. Decision Variables

$$x_{ij} = \begin{cases} 1, \text{ if slab } i \text{ allocated to order } j \\ 0, \text{ else} \end{cases};$$

$$y_{ihl} = \begin{cases} 1, \text{ if slab } i \text{ is processed immediately before slab } h \text{ in the hot rolling unit } l \\ 0, \text{ else} \end{cases};$$

$$z_{ikl} = \begin{cases} 1, \text{ if slab } i \text{ is allocated to the } k \text{ th position of hot rolling unit } l \\ 0, \text{ else} \end{cases};$$

$$u_{jsl} = \begin{cases} 1, \text{ if order } j \text{ is processed immediately before order } s \text{ in the hot rolling unit } l \\ 0, \text{ else} \end{cases};$$

$$\gamma_{l_1 l_2} = \begin{cases} 1, \text{ if hot rolling unit } l_1 \text{ is processed immediately after unit } l_2 \\ 0, \text{else} \end{cases};$$

α_i starting processing time of slab i;
β_i completion time of slab i;
δ_l starting processing time of hot-rolling unit l;
ε_l completion time of hot-rolling unit l;

3.2.3. Model

An integer programming (IP) model can be formulated as follows.
Minimize

$$F_1 \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + F_2 \sum_{i \in I} \sum_{h \in I} \sum_{l \in L} d_{ih} y_{ihl} + F_3 \sum_{i \in I} (\alpha_i - r_i) + F_4 \sum_{j \in J} \sum_{s \in J} \sum_{l \in L} f_{js} u_{jsl} \quad (1)$$

Subject to

$$\sum_{j \in J} x_{ij} \leq 1, \ \forall i \in I \quad (2)$$

$$\sum_{i \in I} w_i x_{ij} \leq D_j + w_h + M(1 - x_{hj}), \ \forall j \in J, h \in I \quad (3)$$

$$\sum_{i \in I} z_{ikl} \leq 1, \forall l \in L, k \in K \quad (4)$$

$$\sum_{l \in L} \sum_{k \in K} z_{ikl} \leq 1, \ \forall i \in I \quad (5)$$

$$z_{ikl} + z_{h,k+1,l} - 1 \leq y_{ihl}, \ \forall i, h \in I, l \in L, k \in K \quad (6)$$

$$z_{h,k+1,l} \leq z_{ikl}, \ \forall i, h \in I, l \in L, k \in K \quad (7)$$

$$\sum_{l\in L}\sum_{k\in K} z_{ikl} \le \sum_{j\in J} x_{ij},\ \forall i \in I \tag{8}$$

$$x_{ij} + x_{hs} + y_{ihl} - 2 \le u_{jsl},\ \forall i,h \in I, j,s \in J, l \in L \tag{9}$$

$$\delta_{l_1} \le \varepsilon_{l_2} + T_g + (1 - \gamma_{l_1 l_2})M,\ \forall l_1, l_2 \in L \tag{10}$$

$$\delta_{l_1} \ge \varepsilon_{l_2} + T_g + (\gamma_{l_1 l_2} - 1)M,\ \forall l_1, l_2 \in L \tag{11}$$

$$\sum_{i\in I} r_i z_{i0l} \le \delta_l + T_0, \forall l \in L \tag{12}$$

$$\delta_l + T_0 \le \alpha_i + (1 - z_{i0l})M,\ \forall l \in L \tag{13}$$

$$\delta_l + T_0 \ge \alpha_i + (z_{i0l} - 1)M,\ \forall l \in L \tag{14}$$

$$\beta_i \le \alpha_h + (1 - y_{ihl})M,\ \forall i,h \in I, l \in L \tag{15}$$

$$\beta_i \ge \alpha_h + (y_{ihl} - 1)M,\ \forall i,h \in I, l \in L \tag{16}$$

$$\beta_i \ge \alpha_i + p_i,\ \forall i \in I \tag{17}$$

$$\beta_i \le \alpha_i + p_i,\ \forall i \in I \tag{18}$$

$$\alpha_i \ge r_i,\ \forall i \in I \tag{19}$$

$$\beta_i + (x_{ij} - 1)M \le \tau_j,\ \forall i \in I, j \in J \tag{20}$$

$$x_{ij} \in \{0,1\},\ \forall i \in I, j \in J \tag{21}$$

$$y_{ihl} \in \{0,1\},\ \forall i,h \in I, l \in L \tag{22}$$

$$z_{ikl} \in \{0,1\},\ \forall i \in I, l \in L, k \in K \tag{23}$$

$$u_{jsl} \in \{0,1\},\ \forall j,s \in J, l \in L \tag{24}$$

$$\gamma_{ij} \in \{0,1\},\ \forall i,j \in L \tag{25}$$

$$\alpha_i, \beta_i \ge 0,\ \forall i \in I \tag{26}$$

$$\delta_l, \varepsilon_l \ge 0,\ \forall l \in L \tag{27}$$

In the model, Formula (1) is the objective function of the slab allocation and hot-rolling scheduling integration problem, where the first term is the attribute difference allocation cost between the slab and order, the second term is the switching cost of adjacent slab processing, the third term is the waiting time after the slab reaches the rolling mill, and the fourth term is the switching cost of adjacent order processing.

