Research Article

A Metaheuristic Method for the Task Assignment Problem in Continuous-Casting Production

Hongtao Hu ¹, Yiwei Wu,² and Tingsong Wang ³

¹College of Logistics Engineering, Shanghai Maritime University, Shanghai 201306, China

²School of Management, Shanghai University, Shanghai 200444, China

³School of Economics and Management, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Tingsong Wang; emswangts@whu.edu.cn

Received 2 July 2018; Accepted 14 September 2018; Published 2 October 2018

Guest Editor: Xinchang Wang

Copyright © 2018 Hongtao Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The steelmaking and continuous-casting (SCC) process in integrated iron and steel enterprises can be described as two stages: the upstream stage and downstream stage. Raw materials are transformed into molten steel in the upstream stage, while the downstream stage is responsible for transforming molten steel which is released at regular intervals and has a limited time for being turned into slabs. This article focuses on the task assignment problem in the downstream stage within the given information resulting from the upstream stage. This problem is formulated as a nonlinear mixed-integer programming model aimed at minimizing total tardiness within the resource constraints and time windows constraints for the tasks. An improved solution algorithm based on particle swarm optimization is developed to efficiently solve the proposed model. Finally, computational experiments are implemented to evaluate the performance of the solution algorithm in terms of solution quality and computational time.

1. Introduction

The iron and steel industry, as one of the most important economy sectors, is a large component of energy consumption and pollution emission. The energy consumption of iron and steel industry accounts for 10%–15% of the total energy consumption in the world, and the CO₂ emission accounts for about 7% of the total anthropogenic CO₂ emission [1]. Therefore, the iron and steel industry is an important study object because of the goal of emission reduction and energy saving. There are many scheduling tasks to be completed in the iron and steel industry. Because steelmaking and continuous-casting (SCC) process contributes the largest portion of the material loss in production and the SCC has important effects on the shape and quality of integrated iron and steel enterprises' final products (slabs), scheduling of SCC process is one of the most important tasks among different scheduling tasks [2]. Hence, studying the effective scheduling methods solving SCC problems has timely significance for improving the iron and steel industry productivity [3]. Figure 1 illustrates the material flow of an integrated iron and steel enterprise, which is a typical process industry, where final products are

produced by a sequence of processing operations (steelmaking, refining, and continuous-casting in sequence) to transform raw materials into solid slabs. The process can be described as two stages: an upstream stage and downstream stage. In the upstream stage, raw material is first turned into molten steel by a LDVC in order to initially reduce impurity content, which is called the steelmaking stage. And then, impurities are further eliminated by some refining techniques, such as RH and LF, which is called the refining stage. After the process of the upstream stage, the molten steel comes to the downstream stage, also called the continuous-casting stage, in which casting machines cast the molten steel into solid slabs uninterruptedly; the production process ends.

In the upstream stage, molten steel is produced in boilers. In order to save energy, upstream machines produce molten steel at regular intervals, which means that when boilers complete the production of a batch of molten steel, boilers continue to produce again rather than be cooled because it would require a lot of energy to restart boilers. Therefore, when upstream machines finish the production of molten steel, molten steel (upstream resource) is released immediately to the casting stage at regular intervals to service the casting machines processing of tasks.

FIGURE 1: Steelmaking and continuous-casting process.

The temperature of molten steel supplied has very important effects on the quality of continuous-casting slabs. In order to prevent nozzle clogging caused by bad fluidity of low temperature molten steel, the continuous-casting system requires strict control of the temperature of molten steel. Even worse, low temperature molten steel can lead to freezing of nozzles, which would subsequently interrupt the casting. Therefore, once molten steel is released, it must be used within the resource available time, or the remaining resource will be cold and idle.

In this article, we focus on the casting plans of the continuous-casting process within given information resulting from the upstream stages of steelmaking and refining. The casting plan is to make a decision on how to distribute all tasks to casting machines while satisfying all constraints.

Our work is motivated by a bottleneck problem arising in the steelmaking and the continuous-casting stage of steel production. Due to lower production capacity, the SCC production is always regarded as a bottleneck and restricts enterprises' development. In steel production, molten steel is supplied to parallel casting machines in the continuous-casting stage, where every task needs its own amount of resources, and the total resource demanded does not exceed the resource supplied at any point in time. Therefore, during the whole processing time, resources are released at regular intervals, where the resource quantity at each resource release time is fixed.

