mathematics

MDPI

Article

An Improved Migrating Birds Optimization Algorithm for a Hybrid Flow Shop Scheduling within Steel Plants

Dayong Han [1,2], Qiuhua Tang [1,2,*], Zikai Zhang [1,2] and Zixiang Li [1,2]

[1] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Ministry of Education, Wuhan 430000, China; wust_han@163.com (D.H.); zhangzikai0703@gmail.com (Z.Z.); zixiangliwust@gmail.com (Z.L.)

[2] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan 430000, China

* Correspondence: tangqiuhua@wust.edu.cn

Received: 9 September 2020; Accepted: 23 September 2020; Published: 26 September 2020

Abstract: Steelmaking and the continuous-casting (SCC) scheduling problem is a realistic hybrid flow shop scheduling problem with continuous-casting production at the last stage. This study considers the SCC scheduling problem with diverse products, which is a vital and difficult problem in steel plants. To tackle this problem, this study first presents the mixed-integer linear programming (MILP) model to minimize the objective of makespan. Then, an improved migrating birds optimization algorithm (IMBO) is proposed to tackle this considered NP-hard problem. In the proposed IMBO, several improvements are employed to achieve the proper balance between exploration and exploitation. Specifically, a two-level decoding procedure is designed to achieve feasible solutions; the simulated annealing-based acceptance criterion is employed to ensure the diversity of the population and help the algorithm to escape from being trapped in local optima; a competitive mechanism is developed to emphasize exploitation capacity by searching around the most promising solution space. The computational experiments demonstrate that the proposed IMBO obtains competing performance and it outperforms seven other implemented algorithms in the comparative study.

Keywords: production scheduling; hybrid flow shop scheduling; steelmaking and continuous-casting; migrating birds optimization; MILP

1. Introduction

The common flow shop scheduling problem (FSP) is a complex combinatorial optimization problem which has great applications in industry [1–3]. A wide variety of approaches have been developed to solve the FSP and its variants. This study considers a realistic hybrid flow shop scheduling problem (HFSP) in the steel plants, namely the steelmaking and continuous-casting (SCC) scheduling problem. In the literature, the SCC scheduling problem is usually regarded as a hybrid flow shop scheduling problem (HFSP) with continuous-casting production at the last stage.

As is common knowledge, SCC is the key process in the steel production system to manufacture billet or slabs with user-defined chemical compositions [4]. In this process, hot metal is first smelted into molten steel with a strictly predefined chemical composition, then it is cast into solidified parts. The general HFSP consists of k stages and each stage has one or more parallel machines. A set of N jobs need to be operated by the machines available at each stage in sequence. Similarly, the SCC scheduling problem consists of three stages (steelmaking, refining, and continuous casting) in series, where each stage can have several machines in parallel, and a set of charges are operated by the machines available

at each stage in sequence. Nevertheless, SCC production has three main characteristics: (1) Continuous production mode and intermittent production mode coexist. At the first two phases, charge is regarded as the minimum production unit, corresponding to the definition of job in the flow shop scheduling. Nevertheless, the charges with similar chemical compositions must be operated together in sequence on the same caster at the last stage. These charges consist of one batch (also called cast in some studies) and hence the batch with several charges is the basic production unit on the casters. (2) The casting process has strict limitations on temperature. Molten steel is required to keep above a certain temperature to ensure the technological requirements for casting. (3) Setup time is required for production preparation between any two adjacent batches. All these features make the scheduling of SCC scheduling problem a more difficult problem than the general HFSP [5,6].

On the other hand, since diversified steel products are in need, the ingredients and specifications of steel products have been further subdivided. This results in the increasing of batch diversity and highly frequent utilization of the refining process. These changes further increase the time complexity for solving the SCC scheduling problem, and lead to a large difficulty in developing a reasonable and effective production schedule. With regard to the objectives of this problem, the minimization objective of the makespan (also called the maximum completion time) is commonly utilized as a performance evaluator. This objective reflects productivity accurately, and has the advantages of reducing production cycle, reducing wait time related to machines, and reducing tardiness penalties responding to punctual delivery [6,7].

Extending the standard notation in [8], this considered SCC scheduling problem is described as $FHm,\ ((PM)^{(i)})_{k=1}^{M_i}\|\ \{C_{max}\}$. In the above expression, FHm means the problem is a hybrid flow shop scheduling problem with predefined number of stages; $((PM)^{(i)})_{k=1}^{M_i}$ means that there are one or more identical parallel machines at each stage, $k = 1, 2 \ldots M_i$; C_{max} represents the objective of optimizing makespan. The simplest scenario of FHm, which involves two stages and a single unit at one stage and two identical units at another stage, has been proven NP-hard [9]. Hence, the proposed SCC scheduling problem also belongs to the NP-hard category and it is difficult to obtain a high-quality schedule with considering an acceptable computation time.

Therefore, this study presents an improved migrating birds optimization algorithm (IMBO) to tackle the SCC scheduling problem with diverse products effectively. The proposed IMBO utilizes two neighborhood structures to intensify exploration and proposes the simulated annealing-based acceptance criterion to ensure the diversity of the population and further to escape from being trapped in local optima [10,11]. Meanwhile, the proposed method also employs a competitive mechanism to emphasize exploitation capacity by searching around the most promising solution space. All these improvements are employed to help the proposed algorithm to achieve the proper balance between exploration and exploitation. Notice that this is the first attempt to apply IMBO to solve the SCC scheduling problem. Computational study demonstrates that the proposed IMBO obtains competing performance and it outperforms the original migrating birds optimization algorithm and seven other well-known algorithms.

The remainder part of this paper is organized as follows: Section 2 provides the literature review, and Section 3 presents the problem description and mathematical formulation. Subsequently, Section 4 introduces the proposed improved migrating birds optimization algorithm. Section 5 reports the experimental results and finally Section 6 provides conclusions and future research venues.

2. Literature Review

For solving the general HFSP scheduling problem and its variants, many approaches have been developed to obtain the optimal or near-optimal solutions [12–14]. These approaches include exact algorithms [15,16], heuristic methods [17,18] and meta-heuristics methods [3,11]. This study mainly focuses on recent achievements on the SCC scheduling problem.

Exact methods attempt to solve the SCC scheduling problem optimally, and they could also provide the duality gaps or lower bounds for evaluating the solutions. Bellabdaoui and Teghem [19]

presented a SCC scheduling model based on mixed-integer mathematical programming; Harjunkoski and Grossmann [20] presented a decomposition-based mathematical programming strategy for solving the large-size scheduling problems. These mathematical models are solved by branch and bound method or branch and cutting method. Tang, et al. [21] presented a mathematical programming formulation and then developed a solution methodology by combining Lagrangian relaxation and heuristics to solve the problem. However, it is worth mentioning that, as the problem size increases, the computational time spent by exact methods grows greatly. Hence, exact methods can only obtain the optimal solutions of small-size instances, whereas they cannot solve the large-size instances effectively within an acceptable computation time.

Heuristic methods are developed to achieve feasible solutions in a short computational time. Peng, et al. [22] developed a heuristic scheduling procedure with an optional figure-based directional search process. Hauber [23] put forward a heuristic algorithm consisting of three levels: cast scheduling on casters, scheduling of the other stages and exact timing of all operations. Buyukozkan, et al. [24] studied the SCC scheduling problem with task start time delay, and they designed an efficient heuristic scheduling method to respond to any interruption in the production quickly. However, heuristic algorithms might not obtain high-quality solutions for some instances.

Compared to heuristic algorithms, meta-heuristics can achieve better solutions at the cost of additional computation time [25,26]. Regarding the meta-heuristics, Atighehchian, et al. [27] combined ant colony optimization and iterative algorithm. Considering the controllable processing times at the last stage, Jiang, et al. [5] presented an improved differential evolution algorithm. To solve the SCC scheduling problem with dynamic operation and skipping features, Li and Pan [28] proposed an improved discrete artificial bee colony (DABC) algorithm. The above studies suggest that meta-heuristics especially the population-based optimization algorithms are quite competent to solve the SCC scheduling problem.

As you can see, the above studies do not consider the SCC scheduling problem with diverse products. Hence, this study formulates this problem and develops a metaheuristic, namely the migrating birds optimization (MBO) algorithm, to solve the considered SCC scheduling problem. The MBO algorithm has been successfully applied to hybrid combination optimization problems including the flow shop scheduling problems [11,29], flexible manufacturing systems [30], assembling line balancing problems [31,32], task allocation problem [10], etc. The MBO algorithm has a unique evolution mechanism with sharing mechanism which differentiates it from the other algorithms. The MBO is selected in this study for two reasons as follows. Firstly, MBO is a population algorithm based on local search and it is capable of achieving the local optimal solutions in a short time. Secondly, the sharing mechanism replaces the poor-quality solutions with the neighbor solutions of other individuals in the swarm, and it accelerates the evolution process by replacing the poor-quality solutions. Last but not the least, we also present several improvements to enhance the MBO algorithm to achieve proper balance between exploration and exploitation. Notice that this is the first attempt to apply the MBO to solve the SCC scheduling problem in consideration.

3. Problem Description and Mathematical Formulation

This section first provides the problem description and later presents the mathematical formulation.

3.1. Problem Description

SCC production mainly contains three phases: steelmaking, refining and continuous casting. Figure 1 presents a schematic diagram of the SCC production process. At the steelmaking phase, hot metal and steel scrap is smelted into molten steel via several identical and parallel smelting converter furnaces, Linz-Donawitzor Gas (LD) furnaces. Subsequently, molten steel is reserved in a ladle pot as a charge. Then, each charge is transferred to the refining stages to adjust its chemical compositions on Ladle-Furnace (LF) or even to vacuum degassing in hydrogen removal on Ruhrstahl-Heraeus (RH). Once the temperature and chemical compositions are in accordance with the technological

requirements, charges are finally sent to their designated continuous caster (CC) to produce slabs. At the steelmaking and refining stages, the charge is assigned to any parallel machine at each stage. Nevertheless, the charges with similar chemical compositions must be operated in together on the same caster, and these charges consist of one batch (cast). Each batch contains one or several charges with similar chemical compositions, and all the charges in a batch should be operated in the continuous production mode. On the continuous caster, when completing a batch on a continuous caster, a setup time is needed for production preparation of the immediately following batch on this caster.

Assume for this realistic hybrid flow shop scheduling problem, there are J (j = 1, 2, . . . , J) charges, L (l = 1, 2, . . . , L) batches. At each processing stage, there are Mi identical and parallel machines. In batch l, LJ (l,j) charges belonging to this batch have exactly one processing route. The processing time of a charge j related to stage i is denoted by $PT_{i,j}$. The setup time for production preparation is Su and the total number of unit-specific event points is denoted as T. The main purpose of the SCC scheduling problem is to determine:

(1) Optimal batch schedule including the allocation, sequencing and timing of each batch at the last stage;
(2) Optimal schedule for all charges including their allocation, sequencing and timings at other stages.

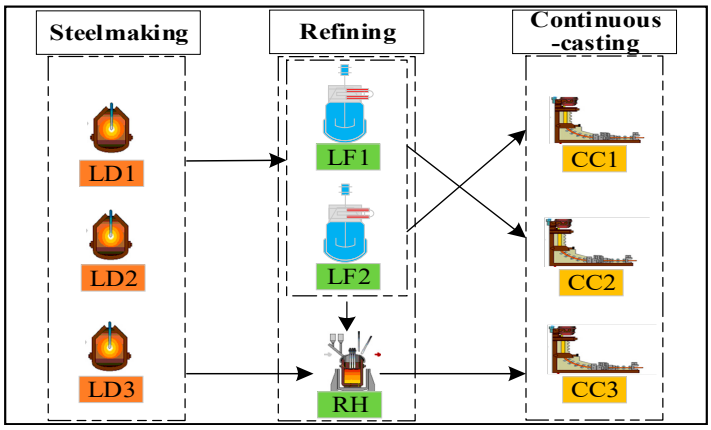


Figure 1. Steelmaking and continuous-casting production.