Constraints (2)–(27) are the constraints of the slab allocation and slab hot-rolling scheduling integrated problem model, in which Constraint (2) represents the allocation pattern constraint; that is, each slab can only be allocated to one order. Constraint (3) is the order excess constraint; that is, if the order is excessive after allocation, the excess weight cannot be greater than the weight of the minimum slab allocated to the order. Constraint (4) ensures that only one slab can be processed at most at each position of each rolling unit. Constraint (5) ensures that each slab can only be processed at one position of a rolling unit at most. Constraint (6) defines the relationship between the tight front and tight back during slab processing on the rolling unit. Constraint (7) ensures that if a slab is arranged at a certain position on a rolling unit, there must be another slab arranged immediately before it. Constraint (8) ensures that only after the slab is allocated to a certain order can it be discharged into the schedule. Constraint (9) defines the processing relationship before and after the order. Constraints (10)–(11) define the time when the rolling unit starts processing. Constraint (12) requires that the rolling unit's starting time is later than the first slab's arrival time on the rolling unit. Constraints (13)–(14) define the starting time of the first slab on each rolling unit. Constraints (15)–(18) define the completion time of slab processing. Constraint (19) ensures that the slab can be processed only after it arrives. Constraint (20) requires that all slabs allocated to an order be rolled before the delivery date of the order in the hot-rolling process. Constraints (21)–(27) define variable ranges.

From the objective function of Formula (1), it can be concluded that the model in this paper is a 0–1-integer programming model. This kind of model is an NP-hard problem [29], and the optimal solution to the problem cannot be obtained in polynomial time. CPLEX is generally only suitable for solving small- and medium-sized practical problems, and it requires much solving time, which is insufficient in meeting the needs of actual production. Based on this, this paper proposes an improved differential evolution algorithm that can solve large-scale practical problems quickly.

4. Algorithm

Given the characteristics of the slab allocation and hot-rolling scheduling integration problem, a differential evolution algorithm [30] can be used to solve it. Classical differential evolution algorithms include several steps, such as population initialization, mutation, crossover, and selection. In this paper, the differential evolution algorithm is improved in the following aspects. The initial population is generated by heuristic and random methods. Two mutation modes are combined to randomly produce mutant individuals. The operation of repairing infeasible solutions is added by finding remainders. Poor individuals in the gene pool are updated. The pseudocode of the improved differential evolution algorithm is provided in Algorithm 1.

Algorithm 1: Improved Differential Evolution Algorithm

INPUT: information on slabs and orders; iteration index, g; max iteration index, G_{max}.

```
Generate a random initial population;
Perform three different simulated manual heuristic algorithms according to Section 4.2 to
obtain three feasible solutions;
Replace three individuals in the initial population and set g = 0;
WHILE g < G_max
  FOR each individual in the population
    Perform a mutation operation according to Equation (28);
    Perform a crossover operation according to Equation (29);
    Perform a reparation operation according to Section 4.5;
    Perform a selection operation according to Equation (30);
  END FOR
  g = g + 1;
END WHILE
OUTPUT the final solution.
```

4.1. Coding and Decoding Strategy

In this paper, given the characteristics of the slab allocation and hot-rolling scheduling integration problem, an efficient encoding and decoding strategy based on real numbers is proposed; it can express slab allocation schemes and rolling schedules at the same time. Each algorithm's solution represents an individual and corresponds to a relationship between the slab, order, or rolling unit. The maximum number of codes is expressed as the number of slabs participating in the allocation. Assuming that the number of slabs participating in the allocation is N, the coding bits are $\{0, 1, 2, \ldots, N-1\}$. Assuming that the number of orders participating in the allocation is B, the value range of the integer part of each coding value (gene) is $\{-1, 0, 1, 2, \ldots, B-1\}$. When the value of the integer part of the gene is -1, it means that the slab is not allocated to any order. At this point, the value of the decimal part is zero, which means that only the slab allocated to one particular order can participate in the rolling schedule. When the integer part of the gene is $\{0, 1, 2, \ldots, B-1\}$, this means that the slab is allocated to the corresponding order. If the number of rolling units is E, the value range of the decimal part is $\{0, 1, 2, \ldots, E-1\}$. The algorithm's encoding method is illustrated with a concrete example, as shown in Table 1.

Table 1. The coding mode of improved differential evolution algorithm.

Coding bit	0	1	2	3	4
Coding value	3.2	2.6	4.5	−1.0	0.8

The code in Table 1 has five bits, which means that there are five slabs participating in the allocation; that is, $N = 5$. The dimension of each individual in the differential evolution algorithm is five. The value encoded by each bit in the code represents the value of each individual gene. For example, the coding value of coding bit 1 is 2.6, which means that slab No. 1 is allocated to order No. 2 and assigned to rolling unit No. 6. The coding value of coding bit 3 is −1.0, which means that slab No. 3 is not allocated to any order, so it cannot be rolled. In the same way, slabs No. 0, No. 2, and No. 4 are allocated to orders 3, 4, and 0, respectively, and distributed to rolling units No. 2, No. 5, and No. 8, respectively. The advantage of this coding method is that it is consistent with the actual structural features of the problem. Because the values of the coding bits of the integer and decimal parts are one case at most, in the subsequent operation, it can be guaranteed that each slab can only be allocated to one order at most; this also ensures that the slab can only be assigned to one position on a rolling unit at most.