The contributions of this article are as follows:

- (1) This study takes account of some realistic factors such as the resource available time for use, nonlinear supply resource, nonrenewable resource, and tardiness minimization. There are related studies that considered some of these factors, but few previous studies have covered all of these factors.
- (2) We formulate the task assignment problem as a nonlinear mixed-integer programming model and propose an efficient solution algorithm to solve the proposed model which combines the particle swarm optimization algorithm and artificial bee colony algorithm.

The remainder of this article is organized as follows. Section 2 provides an overview of related work. Section 3

describes the problem in detail and presents an integer programming model for the problem. In Section 4 we present an iterated local search approach for solving the problem. Section 5 presents computational experiments. Final remarks are given in Section 6.

2. Literature Review

Because the scheduling of SCC process plays an important role in the iron and steel industry, there exist numerous related studies on the SCC process. We review two streams of related literature on SCC scheduling problems. Readers interested in overviews can refer to Tang et al. [4], Tang et al. [5], Tang et al. [6], Pan [3], and Mori and Mahalec [2] to obtain a comprehensive overview on the scheduling of SCC process.

The first research stream is related to the task assignment problem (TAP). The TAP problem is a challenging combinatorial optimization problem which aims to assign different tasks to some distributed processors in a system in order to increase the effectiveness of the system and minimize a certain cost. The beginning of the development of practical solution methods solving the classic TAP is proposed by Kuhn [7]. Numerous solving methods have been presented by scholars for the generalized TAP which aims to find the minimum assignment cost. Studying the problem of assigning drivers to cover tasks with service time windows and uncertain task durations, Cheung et al. [8] propose an adaptive labelling solution procedure which incorporates various practical constraints and work rules. Yan and Tang [9] develop a heuristic approach which is embedded in a framework aimed for stochastic airport gate assignment under uncertain flight delays. Zhen et al. [10] provide a stochastic programming model and a robust optimization model to deal with task workload requirements. Zhen et al. [11] study a strategic-level decision problem on assigning production stages to minimize the fixed cost and the expected value costs under stochastic demands of customers. Task assignment in this article also belongs to the parallel machine scheduling problem which is one of the classical problems in production systems.

The second topic of related works is concerned with the parallel machine scheduling (PMS). The PMS problem is a relatively simple multimachine scheduling problem in

actual production and is applied in a very wide range of production, so it is studied by many researchers. According to the statistical data through the web of science (SCIT + CPCI), there are about 160,000 articles related on the study of the parallel machines scheduling problem, and the number increases every year. Although there have been many researchers developing the parallel machine theory and practical applications, PMS still attracts more and more researchers' interests. There are three main research directions related to the parallel machine problem now: the optimization of the parallel machine scheduling problem whose goal is to minimize total tardiness or makespan, the parallel machine scheduling problem considering a product family, machine processing equipment, etc., and the parallel machine scheduling problem with resource constraints. This article studies the parallel machine scheduling problem with resource constraints which aims to minimize total tardiness.

Many researchers have studied PMS problems and applied them to a variety of manufacturing cases due to their theoretical and practical significance, for example, in the semiconductor manufacturing industry [12] and the electronic industry [13]. However, machines are the only resource considered in most PMS studies. In most real-life manufacturing systems, additional resources, such as automated guided vehicles, machine operators, tools, pallets, and industrial robots, are required for some jobs [14].

The first systematic approach to PMS problems was proposed in the mid-1950s, which was further extended from single machine scheduling problems. Since then, thousands of articles have investigated this problem. Cheng and Sin [15] provide a comprehensive analysis of PMS. In the 1990s, many articles extended PMS problems. Allahverdi et al. [16] provide a comprehensive review of such problems, including dynamic and stochastic problem settings for single and parallel machines, flow shops, and job shops. Mokotoff [17] develops a new local search heuristic algorithm for parallel machine problems which minimizes the total weighted completion time. Pfund et al. [18] review and summarize the literature on nontraditional parallel machine scheduling problems.