The main assumptions of this study are presented as follows. Notice that the SCC scheduling problem addressed in this study is different from those reported in the literature [33] as diverse products are considered. Namely, in this study the charges in different batches may have different processing routes, and clearly the considered problem is more complex than that in [16].

(1) The sequence of the charges belonging to the same batch is predefined;
(2) Machine malfunction is not considered;
(3) Transportation time between any two successive stages is negligible;
(4) Setup time between any two successive batches is independent of batch sequence and properties;
(5) Unlimited storage capacity is provided between stages;
(6) All charges and processing units are available for processing at time zero.

3.2. Mathematical Formulation

Although there are different modelling approaches including discrete- and continuous-time representations for scheduling problems, the unit-specific event-based modelling approach, which is a kind of continuous-time representation, has shown clear advantages in the literature [34,35]. The modeling method based on unit-specific event-based representation is very flexible and capable of expressing many hybrid scheduling characteristics. They have successfully been applied to various scheduling problems both in academic research and industrial applications. Hence, this study utilizes

the unit-specific event-based modelling approach to represent the scheduling horizon, where the scheduling horizon is divided into T (t = 1, 2, . . . , T) event points. Each machine unit has T event points, and the location of the same event points for different machine units may be different as shown in Figure 2.

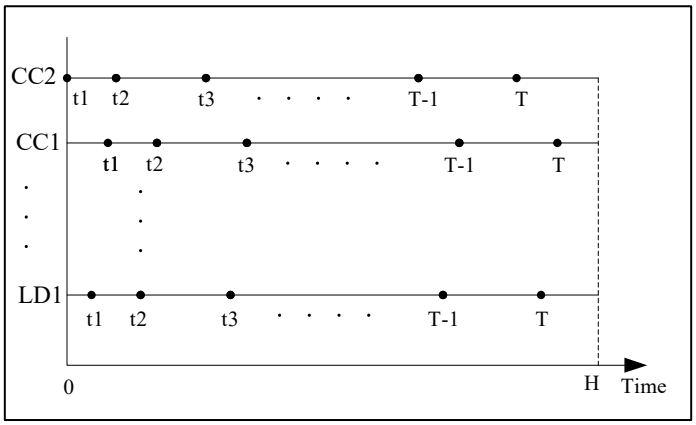


Figure 2. Continuous time representation based on unit specific event.

To formulate the problem, it is supposed that all the charges undergo every stage but the processing time is set to zero when the charges do not need the stages in real production. Before introducing the mathematical formulation, the utilized notations are introduced first as the following in Table 1.

Table 1. Indices, parameters and variables of the proposed model.

Indices:	
i	The stage.
j	The charge.
l	The batch.
m	The machine.
t	The event points.
Sets:	
I	The set of stages, $I = \{1, 2 \cdots s\}$, where s denotes the last casting stage.
J	The set of charges, $J = \{1, 2 \cdots J\}$, where J is the total number of charges.
L	The number of batches.
T	The set of event points, $T = \{1,\ 2. . . T\}$, where T is the total number of unit-specific event points in the scheduling horizon.
$LJ_{(l,j)}$	Set of charge indices in the lth batch, $l = \{1, 2 \cdots L\}$, where L is the total number of batches, and $\sum_l LJ_{(l,j)} = n,\ LJ_{(l,j)} \cap LJ_{(l',j)} = \varnothing$ for *all* $l \neq l'$.
M_i	The set of machines at stage i.
$LJF_{(l,j)}$	The set of first charges of batches.
$LJE_{(l,j)}$	The set of last charges of batches.
Parameters:	
U	A big positive number.
$PT_{i,j}$	The processing time of the charge j at stage i.
S_u	The setup time between adjacent batches on the caster for production preparation for the next batches.
Variables:	
$X_{j,m,t}$	Binary variable. $X_{j,m,t}$ = 1, if charge j is being processed at event point t on machine m; $X_{j,m,t}$ = 0, otherwise.
$Y_{l,m}$	Binary variable. $Y_{l,m}\ = 1$, if batch l is processed on caster m at the last stage; $Y_{l,m}\ = 0$, otherwise.
$Ts_{m,t}$	Continuous variable; start time of event point t on machine m.
$Tf_{m,t}$	Continuous variable; finish time of event point t on machine m.
$Tsi_{i,j}$	Continuous variable; start time of charge j at stage i.
$Cmax$	Makespan; the completion time of the last charge on the continuous casters.

On the basis of the utilized notations, the mathematical model is formulated with expressions (1)–(19) to minimize the makespan.

$$\text{Min } Cmax \tag{1}$$

$$Cmax \geq TS_{m,t} + \sum_j \sum_{t'} X_{j,m,t'} \times PT_{i,j} + \sum_{t'<|T|} \left(TS_{m,t'+1} - TS_{m,t'} - \sum_j \sum_{t'} X_{j,m,t'} \times PT_{i,j} \right), \quad \forall i = s,\ m \in M_i,\ t = 1 \tag{2}$$

$$Cmax \geq TF_{m,t}\ \ , \forall i = s,\ m \in M_i,\ t = T \tag{3}$$

$$\sum\nolimits_j X_{j,m,t} \leq 1,\ \forall m,\ t \tag{4}$$

$$\sum\nolimits_{m=1}^{m_i} \sum\nolimits_t X_{j,m,t} = 1,\ \forall j, i \tag{5}$$

$$\sum\nolimits_{m=1}^{m_i} \sum\nolimits_t Y_{l,m} = 1,\ \forall l,\ i = s \tag{6}$$

$$\sum_{j \in LJ_{(l,j)}} \sum_t X_{j,m,t} \geq Y_{l,m},\ \ \forall i = s, m \in M_i \tag{7}$$

$$X_{j+1,m,t+1} \geq X_{j,m,t},\ \forall (j, j+1) \in LJ_{(l,j)}, i = s, m \in M_i, t < T \tag{8}$$

$$\sum\nolimits_j X_{j,m,t+1} \leq \sum\nolimits_j X_{j,m,t},\quad \forall m, t < T \tag{9}$$

$$Tsi_{i+1,j} \geq Tsi_{i,j} + PT_{i,j},\ \forall j, i < s, m \in M_i \tag{10}$$

$$Tsi_{i,j+1} = Tsi_{i,j} + PT_{i,j}, \forall (j, j+1) \in LJ_{(l,j)}, i = s, m \in M_i \tag{11}$$

$$Ts_{m,t+1} \geq Tf_{m,t},\ \forall m, t < T \tag{12}$$

$$Tf_{m,t} \geq Ts_{m,t} + PT_{i,j},\ \forall i, m \in M_i, t < T \tag{13}$$

$$Ts_{m,t} \leq Tsi_{i,j} + U * \left(1 - X_{j,m,t}\right),\ \forall j, i, m \in M_i, t \tag{14}$$

$$Ts_{m,t} \geq Tsi_{i,j} - U * \left(1 - X_{j,m,t}\right),\ \forall j, i, m \in M_i, t \tag{15}$$

$$Tsi_{i,j} \geq Tf_{m,t} + Su - U \cdot \left(1 - X_{j,m,t+1}\right),\ \forall j \in LJF_{(l,j)}, i = s, m \in M_i, t < T \tag{16}$$

$$X_{j,m,t}, Y_{l,m} \in \{0, 1\},\ \forall l, j, i, m \in M_i, t \tag{17}$$

$$Tsi_{i,j} \geq 0,\ \ \forall j, i \tag{18}$$

$$Ts_{m,t}, Tf_{m,t} \geq 0,\ \forall m, t \tag{19}$$

The formulated model in expression (1) minimizes the makespan. Constraints (2) and (3) calculate the lower bounds of the *Cmax* to reduce the search scope and computational time. Constraint (2) indicates that, at the last stage s, *Cmax* must be larger than or equal to the sum of the start time of first charge, the total processing times of all charges at stage s and the total idle times on machine m. Constraint (3) indicates that, at the last stage s, *Cmax* must be larger than or equal to the end time of the last charge on the machine of the last stage.

For the remaining constraints, constraints (4) and (5) mean that a machine at any stage can operate at most one charge and a charge must be processed exactly once at any stage at each event point. Constraint (6) implies that each batch must be allocated to exactly one caster at the last stage and all charges belonging to a batch must be processed sequentially on the specified machine based on the predetermined order without interruption. Constraint (7) ensures that each event point cannot be assigned unless the previous one has been assigned. This additional constraint guarantees that all event points of each machine are allocated one by one. Constraints (8)–(13) calculate the start and the end time of each process of all charges to ensure their feasibility. Constraints (14)–(16) consider the situation when charge j at stage i is allocated to event point t on machine m. Additionally, the start time of charge j at stage i equals to that of event point t on machine m when charge j at stage i is allocated to

event point t on machine m; the start time of charge j at stage i and that of event point t on machine m are both otherwise relaxed. Constraints (17)–(19) define the scope of the decision variables.

This formulated model is a mixed-integer linear programming model, and small-size problems described with this model can be solved optimally with the Cplex solver. This mathematical model is different from that published by considering the diverse products, and this model will be tested in Section 5.3.

4. Improved Migrating Birds Optimization Algorithm

This section introduces the basic MBO, and later describes the main components of the proposed IMBO algorithm.

4.1. Introduction of the Basic MBO

Migrating birds optimization (MBO) is a recent high-performing meta-heuristic algorithm inspired by the migrating birds' flight in a V-shaped formation. Within this formation, a bird in the first, which is called leader, leads the whole flock and other birds are placed in the right and left sides of the leader equally. During the flight, if the leader gets tired, it steps back to the tail of the left or right line and the following bird with best fitness will be as the new leader. This algorithm has shown competitive performances on many combinatorial optimization problems [36,37].

In the MBO algorithm, solutions are regarded as birds in the flock. This algorithm consists of initialization, leader improvement, follower improvement, new leader selection and this main cycle is repeated until a termination criterion is met. The main procedure of the basic MBO is presented in Algorithm 1. This algorithm has four parameters: the number of individuals in the swarm (α), the number of neighbors around the leader and following bird (β), the number of neighbors to be shared with the following birds (χ), and the number of iterations before replacing the leader, which is also called the number of tours (γ).

Algorithm 1 Procedure of the basic migrating birds optimization

Input: Job permutations and algorithm parameters
%Initialization
Step 1: Generate the initial solutions randomly, and select the best one as the leader and the others are divided to form V-shape with the left line and right line, equally.
%Leader improvement
Step 2: Generate β solutions around the leader. If these new neighbors improve the leader, then substitute the leader with the best neighbor; otherwise, keep the leader unchanged. After that, other unused neighbors around the leader are sorted according to the descending order of objective values, and the 2χ neighbors with best values are divided equally to form the sharing neighbors for left and right set, respectively.
%Follower improvement
Step 3: Conduct the follower improvement along the lines toward the tails. For example, for a follower in the left (right) line, randomly create β-χ neighbors around it. Then, evaluate these β-χ solutions and χ solutions from the sharing neighbor set, which are described as Step 2. If the best neighbor has the better fitness than the follower, then replace it. Subsequently, rebuild the left (right) sharing neighbor set with the best χ unused neighbors. Perform this procedure repeatedly until all followers have been considered.
%New leader selection
Step 4: After conducting Steps 2 and 3 for γ times, reset the leader to the tail of the left or the right line, and then selected the first follower in this line as the new leader.
Step 5: If the termination criterion is satisfied, output the best solution achieved. Otherwise, return to Step 2.
Output: The achieved best solution.