4.2. Population Initialization

The population size is set to *NP*; that is, the initial population contains *NP* individuals. This paper uses two methods to construct the initial population: one is to construct different heuristic algorithms, and the other is to adopt the stochastic strategy. These two methods have different advantages and disadvantages. Based on the actual characteristics of the problem, and combined with the actual production process, heuristic algorithms significantly improve the quality of solutions in the initial population, but the number and the diversity of solutions are limited. Initial solutions generated by a random method can obtain various characteristics, and the number of solutions can be freely scaled according to the scale of the problem, but the quality of the solutions cannot be guaranteed. In this paper, three heuristic algorithms are used to generate the initial solutions: according to the order demand, according to the slab width, and according to the slab thickness. The three heuristic algorithms are briefly introduced below.

Heuristic 1 is based on the order demand. The order demand is sorted from large to small, and the orders are assigned to the corresponding rolling units according to their rolling capacity. Then, the slabs are assigned to the corresponding positions of the rolling units according to the arrival time of the slabs, the processing time of the slabs, and the starting time of the rolling units.

Heuristic 2 is based on slab width. The slab is classified according to its width. As much as possible, slabs with the same or similar widths should be classified into one class. Then, according to the weight of the slabs, the slabs are divided into corresponding orders. Next, slabs with the same attributes are divided into one rolling unit according to the rolling capacity of the rolling units as much as possible. Finally, the rolling schedule is arranged according to the arrival time of the slabs.

Heuristic 3 is based on slab thickness. The slab is classified according to its thickness. As much as possible, slabs with the same or similar thickness should be classified into one class. Then, according to the weight of the slabs, the slabs are divided into corresponding orders. Next, slabs with the same attributes are divided into one rolling unit according to the rolling capacity of the rolling units as much as possible. Finally, the rolling schedule is arranged according to the arrival time of the slabs.

4.3. Mutation Strategy

In a differential evolution algorithm, mutation is an individual-based operation. To reduce the probability of the algorithm falling into a local optimum, based on the mutation operation of the basic differential evolution algorithm, we propose that multiple parent individuals participate in the mutation simultaneously so that some mutant individuals can change greatly. This not only increases the population diversity but also improves the algorithm's optimization ability. The mutation formula is shown in Equation (28).

$$v_i(g+1) = \begin{cases} x_{r1}(g) + F \cdot [x_{r2}(g) - x_{r3}(g) + x_{r4}(g) - x_{r5}(g)] & \text{if } rand(0,1) \le R \\ x_{r1}(g) + F \cdot [x_{r2}(g) - x_{r3}(g)] & \text{else} \end{cases} \quad i \ne r_1 \ne r_2 \ne r_3 \ne r_4 \ne r_5 \tag{28}$$

where $x_i(g)$ represents the i-th individual in the g-th generation population. r_1, r_2, r_3, r_4, and r_5 represent five different individuals randomly selected from the population as parent individuals to participate in the mutation operation. F is the scaling factor, which is used to control the step size of the differential vector. $rand(0,1)$ represents a random real number between 0 and 1. R is a predetermined parameter value. $v_i(g+1)$ represents the mutated individual, which automatically enters the next generation and becomes the i-th parent individual of the $g+1$-th generation population.

4.4. Crossover Operation

In a differential evolution algorithm, crossover is an operation based on each gene of an individual. The genes of the crossover individual are randomly determined by the genes of the parent individual and mutant individual together. The specific operation mode is shown in Equation (29).

$$u_{i,j}(g+1) = \begin{cases} v_{i,j}(g+1) & \text{if } rand(0,1) \le CR, \text{or } j = rand(i) \\ x_{i,j}(g) & \text{else} \end{cases} \tag{29}$$

where $x_{i,j}(g)$ represents the j-th gene of the i-th individual in the g-th generation population. $v_{i,j}(g+1)$ represents the j-th gene of the i-th mutant individual in the $g+1$-th generation population. $u_{i,j}(g+1)$ represents the j-th gene of the i-th crossover individual in the $g+1$-th generation population. CR stands for crossover probability. The smaller the CR, the greater the contribution of the parent individual to the crossover individual, the faster the convergence speed, and the stronger the exploitation ability of the algorithm in a small range. In contrast, if the CR is larger, the mutation individual contributes more to the crossover individual, which is not conducive to accelerating the convergence speed of the algorithm, but it does enhance the exploration ability of the algorithm. $rand(i)$ represents a random integer between $[1, N]$, and $j = rand(i)$ ensures that at least one coding bit in the crossover individual inherits something from the mutant individual; otherwise, there is a small probability that the crossover individual will be exactly the same as the parent

individual. Because the solutions obtained by the crossover operation may be infeasible, it is necessary to repair the infeasible solutions before the selection operation.

4.5. Reparation Strategy

There are two kinds of infeasible solutions produced by crossover operations: one where the slab weight and the order demand violate the allocating condition, and one where the number of rolling units or rolling positions exceeds the boundary. The specific reparation strategy steps are as follows.