Comprehensive surveys for the tardiness minimization scheduling problem are provided by Dileepan [19] and Koulamsa [20]. Compared to Emmons et al. [21] who study tardiness in a single machine scheduling problem, Du and Leung [22] propose a better solution using different algorithms and describe the optimal solution of the problem. Lawler [23] proposes a decomposition algorithm that decomposes the problem into multiple noninterfering subproblems. Later, Shim and Kim [24] propose to use the branch-and-bound method to solve a single machine problem aimed at tardiness minimization. More recently, Lee and Kim [25] apply the branch-and-bound method to parallel machine scheduling and tardiness minimization. In addition, Mensendiek et al. (2015) develop a mathematical model for optimal sequences on parallel machines with fixed delivery dates and present heuristics to solve the problem.

Apart from this theoretical analysis, there are many researchers studying real problems related to industrial activities and considering the problems of resource constraints.

Blazewicz et al. [26] point out that job processing not only needs a machine, but also needs automatic guided vehicles, machine operators, tools, pallets, abrasives, industrial robots, and other additional resources. Blazewicz et al. [27] discuss the single machine scheduling problem with a continual resource and also discuss the nonrenewable resources proposed by Shabtay and Kaspi [14]. Grigoriv et al. (2005) consider the scheduling problem with raw material resource constraints, mainly with the following three constraints: each job has its own raw material, all jobs require a generic resource, and each job has many kinds of raw materials and has to choose one or some materials. The objective of different models in the above articles is to minimize the processing time, or minimize the maximum number of late jobs. Briskorn et al. [28] study the single machine scheduling problem aimed at minimizing the processing time or the maximum number of late jobs when the production process should stop if the resource demand cannot be met. Briskorn et al. [29] study the single machine scheduling problem with inventory constraints, and the goal is to minimize the total processing time. In order to solve the above problems, the branch-and-bound method and dynamic programming method are used to find the optimal solution. It was found that the heuristic algorithm outperforms a state-of-the-art commercial mixed integer programming solver even with 20 jobs. Györgyi et al. [30] study resource constraints in single machine scheduling, such as a nonrenewable raw material, where the goal is to minimize the total processing time. Wang et al. [31] study the PMS problem with two kinds of resource constraints (machines and molds), and the goal is to minimize the total processing time. It discusses preventive maintenance activities for the flexibility of resources.

In summary, many previous studies on SCC optimization problems disregarded the relationship between the PMS problem and the TSP problem. Although few studies have considered this relationship, they did not consider many realistic factors, such as the nonlinear supplied and required resource, and resource available time. Therefore, this article focuses on the scheduling problem with nonlinear required and consumed resource constraints aimed at minimization of total tardiness under parallel machines, and the production resource is a nonrenewable resource and has limited time for use.

This study conducts a comprehensive investigation of the SCC problem by considering several realistic factors, such as nonlinear supplied and required resource, and resource available time; these factors are related to the practical operation and production management in the iron and steel industry. In addition, an improved algorithm is also presented. In comparison to the existing literature, the model proposed in this study can provide more reasonable SCC scheduling plans in real-world application.

3. Problem Statement and Mathematical Model

3.1. Problem Description. The production scheduling problem studied in this article can be described as follows. In iron and steel enterprises, the main production and scheduling

problems are the furnace plans and casting plans. The furnace plan determines how many batches of molten steel should be produced in the upstream stage. The casting plan means the pouring order of all tasks which are distributed to the same casting machine. This article focuses on how to distribute casting machines in the casting stage, which implies that, in order to achieve multifurnace casting and reduce energy consumption, the continuous-casting process in an iron and steel enterprise should have each casting machine produce as many casting sets as possible. Therefore, the main problem to be solved in this article is to determine each task's casting machine and then determine the pouring order of each casting machine. Specifically, there are many tasks waiting to be processed by parallel casting machines. Tasks must be completed one by one because each machine can only process one task at a time. Additionally, each task requires a certain amount of common resource that is provided by the upstream stage.

The key of continuous-casting production schedules is the upstream resource constraints. An example of a production schedules is shown in Figure 2 which illustrates the implications of resource constraints over time. There are 6 tasks which should be processed by any one of two available machines in the continuous-casting process. At each resource's release time, a fixed amount of resource from the upstream stage will be released to service casting machines processing tasks within the resource available time. (1, *a*) in Figure 2 represents that task 1 is processed on machine *a*, and (2, *b*) represents that task 2 is processed on machine *b*. At time 0, upstream resources are released first. At time C_1 , task 1 is finished on machine *a*, so task 3 is processed on machine *a*, and the total resource requirement increases. At time C_3 where task 3 is finished on machine *a*, because the remaining resource is not enough to process another task, machine *a* will stop processing till the next resource release time. At time C_2 , task 2 is finished on machine *b*, but the remaining resource is not enough to process any more tasks, so machine *b* has to wait till the next resource release time, too. f is the fixed quantity of released resource at each resource release time. The cumulative amount of resources required cannot exceed the cumulative amount of resource supplied at any time, and both of them are nonlinear over time. We assume that each task consumes resources at the beginning of the process.