To tackle the considered problem, this section presents an improved migrating birds optimization (IMBO) with several improvements. The main segments and the procedure of the proposed IMBO are presented in the following sections in detail.

4.2. Encoding and Decoding

The proposed IMBO utilizes the charge permutation for encoding, which corresponds to a charge sequence. Specifically, each element in the charge permutation is a charge. Supposed that there is a charge permutation [2, 5, 3, 6, 4, 7, 1, 8], the sequence of operating the charges at the first stage is charge 2, 5, 3, 6, 4, 7, 1 and 8.

Although the encoding for the problem under consideration is similar to that utilized in the general HFSP, it is necessary to develop a new decoding procedure to obtain the feasible solution as the more features are considered in Section 3.2. Hence, on the basis of charge permutation, this study develops the following two-level decoding procedure to obtain feasible solutions. This two-level decoding procedure consists of the FIFO-based forward heuristic and the backward heuristic, where FIFO means first-in and first-out rule.

In the FIFO-based forward heuristic, machine assignment and charge permutation on each machine are determined by the FIFO rule, which has been widely utilized in the scheduling literature. The FIFO rule gives a higher priority to the charge that has the earlier release time, and with the FIFO rule all charges are sequenced in the descending order of release times with the time complexity $O(n \bullet log(n))$. Namely, each charge is assigned to the first available machine at any stage except the last stage. At the last stage, all the charges in a batch should be assigned to the same caster. The proposed FIFO-based forward heuristic for the steelmaking and refining stages is presented in Algorithm 2.

Nevertheless, the forward heuristic is not enough to ensure the continuity of each batch in the last stage. Additionally, the forward heuristic might lead to the large wait time and may result in huge temperature drop of charges. Hence, this study furthers puts forward a backward heuristic to adjust the charges from the last stage to the first to (1) ensure that all the charges in a batch are cast continuously and (2) reduce the wait time between any two successive stages. In contrast to the forward heuristic, the backward heuristic adjusts the timing of each charge reversely from continuous casting stage to refining stages and finally to steelmaking stage. The backward heuristic is provided in Algorithm 3. Notice that Algorithms 2 and 3 present the main idea of the FIFO-based forward heuristic and the backward heuristic and the detailed procedures are presented in Appendix A.

Algorithm 2 The FIFO-based forward heuristic

Input:	Batch plan and charge permutation $\pi(j)$
Step 1:	At the steelmaking stage, select each charge based on the given charge permutation and assign the first available machine (LD) to the selected charge.
Step 2:	At the refining stage, select each charge based on the ascending order of completion times of the steelmaking stage and assign the first available machine (LF) to the selected charge.
Step 3:	At the continuous casting stage, process the charges in each batch sequentially.% The charges in each batch must be operated together in sequence.
Output:	Machine assignment, charge permutation on each machine.

Algorithm 3 The backward heuristic

Input:	Machine assignment, charge permutation on each machine.
Step 1:	At the last casting stage, each charge except for the last one in the same batch, should be moved right to eliminate the batch break when there is a batch break.
Step 2:	At the refining stage, right shift the starting time of each charge on each machine.
Step 3:	At the steelmaking stage, shift the starting time of each charge on each machine.
Output:	Production schedule.

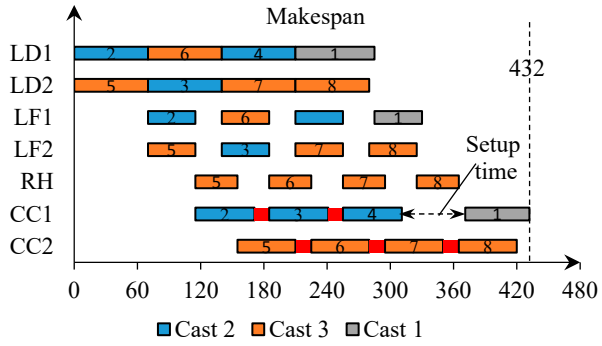
To clarify the proposed decoding process, a small-size instance is presented in Table 2 as an example. This instance has three batches and eight charges to be scheduled, and the detailed characteristics of these charges are represented in Table 2. Note that there are two parallel machines (LDs) at the steelmaking stage and two machines (CCs) at the casting stage, respectively. For the refining phase, there are two process routes, LF and LF-RH, where LF-RH means first being operated by LF and later by RH. Additionally, the LF stage has two parallel machines while the RH stage has only one machine.

Table 2. Characteristics of the given charges.

Batch	Charges	Process Route	Processing Times
1	{1}	LD-LF-CC	75-45-61
2	{2, 3, 4}	LD-LF-CC	70-45-56
3	{5, 6, 7, 8}	LD-LF-RH-CC	70-45-40-55

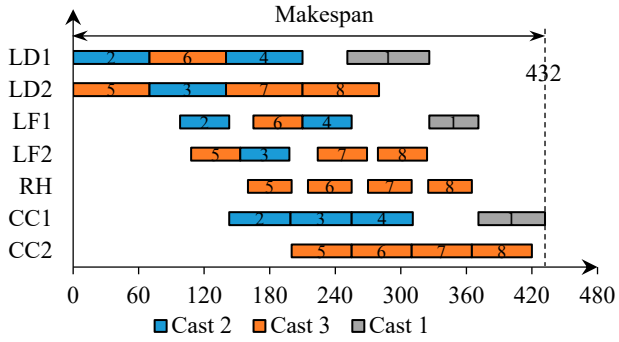
Supposing that production permutation of charges is [2, 5, 3, 6, 4, 7, 1, 8], on the basis of the FIFO-based forward heuristic, the charge 2 is assigned to LD1 and charge 3 is assigned to LD2, as presented in Figure 3a. After that, the forward heuristic allocates charge 1 to LD1 and charge 5 to LD2 and this procedure terminates until all the charges have been assigned. As you can see in Figure 3a, charges 2, 1, 6, and 8 are allocated to LD1 and the others are allocated to LD2. Since charge 2 is the first in the permutation and it does not go through RH, both CC1 and CC2 are available after it is released from LF and here CC1 is selected. Charge 3 also belongs to batch 2, and hence it is also allocated to CC1. When charge 1, the first charge in a new batch, is released from LF1, CC1 is not available. Charge 1 needs to be allocated to CC2 after waiting for a setup time. The detailed schedule by the forward heuristic is depicted in Figure 3a.

At the continuous casting stage, the backward heuristic pulls back the charges 5, 6, 7 towards charge 8 to ensure that they are cast continuously without interruption as all of them belong to batch 3. A similar method is executed for other batches. Then, the backward heuristic pulls back the release times at the previous two stages of charges in order to reduce their wait times between two adjacent stages. For example, as presented in Figure 3b, the backward heuristic pulls back the release time at the LF process of charge 1 until it equals to the start time on CC2. As a result of the backward heuristic, a feasible solution is achieved as illustrated in Figure 3b.



(**a**)

Figure 3. *Cont.*



(**b**)

Figure 3. Two-level decoding procedure for the SCC scheduling problem. (**a**) FIFO-based forward heuristic; (**b**) Backward heuristic.

4.3. Population Initialization

To achieve high-quality charge permutation, this study utilizes the heuristic initialization as presented in Algorithm 4.

Algorithm 4 Procedure of the heuristic initialization

Input:	A random batch sequence.
Step 1	Generate a batch sequence randomly and arrange the charges in each batch in the ascending order.
Step 2	Add the first charge of each batch to set $\pi_1[j]$ according to the batch sequence.
Step 3	Select a charge from $\pi_1[j]$ randomly, add this charge to the charge permutation $\pi[j]$ and then delete this selected charge from $\pi_1[j]$.
Step 4	If there are assignable charges in the batch which the deleted charge belongs to, select the first one from the remaining charges in this batch and add it to the set $\pi_1[j]$.
Step 5	If the set $\pi_1[j]$ is empty, output the achieved charge permutation $\pi[j]$; otherwise, return to Step 3.
Output:	The charge permutation $\pi[j]$.

If taking the small-size instance in Section 4.2 as an example, Table 3 exhibits the detailed procedure of generating a feasible permutation by the heuristic initialization. Supposing that a random batch sequence is [2, 3, 1], the resulting charge permutation is [2, 5, 3, 6, 4, 7, 1, 8].

Table 3. Process of heuristic initialization under batch sequence [2, 3, 1].

No.	Unallocated Batches	Assignable Charges	Selected Charge	Charge Set
1	{1, 2, 3}	{1, 2, 5}	2	2
2	{1, 2, 3}	{1, 3, 5}	5	2, 5
3	{1, 2, 3}	{1, 3, 6}	3	2, 5, 3
4	{1, 2, 3}	{1, 4, 6}	6	2, 5, 3, 6
5	{1, 2, 3}	{1, 4, 7}	4	2, 5, 3, 6, 4
6	{1, 3}	{1, 7}	7	2, 5, 3, 6, 4, 7
7	{1, 3}	{1, 8}	1	2, 5, 3, 6, 4, 7, 1
8	{3}	{8}	8	2, 5, 3, 6, 4, 7, 1, 8

To have a diverse swarm, parts of the initial individuals are obtained utilizing the above heuristic initialization and the others are achieved randomly. After obtaining a swarm of individuals, the best individual is selected as the leader and other birds are put in a hypothetical 'V' formation randomly.

4.4. Neighborhood Structures

The proposed IMBO algorithm generates β neighbor solutions around the leader and β-χ neighbor solutions around followers. This study applies two neighborhood operators: insert and swap. Insert operator removes a charge and reinserts it to another different position randomly. Swap operator selects two charges randomly and exchanges the positions of the selected charges.

The utilization of two neighborhood structures aims at enhancing the exploration capacity. Since two neighbor operators are involved, this study selects one neighborhood operator randomly to modify the individual. In other words, this study first generates a random number within [0, 1]. If this number is less than 0.5, the insert operator is utilized; otherwise, the swap operator is applied.

4.5. New Acceptance Criterion

In the original MBO, one bird is replaced with the best neighbor solution with greedy acceptance criterion. Nevertheless, this acceptance criterion might lead to the situation where the incumbent bird remains unchanged for many times and the algorithm might be trapped in local optima. Hence, this study develops a novel probabilistic acceptance criterion based on the simulated annealing algorithm. If the best neighbor outperforms the incumbent bird, the incumbent bird is directly replaced; otherwise, the incumbent bird is also replaced with the best neighbor according to the probability of $e^{-(\Delta/temp)}$, where the *temp* is calculated using expression (20). Clearly, this new acceptance criterion accepts the worse solution with a certain probability to ensure the diversity of the population and help the algorithm to escape from being trapped into local optima.

$$temp = T \times \frac{\sum_m \sum_j PT_{j,m}}{L \times n \times P} \tag{20}$$

4.6. Competitive Mechanism

In the basic MBO, the birds are put on hypothetical V-shaped formation arbitrarily, some promising birds may be placed in the tail of the flock and have few opportunities to share their neighbors. Hence, this study applies the competitive mechanism proposed by Zhang, et al. [32] to remedy this possible drawback. In the competitive mechanism, the positions of the birds in each line are adjusted as follows. The bird with the best fitness is placed in the first in corresponding line, the bird with the second-best fitness is placed in the second position, and lastly the bird with the worst fitness is placed in the last position. Clearly, this competitive mechanism guarantees that promising birds are placed in the front of the lines and have more opportunities to share their neighbors. In this study, this mechanism is executed after the improvement of leader bird to adjust the positions of the following birds in the left and right lines.