Step 1: Remove the slab that does not meet the allocating requirements from the allocated order.

Step 2: Sort according to the remaining order demand, from large to small, and allocate the open-order slabs to the order in turn.

Step 3: Judge whether the current slab is the last open-order slab. If so, the algorithm ends. Otherwise, go to Step 2.

Step 4: Sequentially perform the remainder operation for the integer bits and decimal bits of the out-of-bounds code until it meets the requirements.

4.6. Selection Operation

The selection operation is carried out according to Equation (30):

$$x_i(g+1) = \begin{cases} u_i(g+1) \text{ if } f(u_i(g+1)) \le f(x_i(g)) \\ x_i(g) \qquad \text{else} \end{cases} \tag{30}$$

where $f(x_i(g))$ represents the fitness value of the i-th individual in the g-th generation population. In the same way, $f(u_i(g+1))$ represents the fitness value of the i-th crossover individual in the $g+1$-th generation population. $x_i(g+1)$ represents the newly generated offspring, that is, the i-th individual in the $g+1$-th generation population. Equation (28) means that the best parent individual and crossover individual are selected to enter the next generation to form a next-generation population. Because a minimization model is constructed in this paper, the selection operation is carried out in the following ways: (1) If both solutions are feasible, the individual with the smaller fitness value is preferentially selected to enter the next-generation population. (2) If the infeasible solution and the feasible solution exist at the same time, the feasible solution is preferentially selected to enter the next-generation population. (3) If both solutions are infeasible, the solution with fewer constraint violations is preferentially selected to enter the next-generation population.

5. Experiment and Discussion

5.1. Instances and Platform

The experimental data are obtained by randomly selecting 14 groups of data of different sizes based on the actual production process of a large domestic iron and steel enterprise. The commercial software CPLEX 12.6 is used to solve the mathematical model of the slab allocation and hot-rolling scheduling integrated optimization problem described in Section 3.2. The improved differential evolution algorithm and other comparison algorithms are implemented using the C + + programming language. The experimental hardware configuration is an Intel (R) Core Q9550 CPU @ 2.83 GHz with 3.2 GB RAM.

5.2. Parameter Settings

The experimental parameters of the classical differential evolution algorithm and improved differential evolution algorithm are set as follows: the population size, NP, is set to 20; the mutation factor, F, is set to 0.5; the crossover factor, CR, is set to 0.5; and the algorithm's maximum iteration number, G_{max}, is set to 500 generations. The CPLEX stop condition is a memory overflow or running for more than 3600 s.

5.3. Comparison Algorithms

The comparison algorithms include CPLEX, the simulated artificial method, and a classical differential evolution algorithm. A heuristic method based on artificial experience, used by most domestic iron and steel enterprises, is adopted as the simulation's artificial method. The main steps are as follows: Firstly, the orders are sorted from large to small according to their insufficient quantities; then, the orders with large insufficient quantities are preferentially allocated without violating the constraints of the allocation relationship until all orders have no insufficient quantities. Secondly, based on the total weight of the slabs that need to be rolled and the rolling capacity of the rolling unit, the number of rolling units is roughly estimated; then, the slabs are reasonably distributed to the corresponding rolling units according to the time after the slabs arrive at the rolling unit. Thirdly, based on the time after the slabs arrive at the rolling unit, the slab-processing time, and the starting rolling time of the rolling units, the slab-rolling sequence in the rolling units is arranged in turn.

5.4. Experimental Results and Discussion

Both the CPLEX and simulated manual methods are run once to obtain results. To eliminate the influence of random errors, the means and variances in the classical differential evolution algorithm and improved differential evolution algorithm are obtained via statistics after running 20 experiments independently. In addition, the significant differences between the classical differential evolution algorithm and improved differential evolution algorithm are identified by executing a non-parametric Wilcoxon Statistical Test. In particular, the significance level is set to 0.05 (5%). Moreover, '-', '+', and '=' indicate that the improved differential evolution algorithm achieves significantly better, significantly worse, and equal performance relative to the classical differential evolution algorithm, respectively. The statistical fitness value and solving time are shown in Table 2, where '*AVG*' represents average fitness value, '*SD*' represents standard deviation of fitness value, and 'Sig.' represents significance test results. 'SA' represents the simulated artificial method, 'CDE' represents the classical differential evolution algorithm, and 'IDE' represents the improved differential evolution algorithm. 'Deviation' refers to the percentage difference between the optimal value obtained by CPLEX and the results obtained by the other three algorithms. '–' indicates that CPLEX cannot solve the instances at these scales, and these items are not compared.

Table 2. Results of CPLEX, the simulated artificial method, the classical differential evolution algorithm, and the improved differential evolution algorithm.