As can be seen from Figure 3, since the cumulative resource demanded cannot exceed the cumulative resource supplied at any time, tasks may have a waiting time when they arrive and cannot be processed, and the remaining resource at the last resource release time is idle. Besides, since the upstream resource has a limited time for use, it will also be idle because of low temperature. Let N denote the set of tasks to be processed by the parallel machines. Each task is characterized by its due date d_i , processing time p_i , and the quantity q_i of resource required by the machine to process the task. The resource is supplied in the fixed quantity f at each resource release time δ . The resource supplied can only be used within the resource available time $\emptyset$, because, after $\emptyset$ time, the resource will be cold and will not meet the temperature requirements. The goal of this article is to find the optimal solution for task assignments which considers

the resource constraints (resource release time and resource available time for use) and aims at minimization of total tardiness.

Before formulating the mathematical model for this problem, the assumptions of this study are clarified as follows:

- (1) Every machine can only process one task at any time and can be used if it is available during the whole production.
- (2) Resource is released and consumed at the beginning of each resource released cycle.
- (3) The preparation time is not necessary for all tasks.
- (4) The completion times of tasks are unknown. The given information is only each task's processing time, amount of resource required, and due date.
- (5) All tasks are independent and must be processed once without interruption.

3.2. Mathematical Model

Notation

0: virtual starting artifacts

d_i : the due date of task i

f : the fixed amount of the resource supplied each time

i : the index of a task, $i = 1, 2, \dots, |N|$, where $|N|$ is the total number of tasks

k : the index of a resource released time; there are $|K|$ times of resource release

m : the index of a machine, $m = 1, 2, \dots, |Q|$, where $|Q|$ is the total number of machines

M : a large positive number

$n + 1$: virtual ending artifacts, where n is the number of tasks

$N^- = N \cup \{0\}$

$N^+ = N \cup \{n + 1\}$

$\overline{N} = N^- \cup N^+$

p_i : the processing time of task i

q_i : the resource required by task i

δ : the resource release cycle

$\emptyset$: the resource availability time for use

Decision Variables

C_i : the completion time of task i

T_i : the tardiness of task i

X_{ijm} : a binary variable, which equals 1 if task j is scheduled to be processed immediately after task i on machine m ; it is 0, otherwise

Y_{ikm} : a binary variable, which equals 1 if task i is scheduled to be processed at the k th resource release time on machine m ; it is 0, otherwise

FIGURE 2: Resource released time constraints in continuous-casting production.

FIGURE 3: Resource available time constraints in continuous-casting production.

The problem can be formulated as the following mathematical model:

$$[M1] \min \sum_{i \in N} T_i \quad (1)$$

subject to

$$\sum_{m \in Q} \sum_{i \in N^+ (i \neq j)} X_{ijm} = 1 \quad j \in N \quad (2)$$

$$\sum_{m \in Q} \sum_{j \in N^+ (i \neq j)} X_{ijm} = 1 \quad i \in N \quad (3)$$

$$C_j > C_i + p_j - (1 - X_{ijm})M \quad (4)$$

$$i \in N^-, j \in N^+ (i \neq j), m \in Q$$

$$T_i \geq C_i - d_i \quad i \in \bar{N} \quad (5)$$

$$f \geq \sum_{m \in Q} \sum_{i \in N} q_i Y_{ikm} \quad k \in K \quad (6)$$

$$\sum_{m \in Q} \sum_{k \in K} Y_{ikm} = 1 \quad i \in N \quad (7)$$

$$\sum_{k' \in K(k' \geq k)} Y_{jk'm} \geq X_{ijm} Y_{ikm} \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (8)$$

$$\sum_{k \in K} Y_{ikm} \geq X_{ijm} \quad i \in N^-, j \in N^+ (i \neq j), m \in Q \quad (9)$$