4.7. Main Procedure of the IMBO

On the basis of the components of the IMBO described above, the main procedure of the proposed IMBO is presented as follows in Algorithm 5. Here, the proposed IMBO has three main improvements: (1) the new neighborhood structures are utilized to improve the leader bird and following birds to intensify exploration; (2) a new acceptance criterion is developed to enhance the diversity of the population and help the algorithm to escape from being trapped into local optima; (3) the competitive mechanism is developed to emphasize exploitation capacity by searching around the most promising solution space. The improvements will be tested in Section 5.3.

Algorithm 5 Procedure of the improved migrating birds optimization

Input: Charge permutations and algorithm parameters
%Initialization
Step 1 Initialize α birds and put birds in a hypothetical V formation.
%Leader improvement
Step 2: Generate β neighborhood solutions around the leader and try to improve the leader bird with the neighborhood solutions. If these new neighbors bring an improvement, replace the leader with the best neighbor; otherwise, preserve the current leader. After that, unused neighbors are sorted in the descending order of objective values, then the best 2χ neighbors are selected and divided to form the left and right sharing neighbor sets, respectively.
Step 3: For each following bird x, do,
3.1 Generate β-χ neighbors according to the neighborhood structures described in Section 4.4;
3.2 Form neighborhood $\Lambda(X)$ with β-χ neighbors around the following bird and χ shared neighbors from the leader;
3.3 If the best neighbor solution x' in $\Lambda(X)$ is better than x then,
x = x';
Else If random $\leq e^{-(\Delta/temp)}$ then,
x = x';
3.4 Seek out the best χ unused neighbors to rebuild the left (right) sharing neighbor set.
%New leader selection
Step 4: After conducting Steps 2 and 3 for γ times, update the best bird and then let the best one become the new leader.
%Competitive mechanism
Step 5: Adjust the positions of the bird in the left and right lines, where the promising birds are placed in the front of the line to have more opportunities to share their neighbors.
Step 6: If the termination criterion is satisfied, output the best solution achieved. Otherwise, return to Step 2.
Output: The achieved best solution.

5. Computational Study

This section first presents the experimental design in Section 5.1, later describes the calibration of the parameters in Section 5.2. Subsequently, the performance of the formulated model and proposed IMBO is tested in Sections 5.3 and 5.4.

5.1. Experimental Design

This section describes the computational experiments employed to evaluate the performance of the proposed algorithm. Firstly, the proposed IMBO is compared with the Cplex solver and original migrating bird's optimization algorithm (MBO). To further evaluate the performance of the proposed IMBO, several other well-known algorithms are selected for comparison, including genetic algorithm (GA) [38], hybrid genetic algorithm in Tseng and Lin (GA_R) [39], memetic algorithm (MA) [40], particle swarm optimization (PSO) [41], simulated annealing (SA) [42], discrete teaching learning based optimization algorithm (DTLBO) [43], and discrete artificial bee colony algorithm (DABC) [26]. The implemented algorithms are presented in Table 4. These algorithms are selected as they have solved the problems that are, to some extent, similar to the problem under consideration. Among them, GA, PSO and SA are the well-known algorithms, GA_R, DTLBO and DABC are variants of GA, TLBO and ABC which have produced competing performances. All these algorithms are programmed in this study based on the cited papers. All the tested algorithms utilize the same encoding and decoding presented in Section 4.2 and the same neighborhood structures in Section 4.4 as highlighted in Table 4. In addition, the parameters of all the tested algorithms are calibrated using the multi-factor analysis of variance (ANOVA) in Section 5.2.

Table 4. Summary of the tested algorithms.

Algorithms	Abbreviation	Description
Genetic algorithm (Lin, et al. [38])	GA	The neighbor operations in Section 4.4 are utilized for mutation operator.
Genetic algorithm (Tseng and Lin [39])	GA_R	The neighbor operations in Section 4.4 are utilized for mutation operator.
Memetic algorithm (Deng, et al. [40])	MA	The neighbor operations in Section 4.4 are utilized for mutation operator.
Particle swarm optimization algorithm (Behnamian [41])	PSO	The neighbor operations in Section 4.4 are utilized.
Simulated annealing algorithm (Aghajani, et al. [42])	SA	The neighbor operations in Section 4.4 are utilized.
Discrete teaching learning based optimization algorithm (Li, et al. [43])	DTLBO	The neighbor operations in Section 4.4 are utilized.
Discrete artificial bee colony algorithm (Peng, et al. [26])	DABC	The neighbor operations in Section 4.4 are utilized.
Improved migrating birds optimization	IMBO	This is the algorithm described in Section 4.

On the basis of the actual production process in steelmaking plant, this study generates a set of 40 instances as follows.

(1) As there are two forms in refining phase (LF or first LF and later RH) as presented in Figure 1 in real applications, the number of stages in the principal processes of primary SCC varies among two levels: 3, 4. Additionally, the numbers of LD, LF, RH and CC are set as 2, 2, 1 and 2, respectively.
(2) The number of charges in each batch varies among five levels: 1, 2, 3, 4 and 5. Additionally, the number of batches and charges is generated uniformly within [1, 34] and [1, 114] respectively.
(3) The processing time $PT_{i,j}$ of steelmaking, refining and casting phases are generated uniformly within [70, 80], [40, 50] and [55, 70] respectively. Note that transportation time is included in the processing time in this study [7, 26].
(4) As a relatively long setup time is in need for production preparation, the setup time is set as 60 in this study.

The generated instances are described in Table 5. The detailed instances are not exhibited for space reasons, but they are available upon request.

To evaluate the performance of IMBO method on different instances, the relative percentage deviation (*RPD*) is utilized to measure the achieved results utilizing expression (21). Here, CM_{Some} is the makespan achieved by one algorithm and CM_{Best} is the best makespan obtained by all algorithms for the same instance.

$$RPD = \frac{CM_{Some} - CM_{Best}}{CM_{Best}} \times 100 \quad (21)$$

Clearly, the algorithm with the smaller *RPD* value has the better performance. All the algorithms solve the instances for 10 times, and two indicators are employed here: *RPD*-Avg and *RPD*-Min, where *RPD*-Avg is the average value of the *RPD* values for each instance and *RPD*-Min is the minimum value of the *RPD* values for each instance in 10 repetitions. The tested problem is solved utilizing the Cplex solver of General Algebraic Modeling System 23.0. The proposed IMBO and all the other tested algorithms are implemented utilizing the C++ programming language and are running on a personal computer equipped with Intel (R) Core (TM) 2.80 GHz i5 CPU and 2.00 GB RAM.

Table 5. The data for 40 instances.

Instance	Casts	Charges	Instance	Casts	Charges
Case 1	3	8	Case 21	20	61
Case 2	3	8	Case 22	20	61
Case 3	3	8	Case 23	20	61
Case 4	3	8	Case 24	20	61
Case 5	3	8	Case 25	20	61
Case 6	5	18	Case 26	25	76
Case 7	5	18	Case 27	25	76
Case 8	5	18	Case 28	25	76
Case 9	5	18	Case 29	25	76
Case 10	5	18	Case 30	25	76
Case 11	10	30	Case 31	30	96
Case 12	10	30	Case 32	30	96
Case 13	10	30	Case 33	30	96
Case 14	10	30	Case 34	30	96
Case 15	10	30	Case 35	30	96
Case 16	15	47	Case 36	34	114
Case 17	15	47	Case 37	34	114
Case 18	15	47	Case 38	34	114
Case 19	15	47	Case 39	34	114
Case 20	15	47	Case 40	34	114

5.2. Calibration of Algorithmic Parameters

Since parameters have a significant effect on the performance of algorithms, this section presents the calibration the parameters of the implemented algorithms. Here, this study utilizes the Taguchi method as the design of experiment (DOE) to decide the desirable values of the parameters. For IMBO, there are five parameters (or controlled factors): the number of initial solutions (α), the number of neighbor solutions (β), the number of neighbor solutions to be shared (χ), the number of tours (γ) and the initial temperature (τ). This study utilizes orthogonal array to arrange experiments and there are L16(4^5) experiment combinations to analyze the instance of case 40. Each experiment combination is run 10 times and the minimum relative percentage deviation or minimum *RPD* is obtained according to Equation (21). Then, the average value of 10 minimum *RPD* for each combination is regarded as the response variable. The levels of these parameters and the corresponding response variable values of the combinations are presented in Table 6.

After obtaining response variable values, the multi-factor analysis of variance (ANOVA), a powerful parametric statistical technique, is carried out after checking the normality, homoscedasticity, and independence of the residuals. The detailed ANOVA results are illustrated in Table 7. Here, the parameter χ has the largest F-ratio of 7.75, indicating that this parameter, the number of neighbor solutions to be shared, has the greatest influence on the proposed algorithm. This also shows that the sharing mechanism makes the MBO a better competitive performance by searching around the most promising solution space. Similarly, the initial temperature (τ), which is related to the simulated annealing-based acceptance criterion to ensure the diversity of the population, has the second greatest influence on the proposed algorithm. In addition, if ranking the parameters in the decreasing order of the F-ratio values, the sequence is χ, τ, α, γ and β, where the former parameters have the greater impacts.

Table 6. Orthogonal array of IMBO.

No.	Levels of Parameters					Minimum *RPD*										Average
	α	β	χ	γ	τ	1	2	3	4	5	6	7	8	9	10	
1	9	5	3	50	0.4	0.14	0.14	0.13	0.17	0.14	0.17	0.17	0.13	0.10	0.10	0.14
2	9	8	4	100	0.5	0.14	0.17	0.14	0.13	0.14	0.14	0.11	0.11	0.11	0.11	0.13
3	9	9	5	150	0.45	0.15	0.15	0.14	0.13	0.12	0.13	0.12	0.12	0.11	0.12	0.13
4	9	12	6	200	0.6	0.12	0.09	0.09	0.09	0.09	0.06	0.06	0.06	0.06	0.10	0.08
5	11	5	4	150	0.6	0.19	0.14	0.14	0.11	0.05	0.11	0.09	0.11	0.10	0.11	0.12
6	11	8	3	200	0.45	0.14	0.15	0.15	0.14	0.11	0.11	0.11	0.11	0.11	0.09	0.12
7	11	9	6	50	0.5	0.09	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
8	11	12	5	100	0.4	0.20	0.13	0.10	0.07	0.06	0.06	0.06	0.03	0.00	0.03	0.07
9	13	5	5	200	0.5	0.18	0.17	0.18	0.11	0.14	0.14	0.15	0.14	0.13	0.11	0.15
10	13	8	6	150	0.4	0.09	0.09	0.09	0.09	0.09	0.09	0.06	0.09	0.09	0.09	0.09
11	13	9	3	100	0.6	0.14	0.11	0.11	0.11	0.11	0.09	0.10	0.07	0.10	0.06	0.10
12	13	12	4	50	0.45	0.14	0.11	0.11	0.11	0.11	0.11	0.11	0.09	0.11	0.11	0.11
13	15	5	3	100	0.45	0.20	0.17	0.13	0.15	0.14	0.09	0.11	0.09	0.09	0.09	0.13
14	15	8	5	50	0.6	0.14	0.11	0.09	0.09	0.09	0.09	0.09	0.11	0.09	0.10	0.10
15	15	9	4	200	0.4	0.09	0.14	0.15	0.11	0.09	0.09	0.09	0.09	0.09	0.05	0.10
16	15	12	6	150	0.5	0.09	0.14	0.15	0.11	0.09	0.09	0.09	0.09	0.09	0.05	0.10

Table 7. The ANOVA results for the parameters.