Scales			Fitness Value						Solving Time(s)			Deviation (%)		
Slab	Order	CPLEX	IDE			CDE		SA	CPLEX	IDE		IDE	CDE	SA
			AVG	*SD*	Sig.	*AVG*	*SD*			*AVG*	*SD*			
3	3	50.13	50.13	0.00	=	50.13	0.00	50.13	0.218	0.20	0.00	0.00	0.00	0.00
5	5	165.84	165.84	0.00	=	165.84	0.00	176.49	0.310	0.23	0.01	0.00	0.00	6.42
10	10	1085.03	1085.03	0.00	-	1096.36	13.67	1194.08	0.483	0.46	0.36	0.00	1.04	10.05
15	10	1604.62	1632.52	21.57	-	1712.26	34.66	1818.36	0.670	0.65	0.24	1.74	6.71	13.32
20	15	3571.73	3675.91	52.91	-	3846.11	69.80	4018.91	1.93	2.21	0.51	2.92	7.68	12.52
20	20	5600.60	5838.42	91.47	-	6026.53	94.52	6275.47	30.92	31.27	0.44	4.25	7.61	12.05
25	20	6200.60	6435.77	106.15	-	6605.26	107.98	6873.37	80.34	40.79	1.06	3.79	6.53	10.85
30	25	6900.50	7120.82	154.61	-	7355.52	137.16	7727.87	150.53	50.85	1.45	3.19	6.59	11.99
35	35	10,440.28	10,769.39	184.02	-	11,238.80	151.94	11,695.20	500.03	61.18	4.46	3.15	7.65	12.02
45	40	23,000.98	23,555.00	294.29	-	24,996.61	244.34	26,685.74	680.05	65.14	5.31	2.41	8.68	16.02
50	45	–	29,932.19	391.42	-	30,801.31	416.65	33,654.48	–	74.96	4.78	–	–	–
60	60	–	32,410.48	419.25	-	33,219.48	446.06	35,654.89	–	89.82	6.32	–	–	–
80	60	–	34,585.71	510.54	-	35,753.11	552.61	38,665.11	–	119.36	7.01	–	–	–
100	80	–	36,506.48	627.12	-	38,380.37	633.56	40,109.79	–	152.21	9.15	–	–	–

It can be seen from Table 2 that the proposed improved differential evolution algorithm is superior to CPLEX in solving large-scale problems, and it can find the near-optimal solution to the problems in a reasonable time. When the number of slabs and orders increases to 50 and 45, respectively, CPLEX cannot solve the problem because of memory overflow, but the improved differential evolution algorithm proposed in this paper can still obtain a high-quality solution. With the increased scale of the problem, CPLEX's computation time obviously increases until it cannot solve the problem. However, the increase in the improved differential evolution algorithm's solution time is not very large. Specifically,

(1) The deviation between the solutions obtained by the simulated artificial method and the optimal solutions obtained by CPLEX is the largest, and the average deviation is about 10.52%. The average deviation between the classical differential evolution algorithm and the optimal solution is about 5.25%. The deviation between the improved differential evolution algorithm and the optimal solution is the smallest, and the average deviation is about 2.14%. The effectiveness of the improved differential evolution algorithm can be seen.

(2) On small and medium scales, the solutions obtained by the improved differential evolution algorithm are very close to the optimal solutions obtained by CPLEX, and the overall deviation between them is less than 3%. In addition, the improved differential evolution algorithm can even obtain the optimal solution in some instances.

(3) In small-scale instances, the improved differential evolution algorithm consumes almost the same amount of time as CPLEX. In medium-scale instances, the solving time of the former is slightly less than that of the latter. In large-scale instances, the solving time of the former is obviously shorter than that of the latter. Even when the instances cannot be solved by CPLEX, the improved differential evolution algorithm can still obtain high-quality solutions in a short time.

(4) From the significant differences, it can be seen that the improved differential evolution algorithm is significantly better than the classical differential evolution algorithm in all other instances except the first two instances.

To observe and analyze the experimental results more intuitively, the experimental fitness value and time are visualized, as shown in Figures 3 and 4, respectively, using mixed-mode column and line charts.

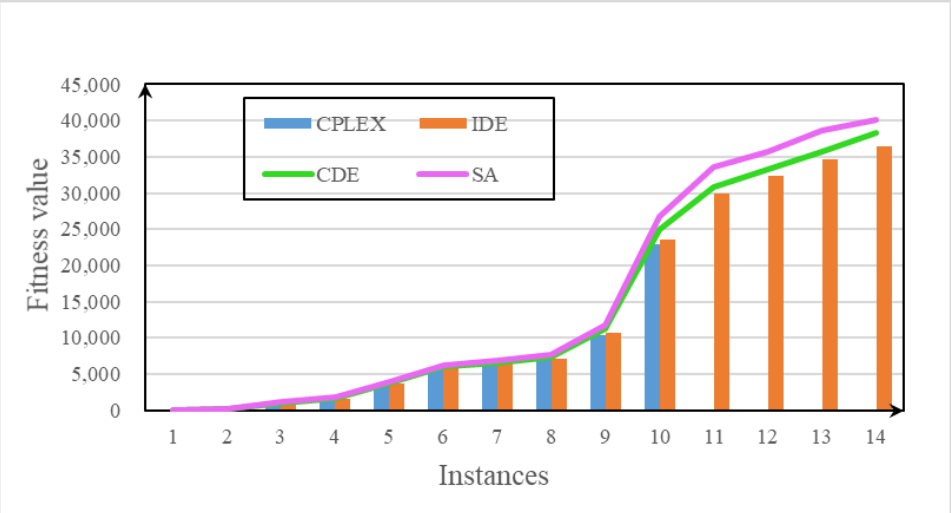


Figure 3. Column and line chart showing fitness values for the improved differential evolution algorithm and comparison algorithms.