$$\sum_{k \in K} Y_{jkm} \geq X_{ijm} \quad i \in N^-, j \in N^+ (i \neq j), m \in Q \quad (10)$$

$$\sum_{j \in N^-(i \neq j)} X_{jim} - \sum_{j \in N^+(i \neq j)} X_{ijm} = 0 \quad i \in N, m \in Q \quad (11)$$

$$\sum_{j \in N^+} X_{0jm} = 1 \quad m \in Q \quad (12)$$

$$\sum_{i \in N^-} X_{i(n+1)m} = 1 \quad m \in Q \quad (13)$$

$$C_i - p_i \geq \delta \sum_{m \in Q} \sum_{k \in K} k Y_{ikm} \quad i \in N \quad (14)$$

$$C_i \leq \delta \sum_{m \in Q} \sum_{k \in K} k Y_{ikm} + \theta \quad i \in N \quad (15)$$

$$X_{ijm} \in \{0, 1\} \quad i \in N^-, j \in N^+, m \in Q \quad (16)$$

$$Y_{ikm} \in \{0, 1\} \quad i \in \bar{N}, k \in K, m \in Q \quad (17)$$

$$T_i, C_i \geq 0 \quad i \in \bar{N} \quad (18)$$

Objective (1) is to minimize the total tardiness of all tasks. Constraints (2) and (3) ensure that tasks are finished one by one. Constraints (4) state that if task j is processed after task i immediately on machine m , the beginning processing time of task j is later than the completion time of task i . Constraints (5) restrict the tardiness of each task i . Constraints (6) are nonlinear supply resource constraints and ensure that the total resource supplied at any time is not less than the resource required in each resource release time. Constraints (7) ensure that each task is fulfilled exactly once. Constraints (8) ensure that if task j is processed after task i immediately on machine m , the index of task j 's resource release time is not less than that of task i . It is noted that Constraints (8) are nonlinear constraints and will be transformed to the linear form. Constraints (9) are the flow conservation constraint. Constraints (10) and (11) ensure that each machine begins to work with virtual task 0 and ends with virtual task $n + 1$. Constraints (12) and (13) ensure that if task i is processed at the k th resource released time, the starting processing time of task i is later than this resource release time, and it should be processed within the resource available time for use θ . Constraints (14) to (16) restrict the domain of the decision variables.

3.3. Linearization of Constraints (8). Constraints (8) contain a nonlinear part $X_{ijm} Y_{ikm}$, which is the product of two binary variables. We define a new binary variable φ_{ijkm} to replace the nonlinear part.

Newly Defined Variables. φ_{ijkm} binary variable is equal to one if and only if task j is processed after task i immediately on machine m , and the index of task i 's resource release time is k .

Constraints (8) become

$$\sum_{k' \in K(k' \geq k)} Y_{jk'm} \geq \varphi_{ijkm} \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (17)$$

In addition, some more constraints need to be defined so that the newly defined variable φ_{ijkm} can replace the function of $X_{ijm} Y_{ikm}$.

$$\varphi_{ijkm} \geq X_{ijm} + Y_{ikm} - 1 \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (18)$$

$$\varphi_{ijkm} \leq X_{ijm} \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (19)$$

$$\varphi_{ijkm} \leq Y_{ikm} \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (20)$$

$$\varphi_{ijkm} \in \{0, 1\} \quad i \in N^-, j \in N^+ (i \neq j), k \in K, m \in Q \quad (21)$$

Constraints (17) are linear and now the following [M2] is a MIP model, which can be solved by off-the-shelf solvers such as CPLEX.

[M2] An MIP model: Objective (1), subject to Constraints (2)–(7) and (9)–(21).

4. Solution Approach

4.1. PSO Algorithm. For some small-scale instances, the proposed model can be solved by using CPLEX solver directly. However, large-scale instances cannot be solved by CPLEX solver within a reasonable time, or can lead to an OutOfMemory error.