Sources	df	Seq SS	Adj SS	Adj MS	F-Ratio	*p* Value
α	3	0.19837	0.19837	0.06612	5.47	0.001
β	3	0.60459	0.09244	0.03081	2.55	0.058
χ	3	0.21238	0.28081	0.0936	7.75	0
γ	3	0.08074	0.11768	0.03923	3.25	0.024
τ	3	0.26197	0.26197	0.08732	7.23	0
Error	144	1.73975	0.01208			
Total	159	3.0977				

The signal–noise ratio (SNR)'s main effects on the plot of the algorithm parameters is illustrated in Figure 4. As seen in this figure, the best combination of parameters is: $\alpha = 11$, $\beta = 12$, $\chi = 6$, $\gamma = 150$, and $T = 0.4$. The selected parameter values of the proposed IMBO are also presented in Table 8.

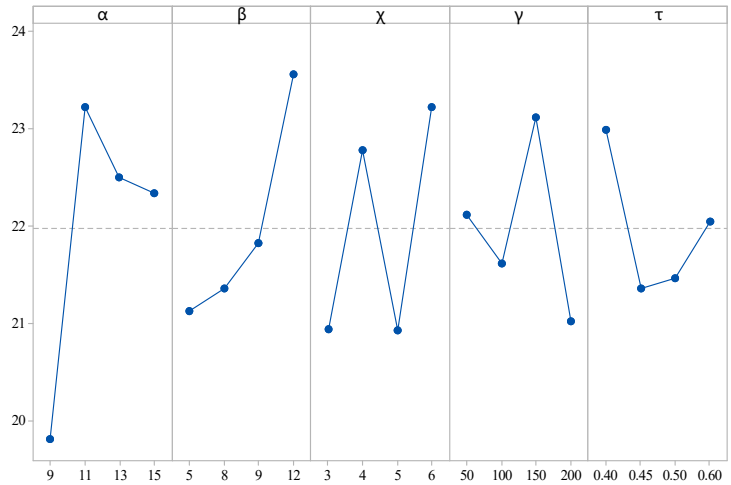
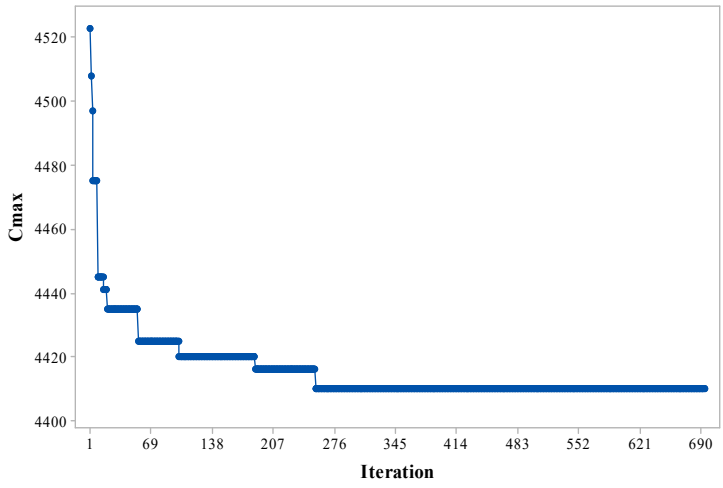


Figure 4. SNR main effects plot.

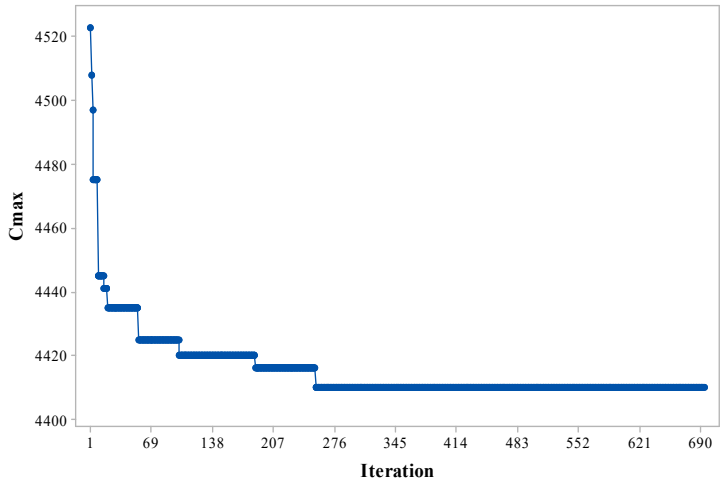
Table 8. Selected parameter values of the proposed IMBO.

Parameters	Symbol	Levels	Selected Value
The number of initial solutions	α	9, 11, 13, 15	11
The number of neighbor solutions	β	5, 8, 9, 12	12
The number of neighbor solutions to be shared	χ	3, 4, 5, 6	6
The number of tours	γ	50, 100, 150, 200	150
The initial temperature	τ	04, 0.45, 0.5, 0.6	0.4

After the determination with the best parameters' combination, $\alpha = 11$, $\beta = 12$, $\chi = 6$, $\gamma = 100$, and $\tau = 0.4$. Figure 5 is here to further illustrate the evolution of the IMBO for different instances and note that the average of 10 objective values of Cmax for each iteration is regarded as the response variable. As can be seen, there is a clear decreasing trend which shows the convergence of the proposed IMBO.



(**a**)



(**b**)

Figure 5. Convergence analysis for different instances when $\rho = 10$. (**a**) The instance with 20 casts and 61 charges; (**b**) The instance with 40 casts and 114 charges.

In addition, the parameters of other implemented algorithms are also calibrated utilizing the same method above. For space reasons, the detailed calibration results are omitted and Table 9 presents the selected best combinations of these algorithmic parameters.

Table 9. Parameter values of the comparison algorithms.

Parameters	Testing Range	Value
GA algorithm (GA)		
Population size	40, 60, 80	60
Crossover rate	0.7, 0.8, 0.9	0.9
Mutation rate	0.1, 0.2, 0.3	0.2
GA algorithm (GAR)		
Population size	40, 80, 120	80
Crossover rate	0.7, 0.8, 0.9	0.8
Mutation rate	0.1, 0.2, 0.3	0.3
Memetic Algorithm (MA)		
Population size	40, 60, 80	60
Crossover rate	0.7, 0.8, 0.9	0.9
Mutation rate	0.1, 0.15, 0.2	0.2
PSO algorithm (PSO)		
The number of particles	40, 60, 80	60
SA algorithm (SA)		
Initial temperature	0.5, 0.75, 1.0	0.5
Cooling rate	0.8, 0.9, 0.95	0.9
Maximum number of iterations	500, 1000, 1500	500
Discrete TLBO algorithm (DTLBO)		
Population size	30, 40, 80	40
Discrete artificial bee colony algorithm (DABC)		
The number of scout bees	3, 5, 6	5
The number of following bees	10, 15, 18	15
Life time	30, 40, 60	40

5.3. Performance Evaluation of the Proposed IMBO

To test the performance of the proposed IMBO, the small-size instances are solved by the mathematical model in Section 3.2 utilizing the Cplex solver, and the achieved results are also compared with that by the MBO and proposed IMBO. For each instance, the Cplex solver terminates when the optimal solution is obtained or computational time reaches 300 s (s), and the termination criterion of the MBO and IMBO is an elapsed computation time of $L \times J \times \rho$ milliseconds ($\rho = 10$). As MBO and IMBO are the stochastic methods due to the randomness involved in their nature, each test problem is solved for 10 times to evaluate the robustness of algorithms. Table 10 reports the detailed computational results for the tested instances by the Cplex and the algorithms. In this table, column 2 presents the problem size $L \times J$ (batches $\times$ charges). Columns 3 to 10 report the results by the Cplex solver, MBO and IMBO. Here, Best means the best value of the *RPD* values for each instance, Worst means the worst value of the *RPD* values for each instance and Avg means the average value of the *RPD* values for each instance in 10 repetitions. Note that the smaller *RPD* value denotes the better performance.

Table 10. Computational results by the Cplex, MBO and IMBO.

Instances	Problem Size	Cplex		MBO			IMBO		
		RPD	Time (s)	*RPD*			*RPD*		
				Best	Worst	Avg	Best	Worst	Avg
1	3 × 8	0.00	24.32	0.00	0.00	0.00	0.00	0.00	0.00
2	5 × 18	0.00	232.34	0.00	0.00	0.00	0.00	0.00	0.00
3	6 × 19	1.90	300.00	0.00	0.24	0.00	0.00	0.00	0.00
4	7 × 21	3.63	300.00	0.00	0.00	0.00	0.00	0.00	0.00
5	8 × 24	6.13	300.00	0.20	1.48	0.59	0.00	0.49	0.10
6	10 × 30	4.33	300.00	0.55	1.81	0.79	0.00	0.55	0.08

It is observed from Table 10 that all methods obtain the optimal solutions when the problem size (batches × charges) is 3 × 8 or 5 × 18. As the problem size becomes the larger, the achieved solutions by MBO and IMBO are significantly better than those by the Cplex. It is also clear that the achieved solutions by the proposed IMBO are smaller than those by MBO for the large-size instances. This finding suggests that the IMBO outperforms the MBO and demonstrates the efficiency of the proposed improvements. In summary, this comparative study demonstrates that the proposed IMBO outperforms the Cplex and MBO, and the improvements enhance the IMBO by a significant margin.

5.4. Comparative Study Among Algorithms

To further evaluate the performance of the proposed IMBO, the results by the IMBO algorithm are compared with that by seven other meta-heuristics. Additionally, the termination criterion of an elapsed computation time of L × J × ρ milliseconds is utilized, where L and J denote the number of batches and the number of charges, respectively. To observe the performances of the algorithms under different computation time, ρ is set as 10 and 20. All the algorithms solve all the 40 tested instances for 10 times, and namely a total number of 6400 data are achieved. Tables 11 and 12 present the detailed results by these algorithms when ρ = 10 and ρ = 20, where Min and Avg are the minimum value and the average value of the *RPD* values in 10 repetitions.

From Table 11, it is observed that the proposed IMBO outperforms the GA, GA_R, MA, PSO, SA, DTLBO and DABC for most of the instances. Specifically, all the algorithms obtain the current best solutions for small-size instances. For the medium-size instances from case 11 to case 20, all the results by IMBO are better than or equal to those by other compared algorithms in terms of minimum *RPD* and average *RPD*. For the large-size instances, all the results by IMBO are better than those by GA, GA_R, MA, PSO, SA, and DTLBO. Specifically, IMBO outperforms the compared methods in terms of minimum *RPD* for all the 40 instances and IMBO outperforms those methods for 34 out of 40 instances in terms of average *RPD*. This indicates the superiority of IMBO over these compared algorithms. From Table 12, similar findings can be achieved. Clearly, the proposed IMBO outperforms the other algorithms of almost all the tested instances in terms of minimum *RPD* and average *RPD*.

Table 11. Comparison results for minimum *RPD* and average *RPD* when ρ = 10.