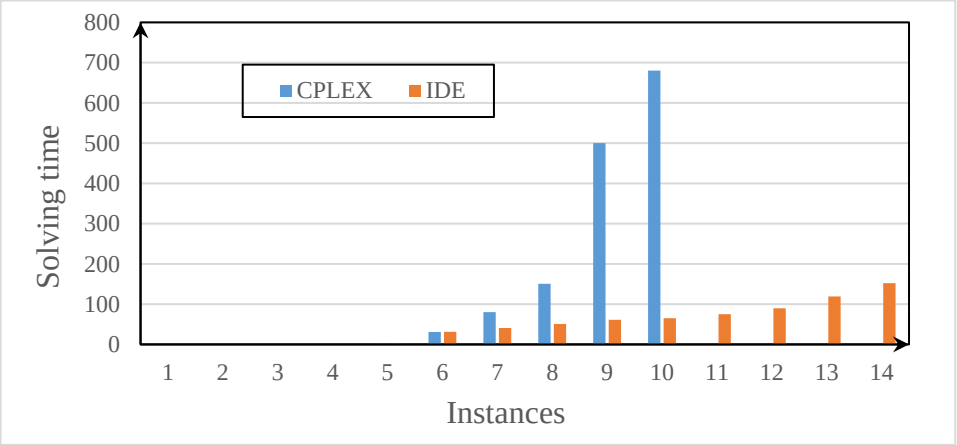


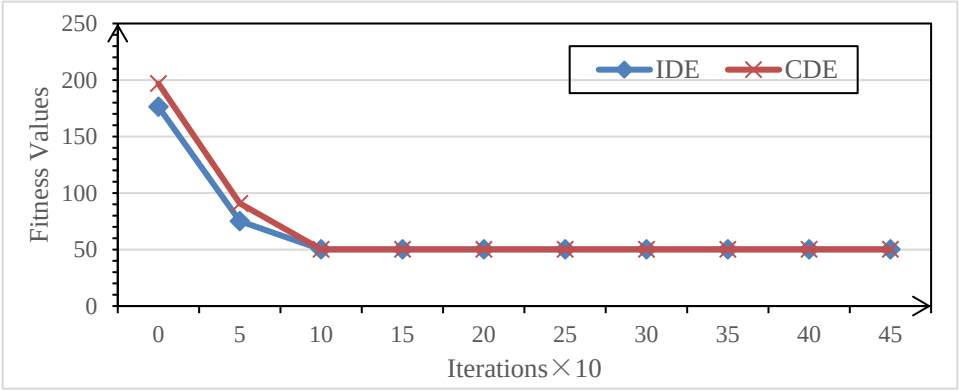
Figure 4. Column chart obtaining the times of the improved differential evolution algorithm and comparison algorithm.

In Figure 3, the abscissa represents the instance scale, and the ordinate represents the fitness value obtained by the algorithm. The blue and orange columns represent the fitness values obtained by CPLEX and the improved differential evolution algorithm, respectively. The absence of blue columns in the last four instances shows that CPLEX can only find solutions to small- and medium-scale instances; it cannot find solutions to large-scale instances. However, the improved differential evolution algorithm can find solutions to all-scale instances. Purple and green lines represent the fitness values obtained by the simulated artificial method and the classical differential evolution algorithm, respectively. The orange columns are always below the green lines, which shows the effectiveness of the heuristic initial solution and multi-mutation operator strategy in the improved differential evolution algorithm.

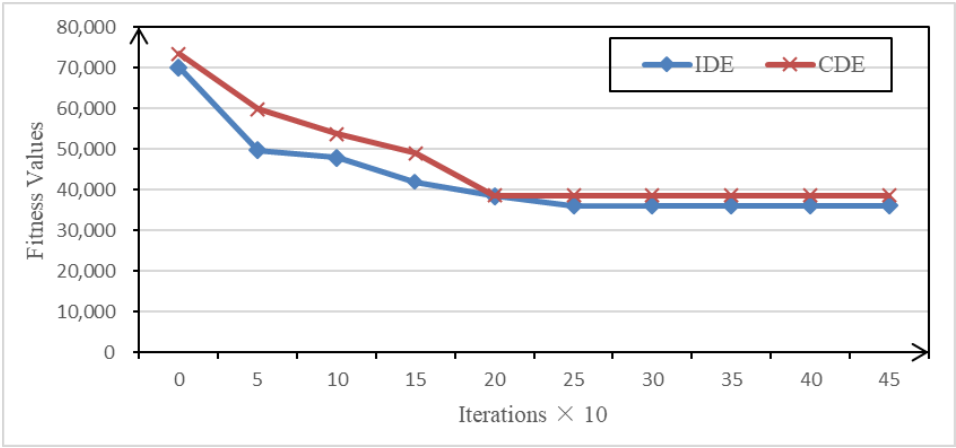
In Figure 4, the abscissa represents the problem scale, and the ordinate represents the solving time. Blue and orange columns represent the CPLEX and improved differential evolution algorithm solving times, respectively. The first five instances do not appear in the column chart because the solving time is zero and because the CPLEX and improved differential evolution algorithm solving time is very small (less than 2.3 s) when calculating the first five small-scale instances. Furthermore, the ratio is very small compared with the latter medium- and large-scale instances; it is not in an order of magnitude, so it cannot be displayed in the chart. Similarly, the absence of blue columns in the last four instances shows that these scale instances cannot be solved by CPLEX within the specified time. It can be seen that the CPLEX solving time increases rapidly with the increased problem scale, while the increased improved differential evolution algorithm solving time is relatively stable, which verifies the efficiency of the improved differential evolution algorithm.