Particle swarm optimization (PSO) is a population-based optimization algorithm, where each particle has a position and velocity representing a solution. The position reflects the quality of the solution, and the velocity determines where the particle will move in the next iteration. The ordinary updating formulae for velocities and positions are as follows:

$$v_{ik}^{t+1} = v_{ik}^t + c_1 r_1 (x_{pbest_{ik}}^t - x_{ik}^t) + c_2 r_2 (x_{gbest_{ik}}^t - x_{ik}^t) \quad (22)$$

$$x_{ik}^{t+1} = x_{ik}^t + v_{ik}^t \quad (23)$$

where x_{ik}^{t+1} and x_{ik}^t represent the current and previous positions of particle i in dimension k , respectively; v_{ik}^{t+1} and v_{ik}^t represent the current and previous velocity of particle i in dimension k , respectively; the weights c_1 and c_2 are acceleration constants which are generally viewed as the cognitive

```

 $i \leftarrow 1$ 
 $k \leftarrow \text{maxEpoch}$ 
while  $i \leq k$  do
    decode the particle to generate genes, obtain the reduced MIP and solve it by using CPLEX
    compare and update each swarm's best position and particle's best position
    update the velocity and position accordingly
     $i \leftarrow i + 1$ 
end while

```

ALGORITHM 1: PSO algorithm main steps.

acceleration coefficient and social acceleration coefficient, respectively; r_1 and r_2 are random numbers in $[0, 1]$; $x_{pbest_{ik}}^t$ is the best position of particle i in dimension k up to iteration t ; and $x_{gbest_{ik}}^t$ is the best position of the whole swarm in dimension k up to iteration t .

The pseudocode of our PSO algorithm is shown in Algorithm 1. The approach starts by generating an initial solution, which can be any sequence of tasks. It then initializes an iteration counter and repeats the same setup until the iteration counter reaches a given limit. In each iteration, the approach selects the next neighborhood operator from a randomly generated sequence of operators. For each operator, all possible moves are examined. If an improved solution is found, this solution is accepted as the new incumbent solution. If no improvement can be obtained by any of the operators, a locally optimal solution is found. The search process is repeated in order to be able to escape from a local optimum. Before continuing with the next iteration, the incumbent solution is perturbed in order to obtain a new solution that potentially can be improved using the operators.

Apart from the main steps, this article also designs a new encoding and decoding method to improve solution quality. This encoding and decoding method is specific to X_{ijm} and is illustrated as follows. As Figure 4 shows, we have 5 tasks, and the number of available machines is 3. We first generate 5 random numbers in $[0, 1]$ as genes, then use the value of genes times the total number of machines, round down fractions, and obtain each task's processing machine. Then, we calculate the due date for each task and determine its urgency degree based on its due date. The task with an earlier due date is prior. We compare the urgency degrees of all tasks distributed to the same machine, and tasks are discharged from the processing sequence according to their degree of urgency. For example, as in Figure 4, tasks 2, 3, and 4 are arranged to be processed on machine 2, task 1 is arranged to machine 0, and task 5 is arranged to machine 1. Then, as Table 1 shows, each machine's processing order is arranged according to the degree of urgency. So, just as Table 2 shows, the processing order on machine 0 is task 1, the order on machine 1 is task 5, and the order on machine 2 is tasks 3, 2, and 4. Therefore, $X_{322} = X_{242} = 1$.

4.2. Algorithm Improvement. The neighborhood structure is a key component of the algorithmic framework, and neighborhood search has an effect on the performance of heuristic algorithm. Therefore, we design an improved neighborhood

search method for the PSO algorithm in order to improve solving ability. According to the experimental analysis of the example, we find that when CPLEX solves this problem directly, the time that the Gap value (percent) drops from 100% to 10% is very fast. In most cases, almost 80% of the whole computation time is spent on dropping the Gap value from 10% to 0. Therefore, it may need to take a long time to find the neighborhood of the optimal solution. And if the local search of the PSO algorithm does not find an improved solution, we employ another local search method to find it again. We use the behavior of bees in nature to explain the core idea of our improved algorithm more easily. During the searching process, a food source represents a candidate solution and the best food source found by bees is the model's optimal solution. The initial solution $x_i (i = 1, 2, \dots, SN, SN$ is the number of nectar source) of swarm is a vector of D dimension, where D is the number of the parameters to be optimized. Each x_i has n variables ($x_{id}, d = 1, 2, \dots, n$) and these variables will all be optimized to make the problem's objective minimal. So all the bees' first solution is generated by formula (24). In later iterations, scouts move to other solutions according to formula (25). And greedy selection is applied between the old solution and the new solution to find a better solution. After that, onlooker bees go out to exploit better solutions in their surroundings. Roulette wheel selection is applied to create further generation of new solutions. New solutions are updated by formula (26). When the number of iterations reaches the limit, the current global optimal solution is taken as the solution, and solution fitness is defined as the ABC algorithm's solution tardiness.