Instances	GA		GA_R		MA		PSO		SA		DTLBO		DABC		IMBO	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Case 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
Case 11	0.00	0.23	0.00	0.10	0.00	0.00	0.00	0.11	0.00	0.21	0.00	0.23	0.00	0.00	0.00	0.00
Case 12	0.00	0.43	0.00	0.37	0.00	0.18	0.00	0.25	0.00	0.59	0.31	0.75	0.00	0.23	0.00	0.00
Case 13	0.00	0.37	0.00	0.05	0.09	0.11	0.00	0.08	0.00	0.31	0.00	0.22	0.00	0.06	0.00	0.00
Case 14	0.31	0.59	0.00	0.26	0.33	0.18	0.12	0.18	0.31	0.37	0.31	0.53	0.00	0.09	0.00	0.00
Case 15	0.00	0.65	0.00	0.28	0.00	0.30	0.27	0.31	0.00	0.66	0.00	0.50	0.00	0.24	0.00	0.00
Case 16	0.53	1.41	0.67	1.29	0.55	0.51	0.00	0.48	0.53	1.41	1.06	1.68	0.00	0.36	0.00	0.33
Case 17	0.52	1.16	1.02	1.19	1.16	1.05	0.67	0.91	1.10	1.29	0.94	1.25	0.47	0.82	0.00	0.43
Case 18	0.63	1.05	0.84	1.00	0.88	0.92	0.86	1.03	0.84	1.35	0.58	1.28	0.05	0.48	0.00	0.35
Case 19	0.31	0.53	0.26	0.39	0.33	1.00	0.90	1.06	0.31	0.51	0.31	0.57	0.26	0.30	0.00	0.28
Case 20	0.89	1.36	0.35	0.90	0.94	0.90	0.80	0.94	0.89	1.33	1.47	1.62	0.00	0.29	0.00	0.26
Case 21	0.78	1.10	1.22	1.47	1.00	0.55	0.49	0.58	0.95	1.33	1.03	1.49	0.12	0.57	0.00	0.54
Case 22	0.78	1.20	0.73	0.95	1.08	0.30	0.27	0.31	1.03	1.31	0.82	1.27	0.49	0.73	0.00	0.62
Case 23	0.21	1.02	1.18	1.31	1.04	1.40	1.25	1.47	0.99	1.44	1.20	1.47	0.08	0.68	0.00	0.44
Case 24	0.41	0.70	0.48	0.69	0.73	0.98	0.88	1.03	0.70	1.04	0.49	0.94	0.21	0.39	0.00	0.37
Case 25	0.20	0.57	0.52	0.78	0.60	0.78	0.70	0.82	0.57	1.02	0.61	1.02	0.16	0.27	0.00	0.29
Case 26	0.34	0.93	1.22	1.45	1.23	1.14	1.02	1.20	1.17	1.64	1.01	1.67	0.00	0.26	0.00	0.17
Case 27	0.90	1.25	0.71	1.26	1.34	0.47	0.42	0.49	1.27	1.70	1.58	1.97	0.00	0.33	0.00	0.37
Case 28	0.67	1.28	1.19	1.49	1.20	0.58	0.52	0.61	1.14	1.46	1.14	1.62	0.07	0.37	0.00	0.18
Case 29	0.60	1.20	1.04	1.32	1.75	0.96	0.86	1.01	1.67	1.91	1.14	1.70	0.00	0.28	0.00	0.10
Case 30	0.80	1.35	1.10	1.35	0.85	1.50	1.34	1.58	1.00	1.70	1.44	1.86	0.07	0.28	0.00	0.36
Case 31	0.59	1.11	0.88	1.10	0.93	1.09	0.97	1.14	1.13	1.51	1.10	1.58	0.13	0.37	0.00	0.38
Case 32	0.75	1.35	0.74	1.00	1.04	1.08	0.97	1.14	1.18	1.50	1.53	1.81	0.03	0.38	0.00	0.26
Case 33	0.70	1.03	0.70	1.05	1.07	1.37	1.22	1.44	1.02	1.45	1.29	1.56	0.21	0.46	0.00	0.35

Table 11. *Cont.*

Instances	GA		GA_R		MA		PSO		SA		DTLBO		DABC		IMBO	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Case 34	0.73	1.20	0.79	1.08	1.10	1.04	0.93	1.10	1.05	1.64	1.26	1.64	0.03	0.47	0.00	0.20
Case 35	0.62	0.94	0.77	1.00	1.24	1.45	1.30	1.53	1.18	1.34	0.83	1.28	0.05	0.28	0.00	0.44
Case 36	0.48	0.72	0.50	0.75	0.86	1.22	1.09	1.29	0.82	1.02	1.02	1.20	0.16	0.29	0.00	0.15
Case 37	0.52	0.77	0.29	0.64	0.69	1.20	1.07	1.26	0.66	0.99	0.86	1.13	0.00	0.27	0.00	0.24
Case 38	0.32	0.59	0.54	0.73	0.36	0.79	0.71	0.83	0.34	0.85	0.89	1.18	0.00	0.16	0.00	0.19
Case 39	0.30	0.64	0.65	0.78	0.74	0.97	0.87	1.02	0.70	0.87	0.59	0.96	0.02	0.13	0.00	0.14
Case 40	0.30	0.72	0.46	0.90	0.67	0.82	0.73	0.86	0.64	0.90	0.55	1.10	0.00	0.32	0.00	0.17
Avg	0.35	0.69	0.47	0.67	0.60	0.62	0.53	0.65	0.58	0.87	0.63	0.93	0.07	0.25	0.00	0.19

Note: The smallest values of *RPD* are the best.

Table 12. Comparison results for minimum RPD and average RPD when ρ = 20.

Instances	GA		GAR		MA		PSO		SA		DTLBO		DABC		IMBO	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Case 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Case 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
Case 11	0.00	0.13	0.00	0.10	0.00	0.00	0.00	0.11	0.00	0.12	0.00	0.23	0.00	0.00	0.00	0.00
Case 12	0.00	0.35	0.00	0.31	0.00	0.18	0.00	0.18	0.00	0.44	0.00	0.75	0.00	0.07	0.00	0.00
Case 13	0.00	0.19	0.00	0.05	0.00	0.05	0.00	0.08	0.00	0.06	0.00	0.22	0.00	0.00	0.00	0.00
Case 14	0.00	0.36	0.00	0.26	0.00	0.18	0.00	0.18	0.00	0.31	0.31	0.53	0.00	0.03	0.00	0.00
Case 15	0.00	0.17	0.00	0.15	0.00	0.30	0.16	0.21	0.00	0.34	0.00	0.46	0.00	0.03	0.00	0.00
Case 16	0.53	1.28	0.67	1.11	0.55	0.51	0.00	0.48	0.53	1.41	0.79	1.53	0.00	0.34	0.00	0.17
Case 17	0.52	1.16	1.00	1.11	1.16	1.05	0.67	0.91	1.10	1.29	0.79	1.20	0.37	0.60	0.00	0.43
Case 18	0.58	1.00	0.55	0.87	0.88	0.92	0.86	0.52	0.84	1.19	0.58	1.28	0.05	0.32	0.00	0.31

Table 12. *Cont.*

Instances	GA		GAR		MA		PSO		SA		DTLBO		DABC		IMBO	
	Min	**Avg**	**Min**	**Avg**	**Min**	**Avg**	**Min**	**Avg**	**Min**	**Avg**	**Min**	**Avg**	**Min**	**Avg**	**Min**	**Avg**
Case 19	0.31	0.44	0.26	0.38	0.33	0.42	0.40	0.29	0.31	0.46	0.26	0.48	0.00	0.28	0.00	0.24
Case 20	0.89	1.28	0.35	0.90	0.44	0.90	0.80	0.86	0.42	1.07	0.63	1.27	0.00	0.16	0.00	0.14
Case 21	0.46	1.10	0.43	1.17	1.00	0.55	0.49	0.41	0.95	1.33	1.03	1.49	0.12	0.57	0.00	0.25
Case 22	0.66	0.84	0.63	0.73	0.90	0.30	0.27	0.31	0.86	1.12	0.82	1.10	0.08	0.51	0.00	0.41
Case 23	0.21	1.02	0.35	1.02	1.04	1.40	1.07	0.93	0.99	1.44	1.20	1.47	0.08	0.68	0.00	0.34
Case 24	0.37	0.61	0.35	0.53	0.60	0.76	0.56	0.53	0.57	0.83	0.37	0.78	0.21	0.32	0.00	0.18
Case 25	0.20	0.57	0.23	0.51	0.60	0.78	0.54	0.42	0.57	0.93	0.61	0.89	0.12	0.25	0.00	0.23
Case 26	0.34	0.84	0.57	0.73	1.23	1.14	0.77	0.54	1.17	1.64	1.01	1.60	0.00	0.20	0.00	0.13
Case 27	0.64	1.10	0.61	0.96	0.88	0.47	0.42	0.49	0.84	1.49	1.07	1.53	0.00	0.33	0.00	0.24
Case 28	0.67	1.28	0.99	1.25	1.20	0.58	0.52	0.61	1.14	1.46	0.78	1.62	0.07	0.37	0.00	0.18
Case 29	0.60	1.04	0.64	0.90	1.30	0.96	0.86	0.81	1.24	1.57	1.14	1.64	0.00	0.28	0.00	0.10
Case 30	0.44	1.22	0.41	1.06	0.85	1.46	1.11	1.09	1.00	1.60	1.44	1.86	0.07	0.28	0.00	0.26
Case 31	0.59	0.98	0.64	0.86	0.93	1.09	0.90	0.91	1.05	1.31	1.10	1.42	0.00	0.36	0.00	0.34
Case 32	0.43	0.93	0.41	0.81	0.92	1.08	0.85	0.89	0.88	1.19	1.12	1.38	0.03	0.30	0.00	0.20
Case 33	0.59	0.86	0.56	0.75	0.87	1.13	0.79	0.83	0.83	1.24	0.94	1.44	0.11	0.32	0.00	0.35
Case 34	0.73	0.92	0.71	0.80	0.99	1.04	0.84	0.97	0.94	1.29	1.07	1.46	0.03	0.34	0.00	0.18
Case 35	0.62	0.94	0.71	0.87	0.96	1.07	0.91	1.07	0.91	1.18	0.83	1.28	0.05	0.26	0.00	0.13
Case 36	0.43	0.58	0.41	0.50	0.62	0.81	0.93	0.49	0.59	0.89	0.48	0.93	0.05	0.14	0.00	0.08
Case 37	0.32	0.60	0.29	0.52	0.36	0.69	0.85	0.69	0.34	0.76	0.80	0.98	0.00	0.25	0.00	0.22
Case 38	0.32	0.59	0.39	0.60	0.36	0.79	0.71	0.77	0.34	0.85	0.89	1.18	0.00	0.16	0.00	0.19
Case 39	0.30	0.62	0.32	0.54	0.74	0.85	0.57	0.65	0.70	0.87	0.59	0.96	0.02	0.13	0.00	0.14
Case 40	0.30	0.72	0.46	0.67	0.67	0.82	0.70	0.86	0.55	0.90	0.55	1.10	0.00	0.32	0.00	0.17
Avg	0.30	0.59	0.32	0.53	0.51	0.56	0.44	0.45	0.49	0.76	0.53	0.85	0.04	0.21	0.00	0.14

Note: The smallest values of *RPD* are the best.

To evaluate the algorithms' performance statistically, this study also conducts the ANOVA test to check whether the IMBO outperforms other algorithms statistically. Here, the algorithms type is the controlled factor and the average *RPD* is the response variable. Detailed ANOVA results are not exhibited for space limit, and instead this section depicts the means plots and 95% confidence intervals by the implemented algorithms in Figure 6 (ρ = 10) and Figure 7 (ρ = 20). From these figures, we can see clearly that the proposed IMBO outperforms other algorithms in terms of the minimum *RPD* and average *RPD* under two termination criteria. The statistical analysis shows that the improved migrating birds optimization algorithm is more efficient to solve the SCC scheduling problem.