The convergence curves of the improved differential evolution algorithm and classical differential evolution algorithm are shown in Figure 5, where the blue and red curves represent the improved differential evolution algorithm and classical differential evolution algorithm, respectively. Because of the limited space, only two instances are randomly selected.

In Figure 5a, the initial starting point of the improved difference evolution algorithm is lower than that of the classical differential evolution algorithm. The reason for this is that the improved difference evolution algorithm replaces three random solutions with three feasible solutions in the initial population. Later, both of them quickly find the same optimal value. In Figure 5b, the classical differential evolution algorithm falls into local optimum in the 200th generation. However, the improved differential evolution algorithm finds a better solution. This shows the effectiveness of the improved mutation strategy.



(**a**)



(**b**)

Figure 5. Convergence curves of the improved differential evolution algorithm and comparison algorithm. (**a**) 3–3, (**b**) 100–80.

6. Conclusions

Based on the production process of a large domestic iron and steel enterprise, this paper analyzes problems in slab allocation and hot-rolling scheduling. To make full use of latent heat from high-temperature slabs to reduce energy consumption and hot slab waiting times, two subproblems are specially integrated for research. A linear 0–1-integer programming model is established in accordance with the objectives and constraints of the slab allocation and hot-rolling scheduling integrated optimization problem.

In view of the large scale and large amount of data, an improved differential evolution algorithm is adopted to solve the problem. Based on the complex data structure of this problem, an efficient real coding method is proposed; that is, the integer part represents the order to which the slab is allocated, and the decimal part represents the rolling unit to which the slab is assigned. To improve the quality of solutions in the initial population, three different heuristic algorithms are proposed to generate initial solutions. In addition, multiple parent individuals participate in mutation, which improves population diversity. Furthermore, infeasible solutions are repaired, and the population gene pool is updated. In 14 data groups of different scales, the improved differential evolution algorithm can obtain fairly good solutions in a reasonable amount of time compared with experiments using CPLEX, the simulated artificial method, and the classical differential evolution algorithm. Compared with CPLEX, the improved differential evolution algorithm is better at solving large-scale slab allocation and hot-rolling scheduling integrated optimization problems.

By using the model and algorithm proposed in this paper, high-quality allocation schemes and hot-rolling schedules can be generated quickly. The utilization rate and rolling quality of slabs can be improved. The coordinal optimization of slab allocation and hot-

rolling scheduling can be realized. The benefits and market competitiveness of enterprises can be enhanced. This not only provides a theoretical reference for solving large-scale practical problems of the same type but also expands differential evolution algorithm application fields. Future research can be carried out in two ways. One further improves the differential evolution algorithm. The other applies the differential evolution algorithm to other fields, such as coil allocation and cold rolling in iron and steel enterprises.

Author Contributions: Conceptualization, L.S. and Y.M.; methodology, L.S. and Y.M.; software, L.S. and X.G.; validation, L.S. and Q.G.; formal analysis, L.S.; investigation, Q.G.; resources, Y.M. and X.G.; writing—original draft preparation, L.S. and Y.M.; writing—review and editing, L.S. and Q.G.; supervision, Q.G.; project administration, L.S. and Y.M.; funding acquisition, Y.M. and Q.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Major Program of the National Natural Science Foundation of China under Grant 72192830 and Grant 72192835, the National Natural Science Foundation of China under Grant 72002028, the National Key Research and Development Program of China under Grant 2021YFC2902403, and the 111 Project under Grant B16009.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tang, L.; Meng, Y. Data analytics and optimization for smart industry. *Front. Eng. Manag.* **2021**, *8*, 157–171. [CrossRef]
2. Liu, C.; Tang, L.; Zhao, C. A Novel dynamic operation optimization method based on multiobjective deep reinforcement learning for steelmaking process. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–15. [CrossRef]
3. Zhao, G.; Liu, J.; Tang, L.; Zhao, R.; Dong, Y. Model and heuristic solutions for the multiple double-load crane scheduling problem in slab yards. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1307–1319. [CrossRef]
4. Jiang, S.-L.; Li, W.; Zhang, X.; Xu, C. An improved pareto local search for solving bi-objective scheduling problems in hot rolling mills. *Comput. Ind. Eng.* **2022**, *172*, 108561. [CrossRef]
5. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
6. Wang, X.; Dong, Z.; Tang, L.; Zhang, Q. Multiobjective multitask optimization-neighborhood as a bridge for knowledge transfer. *IEEE Trans. Evol. Comput.* **2023**, *27*, 155–169. [CrossRef]
7. Omidvar, M.N.; Li, X.; Yao, X. A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization—Part I. *IEEE Trans. Evol. Comput.* **2022**, *26*, 802–822. [CrossRef]
8. Zhang, T.; Zheng, Q.P.; Fang, Y.; Zhang, Y. Multi-level inventory matching and order planning under the hybrid make-to-order/make-to-stock production environment for steel plants via particle swarm optimization. *Comput. Ind. Eng.* **2015**, *87*, 238–249. [CrossRef]
9. Tang, L.; Luo, J.; Liu, J. Modelling and a tabu search solution for the slab reallocation problem in the steel industry. *Int. J. Prod. Res.* **2013**, *51*, 4405–4420. [CrossRef]
10. Lv, Y.; Wang, G. Data analytics for slab matching time problem. In Proceedings of the 11th World Congress on Intelligent Control and Automation, Shenyang, China, 29 June–4 July 2014; pp. 2761–2764. [CrossRef]
11. Lv, Y.; Wang, G.; Tang, L. Scenario-based modeling approach and scatter search algorithm for the stochastic slab allocation problem in steel industry. *ISIJ Int.* **2014**, *54*, 1324–1333. [CrossRef]
12. Tang, L.; Liu, J.; Rong, A.; Yang, Z. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *Eur. J. Oper. Res.* **2000**, *124*, 267–282. [CrossRef]
13. Tang, L.; Wang, X. Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *Int. J. Adv. Manuf. Technol.* **2006**, *29*, 1246–1258. [CrossRef]
14. Li, K.; Tian, H. A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Appl. Soft Comput.* **2016**, *43*, 469–479. [CrossRef]
15. Özgür, A.; Uygun, Y.; Hütt, M.-T. A review of planning and scheduling methods for hot rolling mills in steel production. *Comput. Ind. Eng.* **2021**, *151*, 106606. [CrossRef]
16. Yang, L.; Jiang, G.; Chen, X.; Li, G.; Li, T.; Chen, X. Design of integrated steel production scheduling knowledge network system. *Clust. Comput.* **2019**, *22*, 10197–10206. [CrossRef]
17. Li, F.; Zhang, Y.; Wei, H.; Lai, X. Integrated problem of soaking pit heating and hot rolling scheduling in steel plants. *Comput. Oper. Res.* **2019**, *108*, 238–246. [CrossRef]

18. Tan, Y.; Zhou, M.; Wang, Y.; Guo, X.; Qi, L. A Hybrid MIP–CP approach to multistage scheduling problem in continuous casting and hot-rolling processes. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1860–1869. [CrossRef]
19. Pan, Q.-K.; Chen, Q.-D.; Meng, T.; Wang, B.; Gao, L. A mathematical model and two-stage heuristic for hot rolling scheduling in compact strip production. *Appl. Math. Model.* **2017**, *48*, 516–533. [CrossRef]
20. Puttkammer, K.; Wichmann, M.G.; Spengler, T.S. A GRASP heuristic for the hot strip mill scheduling problem under consideration of energy consumption. *J. Bus. Econ.* **2016**, *86*, 537–573. [CrossRef]
21. Hu, W.; Zheng, Z.; Gao, X.; Pardalos, P.M. An improved method for the hot strip mill production scheduling problem. *Int. J. Prod. Res.* **2019**, *57*, 3238–3254. [CrossRef]
22. Pan, Q.-K.; Gao, L.; Wang, L. A multi-objective hot-rolling scheduling problem in the compact strip production. *Appl. Math. Model.* **2019**, *73*, 327–348. [CrossRef]
23. Li, J.-Y.; Zhan, Z.-H.; Tan, K.C.; Zhang, J. A meta-knowledge transfer-based differential evolution for multitask optimization. *IEEE Trans. Evol. Comput.* **2022**, *26*, 719–734. [CrossRef]
24. Tian, M.; Gao, X. Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. *Inf. Sci.* **2019**, *478*, 422–448. [CrossRef]
25. Li, Y.; Fang, L. Robust multi-objective optimization of rolling schedule for tandem cold rolling based on evolutionary direction differential evolution algorithm. *J. Iron Steel Res. Int.* **2017**, *24*, 795–802. [CrossRef]
26. Chakraborti, N.; Kumar, A. The optimal scheduling of a reversing strip mill: Studies using multipopulation genetic algorithms and Differential Evolution. *Mater. Manuf. Process.* **2003**, *18*, 433–445. [CrossRef]
27. Tang, L.; Zhao, Y.; Liu, J. An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production. *IEEE Trans. Evol. Comput.* **2014**, *18*, 209–225. [CrossRef]
28. Zhao, F.; Shao, Z.; Wang, J.; Zhang, C. A hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search for job shop scheduling problems. *Int. J. Prod. Res.* **2016**, *54*, 1039–1060. [CrossRef]
29. Tang, L.; Li, Z.; Hao, J.-K. Solving the single-row facility layout problem by K-Medoids memetic permutation group. *IEEE Trans. Evol. Comput.* **2022**, *27*, 251–265. [CrossRef]
30. Li, T.; Meng, Y.; Tang, L. Scheduling of continuous annealing with a multi-objective differential evolution algorithm based on deep reinforcement learning. *IEEE Trans. Autom. Sci. Eng.* **2023**, 1–14. [CrossRef]