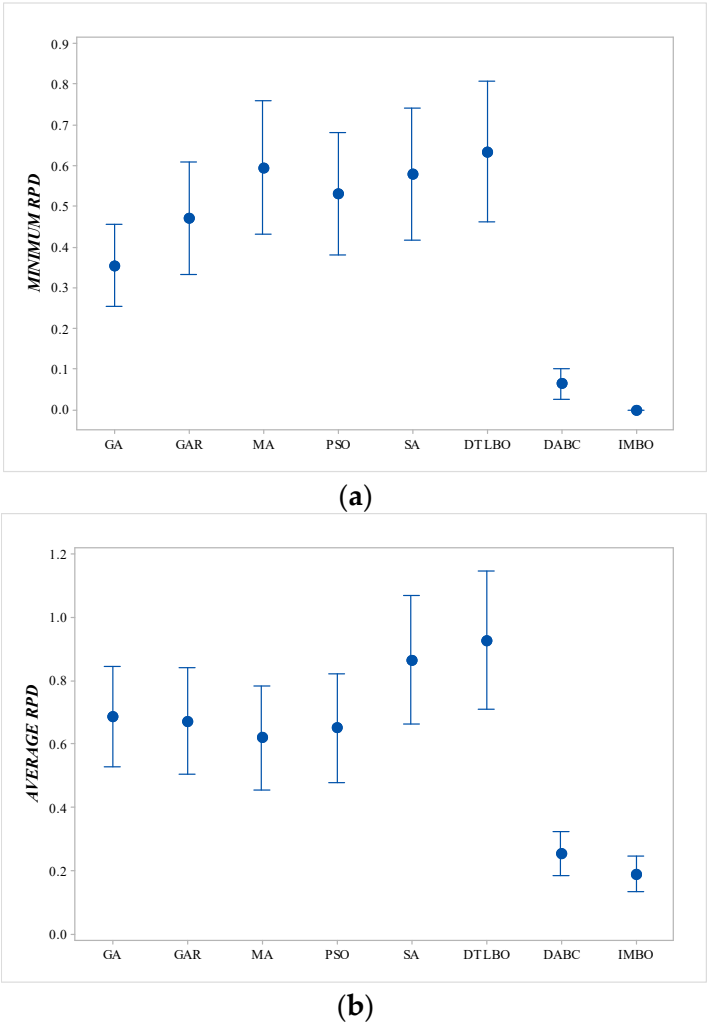
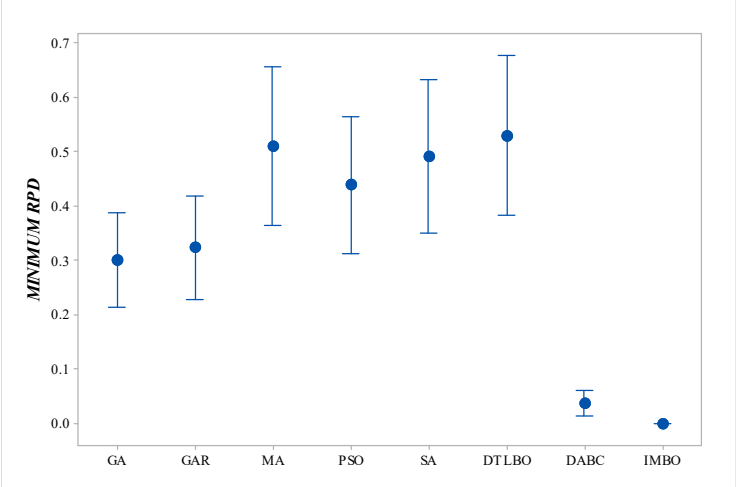
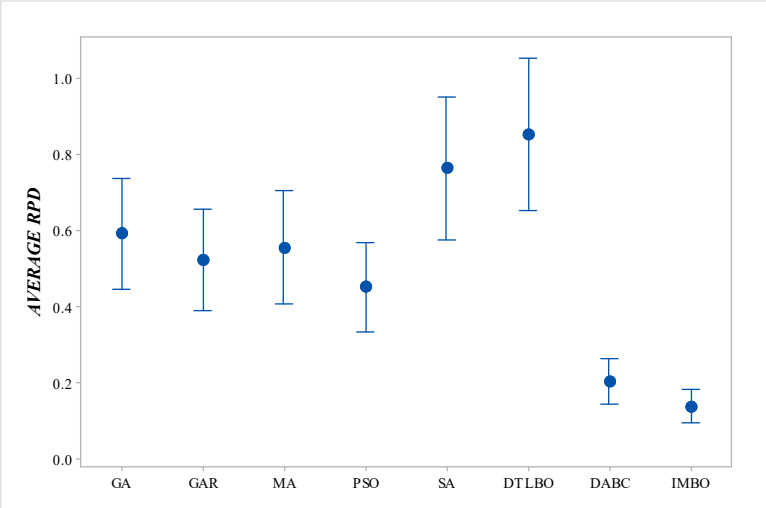


Figure 6. Means plots and 95% confidence intervals regarding minimum *RPD* or average *RPD* when ρ = 10. (**a**) The value of minimum *RPD* by algorithms; (**b**) The value of average *RPD* by algorithms.



(**a**)



(**b**)

Figure 7. Means plots and 95% confidence intervals regarding the minimum *RPD* or average *RPD* when $\rho = 20$. (**a**) The value of minimum *RPD* by algorithms; (**b**) The value of average *RPD* by algorithms.

In sum, the results demonstrate that the superiority of the proposed IMBO over the compared algorithms. In fact, the superiority of the IMBO is attributed to the optimization mechanism of the MBO and the developed improvements. Namely, the proposed IMBO utilizes two neighborhood structures to intensify exploration, the simulated annealing-based acceptance criterion to ensure the diversity of the population and the competitive mechanism emphasizes the exploitation capacity by searching around the most promising solution space. In summary, the computational studies demonstrate that the improvements enhance the IMBO by a significant margin and the proposed IMBO is a powerful algorithm for the considered SCC scheduling problem by outperforming all the compared algorithms.

6. Conclusions and Future Research

This study solves the realistic hybrid flow shop scheduling problem arising from the steelmaking and continuous-casting (SCC) process, namely the SCC scheduling problem with diverse products. To tackle this NP-hard problem, this study formulates a mixed-integer linear programming model to minimize the makespan and presents an improved migrating birds optimization algorithm (IMBO). The proposed IMBO utilizes two neighborhood structures to intensify exploration, develops a simulated

annealing-based acceptance criterion to enhance the diversity of the population, and employs a competitive mechanism to emphasize exploitation capacity. All these improvements help the IMBO algorithm to achieve the proper balance between exploration and exploitation. To evaluate the performance of the proposed algorithm, this study generates a set of 40 instances based on realistic production process in SCC plants. Experimental results demonstrate that the proposed IMBO is capable of achieving the optimal solutions for small-size instances, and outperforms the original migrating birds optimization algorithm and seven other reimplemented algorithms.

Future research might extend the considered problem by taking into account the processing time variation or machine breakdowns. In addition, since the transportation time is often limited by the carriers, the ladle pot and travelling crane, resulting to the complex uncertainties in the schedule process, considering the integrated problem of production scheduling and transportation process will effectively improve production continuity and reduce waiting time related costs. It is also suggested to consider the energy consumption or carbon emissions in steel plants. Meanwhile, since the developed IMBO produces competing performance, this algorithm can be extended to solve other scheduling problems.

Author Contributions: Conceptualization, D.H. and Q.T.; methodology, D.H. and Z.L.; software, D.H. and Z.Z.; validation, Q.T. and Z.L.; formal analysis, D.H.; investigation, D.H. and Z.Z.; resources, Q.T. and Z.L.; data curation, D.H.; writing—original draft preparation, D.H.; writing—review and editing, D.H.; visualization, D.H.; supervision, Z.L.; project administration, Q.T. and Z.L.; funding acquisition, Q.T. and Z.L. All authors have read and agree to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 51875421 and Grant 61803287.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A Detailed Two-Level Decoding Procedure

Algorithm A1 Detailed procedure of the FIFO-based forward heuristic

Input: Batch plan, charge permutation π(j) and the charge at position k in permutation {πk(j)}.
Step 1: Set the release time of each machine and each charge, $r_{i,m}$ and r_j, to be zero.
%Forward heuristic for first stage
Step 2: For the first stage
While ($k \leq$ J) do
2.1 Select charge $\pi_k(j)$ sequentially according to charge permutation $\pi(j)$;
2.2 Assign charge $\pi_k(j)$ to the earliest available machine m^*, where $r_{1,m^*} = \min_{m \in M_1} r_{1,m}$;
2.3 Schedule charge $\pi_k(j)$ on the assigned machine, set $ST_{1,\pi_k(j)} = max\left\{r_{1,m^*}, r_{\pi_k(j)}\right\}$ and update $r_{1,m^*} = ST_{1,\pi_k(j)} + PT_{1,\pi_k(j)}$.
Endwhile
Endfor
%Forward heuristic for refining stages
Step 3: For the next stages except the last stage ($i = 2, 3 \ldots s-1$)
If (the charge completes its task at the previous stage and is transferred to the current stage)
3.1 Select the first available machine (marked as refining machine m^*) or randomly select one if more than one machine is available simultaneously, where $r_{i,m^*} = \min_{m \in M_i} r_{i,m}$;
3.2 Set $ST_{i,\pi_k(j)} = max\left\{r_{s,m^*}, ST_{i-1,\pi_k(j)} + PT_{i-1,\pi_k(j)}\right\}$ and update $r_{1,m^*} = ST_{i,\pi_k(j)} + PT_{i,\pi_k(j)}$.
Endif
Endfor
%Forward heuristic for last stage

Step 4: For the last stage
While ($l \le L$) do
4.1 Select the earliest available caster for batch l, $m^* = \arg_{m \in M_s} \min\{r_{s,m}\}$;
4.2 Compute the casting start time of batch l,$Tsi_{s,LJ_{(l,ljf_{(l,j)})}} = \max\left\{Tf_{s-1,LJ_{(l,ljf_{(l,j)})}}, r_{s,m^*} + Su\right\}$, where r_{s,m^*} denotes the earliest available time of caster m^* on which batch l is processed;
4.3 Set $r = ljf_{(l,j)} + 1$;
Endwhile
4.4 While ($r \le lje_{(l,j)}$) do
4.4.1 Compute the casting start time of the remaining charges,
$Tsi_{s,LJ_{(l,r)}} = \max\left\{Tsi_{s,LJ_{(l,r-1)}} + PT_{s,LJ_{(l,r-1)}}, Tf_{s-1,LJ_{(l,j)}}\right\}$;
4.4.2 Set $r = r + 1$.
Endwhile
4.5 Set $l = l + 1$.
Endfor
Output: Machine assignment, charge permutation on each machine.

Algorithm A2 Detailed procedure of the backward heuristic

Input: Machine assignment, charge permutation on each machine
Step 1: Sort all the charges in the increasing order according to finish time at each stage and generate a charge list $\boldsymbol{\lambda}_m$ on each machine, where $|\boldsymbol{\lambda}_m|$ denotes the total number of charges on machine m;
%Backward heuristic for last stage
Step 2: While ($l \le L$) do
2.1 Calculate the start time of the last charge in batch l, $Tsi_{s,LJ_{(l,lje_{(l,j)})}} = \max\left\{Tsi_{s,LJ_{(l,lje_{(l,j)}-1)}} + PT_{s,LJ_{(l,lje_{(l,j)}-1)}}, Tf_{s-1,LJ_{(l,lje_{(l,j)})}}\right\}$, where $lje_{(l,j)}$ denotes the last charge in batch l processed on caster m^*.
2.2 Set $r = lje_{(l,j)} - 1$;
2.3 While ($r \ge ljf_{(l,j)}$) do
2.3.1 Adjust the start time of each charge to ensure it to be processed continuously in a batch $Tsi_{s,LJ_{(l,r)}} = Tsi_{s,LJ_{(l,r+1)}} - PT_{LJ_{(l,r)},m^*}$, update $r_{s,m^*} = Tsi_{s,LJ_{(l,r)}} + PT_{s,LJ_{(l,r)}}$;
2.3.2 Set $r = r - 1$.
Endwhile
2.4 Set $l = l + 1$.
Endwhile
%Backward heuristic for refining stages
Step 3: For the refining stages ($i = 2, 3 \ldots s - 1$)
Set $i =$ s;
While ($i > 2$) do
3.1 Adjust the start time of the last charge on each machine m by the right-shift method, $Tsi_{i-1,LJ_{(l,|\lambda_{ml}|)}} = Tsi_{i,LJ_{(l,|\lambda_{ml}|)}} - PT_{i,LJ_{(l,|\lambda_{ml}|)}}$;
3.2 Update $r_{i-1,m} = Tsi_{i-1,LJ_{(l,|\lambda_{ml}|)}}$;
3.3 set $j = |\boldsymbol{\lambda}_m| - 1$;
3.4 While ($j \ge 1$) do
3.5.1 Adjust the start time of the remaining charges on each machine m, $Tsi_{i-1,j} = \min\left\{r_{i-1,m}, Tsi_{i,j}\right\} - PT_{i,j}$ and update $r_{i-1,m} = Tsi_{i-1,j}$;
3.5.2 Set $j = j - 1$;
Endwhile
3.6 Set $i = i - 1$;
Endwhile
Endfor
%Backward heuristic for first stage

Step 4: For the first stage ($i = 1$)
4.1 Adjust the start time of the last charge processed on machine m by the right-shift method, set $Tsi_{1,LJ_{(l,|\lambda_{ml}|)}} = Tsi_{2,LJ_{(l,|\lambda_{ml}|)}} - PT_{1,LJ_{(l,|\lambda_{ml}|)}}$;
4.2 Update $r_{1,m} = Tsi_{1,LJ_{(l,|\lambda_{ml}|)}}$;
4.3 Set $j = |\lambda_m| - 1$;
4.4 While ($j \geq 1$) do
4.4.1 Adjust the start time of the remaining charges, $Tsi_{1,j} = \min\left\{r_{1,m}, Tsi_{2,j}\right\} - PT_{1,j}$ and update $r_{1,m} = Tsi_{1,j}$;
4.4.2 Set $j = j - 1$;
Endwhile
Endfor
Output: Production schedule

References

1. Rahman, H.F.; Sarker, R.; Essam, D. Multiple-order permutation flow shop scheduling under process interruptions. *Int. J. Adv. Manuf. Technol.* **2018**, *97*, 2781–2808. [CrossRef]
2. Zhao, F.; Xue, F.; Zhang, Y.; Ma, W.; Zhang, C.; Song, H. A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization. *Appl. Intell.* **2019**, *49*, 3362–3382. [CrossRef]
3. Jiang, T.; Zhang, C.; Zhu, H.; Gu, J.; Deng, G. Energy-Efficient Scheduling for a Job Shop Using an Improved Whale Optimization Algorithm. *Mathematics* **2018**, *6*, 220. [CrossRef]
4. Yang, J.P.; Wang, B.L.; Liu, Q.; Guan, M.; Li, T.K.; Gao, S.; Wei, D.G.; Liu, Q. Scheduling Model for the Practical Steelmaking-continuous Casting Production and Heuristic Algorithm Based on the Optimization of "Furnace-caster Matching" Mode. *ISIJ Int.* **2020**, *60*, 1213–1224. [CrossRef]
5. Jiang, S.-L.; Zheng, Z.; Liu, M. A preference-inspired multi-objective soft scheduling algorithm for the practical steelmaking-continuous casting production. *Comput. Ind. Eng.* **2018**, *115*, 582–594. [CrossRef]
6. Xu, Z.; Zheng, Z.; Gao, X. Energy-efficient steelmaking-continuous casting scheduling problem with temperature constraints and its solution using a multi-objective hybrid genetic algorithm with local search. *Appl. Soft Comput.* **2020**, *95*, 106554. [CrossRef]
7. Pan, Q.K. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *Eur. J. Oper. Res.* **2016**, *250*, 702–714. [CrossRef]
8. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]
9. Guirchoun, S.; Martineau, P.; Billaut, J.-C. Total completion time minimization in a computer system with a server and two parallel processors. *Comput. Oper. Res.* **2005**, *32*, 599–611. [CrossRef]
10. Oz, D. An improvement on the Migrating Birds Optimization with a problem-specific neighboring function for the multi-objective task allocation problem. *Expert Syst. Appl.* **2017**, *67*, 304–311. [CrossRef]
11. Meng, T.; Pan, Q.K.; Li, J.Q.; Sang, H.Y. An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem. *Swarm Evol. Comput.* **2018**, *38*, 64–78. [CrossRef]
12. Ferraro, A.; Rossit, D.; Toncovich, A.; Frutos, M. Lot Streaming Flow Shop with a Heterogeneous Machine. *Eng. Manag. J.* **2019**, *31*, 113–126. [CrossRef]
13. Gmys, J.; Mezmaz, M.; Melab, N.; Tuyttens, D. A computationally efficient Branch-and-Bound algorithm for the permutation flow-shop scheduling problem. *Eur. J. Oper. Res.* **2020**, *284*, 814–833. [CrossRef]
14. Hakeem-Ur-Rehman; Wan, G.; Zhan, Y. Multi-level, multi-stage lot-sizing and scheduling in the flexible flow shop with demand information updating. *Int. Trans. Oper. Res.* **2019**, 1–27. [CrossRef]
15. Sun, L.; Jin, H.; Yu, Y.; Li, Z.; Xi, J. Research on Steelmaking-Continuous Casting Production Scheduling Problem Based on Augmented Lagrangian Relaxation Algorithm under Multi-Coupling Constraints. *IFAC-PapersOnLine* **2019**, *52*, 820–825. [CrossRef]
16. Cui, H.J.; Luo, X.C.; Wang, Y. Scheduling of steelmaking-continuous casting process using deflected surrogate Lagrangian relaxation approach and DC algorithm. *Comput. Ind. Eng.* **2020**, *140*, 106271. [CrossRef]
17. Kim, S.-i.; Han, J.; Lee, Y.; Park, E. Decomposition based heuristic algorithm for lot-sizing and scheduling problem treating time horizon as a continuum. *Comput. Oper. Res.* **2010**, *37*, 302–314. [CrossRef]

18. Slotnick, S.A. Optimal and heuristic lead-time quotation for an integrated steel mill with a minimum batch size. *Eur. J. Oper. Res.* **2011**, *210*, 527–536. [CrossRef]
19. Bellabdaoui, A.; Teghem, J. A mixed-integer linear programming model for the continuous casting planning. *Int. J. Prod. Econ.* **2006**, *104*, 260–270. [CrossRef]
20. Harjunkoski, I.; Grossmann, I.E. A decomposition approach for the scheduling of a steel plant production. *Comput. Chem. Eng.* **2001**, *25*, 1647–1660. [CrossRef]
21. Tang, L.; Liu, J.; Rong, A.; Yang, Z. A mathematical programming model for scheduling steelmaking-continuous casting production 1. *Eur. J. Oper. Res.* **2000**, *120*, 423–435. [CrossRef]
22. Peng, J.G.; Liu, M.Z.; Zhang, X.; Ling, L. Hybrid heuristic algorithm for multi-objective scheduling problem. *J. Syst. Eng. Electron.* **2019**, *30*, 327–342. [CrossRef]
23. Hauber, W. A scheduling system for the steelmaking-continuous casting process. A case study from the steel-making industry. *Int. J. Prod. Res.* **2009**, *47*, 4147–4172.
24. Buyukozkan, K.; Kucukkoc, I.; Satoglu, S.I.; Zhang, D.Z. Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters. *Expert Syst. Appl.* **2016**, *50*, 151–166. [CrossRef]
25. Meng, T.; Pan, Q.-K.; Sang, H.-Y. A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations. *Int. J. Prod. Res.* **2018**, *56*, 5278–5292. [CrossRef]
26. Peng, K.K.; Pan, Q.K.; Zhang, B. An improved artificial bee colony algorithm for steelmaking-refining-continuous casting scheduling problem. *Chin. J. Chem. Eng.* **2018**, *26*, 1727–1735. [CrossRef]
27. Atighehchian, A.; Bijari, M.; Tarkesh, H. A novel hybrid algorithm for scheduling steel-making continuous casting production. *Comput. Oper. Res.* **2009**, *36*, 2450–2461. [CrossRef]
28. Li, J.-Q.; Pan, Q.-K. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Inf. Sci.* **2015**, *316*, 487–502. [CrossRef]
29. Han, Y.; Li, J.-Q.; Gong, D.; Sang, H. Multi-objective migrating birds optimization algorithm for stochastic lot-streaming flow shop scheduling with blocking. *IEEE Access* **2018**, *7*, 5946–5962. [CrossRef]
30. Niroomand, S.; Hadi-Vencheh, A.; Şahin, R.; Vizvári, B. Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Syst. Appl.* **2015**, *42*, 6586–6597. [CrossRef]
31. Zhang, Z.; Tang, Q.; Li, Z.; Zhang, L. Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. *Int. J. Prod. Res.* **2018**, *57*, 5520–5537. [CrossRef]
32. Zhang, Z.; Tang, Q.; Han, D.; Li, Z. Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment. *Neural Comput. Appl.* **2018**, *31*, 7501–7515. [CrossRef]
33. Li, J.; Xiao, X.; Tang, Q.; Floudas, C.A. Production Scheduling of a Large-Scale Steelmaking Continuous Casting Process via Unit-Specific Event-Based Continuous-Time Models: Short-Term and Medium-Term Scheduling. *Ind. Eng. Chem. Res.* **2012**, *51*, 7300–7319. [CrossRef]
34. Shaik, M.A.; Floudas, C.A. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. *Comput. Chem. Eng.* **2008**, *32*, 260–274. [CrossRef]
35. Verderame, P.M.; Elia, J.A.; Li, J.; Floudas, C.A. Planning and Scheduling under Uncertainty: A Review Across Multiple Sectors. *Ind. Eng. Chem. Res.* **2010**, *49*, 3993–4017. [CrossRef]
36. Gao, L.; Pan, Q.-K. A shuffled multi-swarm micro-migrating birds optimizer for a multi-resource-constrained flexible job shop scheduling problem. *Inf. Sci.* **2016**, *372*, 655–676. [CrossRef]
37. Janardhanan, M.N.; Li, Z.X.; Bocewicz, G.; Banaszak, Z.; Nielsen, P. Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times. *Appl. Math. Model.* **2019**, *65*, 256–270. [CrossRef]
38. Lin, S.; Luan, F.J.; Han, Z.H.; Lu, X.S.; Zhou, X.F.; Liu, W. Genetic Algorithm Based on Duality Principle for Bilevel Programming Problem in Steel-making Production. *Chin. J. Chem. Eng.* **2014**, *22*, 742–747. [CrossRef]
39. Tseng, L.-Y.; Lin, Y.-T. A hybrid genetic algorithm for no-wait flowshop scheduling problem. *Int. J. Prod. Econ.* **2010**, *128*, 144–152. [CrossRef]
40. Deng, J.; Wang, L.; Wang, S.-Y.; Zheng, X.-L. A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem. *Int. J. Prod. Res.* **2016**, *54*, 3561–3577. [CrossRef]
41. Behnamian, J. Diversified particle swarm optimization for hybrid flowshop scheduling. *J. Optim. Ind. Eng.* **2019**, *12*, 107–119.

42. Aghajani, M.; Ghodsi, R.; Javadi, B. Balancing of robotic mixed-model two-sided assembly line with robot setup times. *Int. J. Adv. Manuf. Technol.* **2014**, *74*, 1005–1016. [CrossRef]
43. Li, J.Q.; Pan, Q.K.; Mao, K. A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Eng. Appl. Artif. Intell.* **2015**, *37*, 279–292. [CrossRef]