$$X_{id}^0 = L_d + r(U_d - L_d) \quad (24)$$

where X_{id} is the i th potential solution in the swarm, U_d and L_d are bounds of X_i in the d th dimension, and r is a random number in the interval $[0, 1]$.

$$\text{new_}X_i^j = X_i^j + \varphi(X_i^j - X_k^j) \quad (25)$$

where i is the current solution, $k (k \neq i)$ is a randomly selected solution among the SN solutions, and φ is a random number in the interval $[-1, 1]$.

$$X_{id}^t = L_d + r(U_d - L_d) \quad (26)$$

where X_{id}^t is the t th iteration solutions, U_d and L_d are bounds of X_i in the d th dimension, and r is a random number in the interval $[0, 1]$.

FIGURE 4: The encoding and decoding method.

TABLE 1: An example of urgency degree.

Index of tasks	Due date	Processed machine	Urgency degree
1	15	0	1
2	14	2	2
3	8	2	1
4	20	2	3
5	16	1	1

TABLE 2: An example of decoding results.

Index of machines	Task processing order
0	1
1	5
2	3-2-4

This local search method searches iteratively in a feasible neighborhood till an upper limit on time, and then it starts to search in another feasible neighborhood. So, too much time is spent doing local search, which leads to poor results. Therefore, this article conjectures that the improved local search method is more suitable for a local search to solve the proposed model. Considering the global search ability of the particle swarm algorithm, and the local search ability of the abovementioned method, this article proposes an improved algorithm, called PSOB, to improve the existing algorithm, accelerate the resolving speed, improve the resolving ability, and generate better solutions.

The pseudocode of the PSOB algorithm is shown in Algorithm 2. In order to find the optimal neighborhood near the optimal solution as soon as possible, the PSOB algorithm searches the solution globally quickly like the PSO algorithm. Then, it uses the abovementioned local search algorithm to do a local search. The PSOB algorithm's solving process is as follows. When the first part of the solving process is finished, the second part's initial global optimal position is set as the current GlobalMin and GlobalParas[i]. Then, the abovementioned local search algorithm is iterated. After maxCycle times, the final result is the PSOB algorithm's optimal solution.

TABLE 3: Parameter values of the model.

Parameter	Value	Parameter	Value	Parameter	Value
d_i	$U(10,60)$	f	15	δ	25
p_i	$U(2,15)$	q_i	$U(2,7)$	$\emptyset$	23

TABLE 4: Parameter values of PSO algorithm and the proposed algorithm.

Parameter	Value
$max\ iteration$	15
$tabu\ list\ size$	12
$N\ size$	10

5. Computational Experiments

This section presents the computational experiments conducted to evaluate the proposed algorithm. Several experiments are performed on a PC (Intel Core i7, 2.6 G Hz; Memory, 8 G). Each algorithm based heuristic is coded by using C#. In addition, Tables 3 and 4 list the parameter values of the proposed model, and PSO algorithm and the proposed algorithm, respectively.

The quality of solutions is affected by the number of machines m and the number of tasks i . Tables 5 and 6 summarize the small-scale instances and large-scale instances' computational experiments solved by CPLEX, the PSO algorithm method, and the PSOB algorithm method in terms of the average computation time (CPU) and the solution quality. Each set has ten randomly generated cases and we record their corresponding average values.


```

s ← initialsolution()
i ← 1, N ← 20, k ← 30,
while i ≤ k do
  repeat
    select neighborhood operator  $\varphi$ 
    for all solutions  $s'$  that can be generated by applying  $\varphi$  to  $s$  do
      if tardiness( $s'$ ) < tardiness( $s$ ) then
        s ←  $s'$ 
      end if
    end for
  until s is locally optimal for all operators
  s ← update()
  i ← i+1
end while
BestSolution ← GlobalMin, Solution[s] ← GlobalParas[i]
initialsolution(): iter ← 1, N ← 20, limit ← 30, i ← 1, maxCycle ← 20
k ← maxEpoch
while iter ≤ limit do
  scouts: decode the particle to generate genes, obtain the reduced MIP and solve it by
    CPLEX
  calculate fitness
  generate employed bees and onlooker bees: trail(i) ← 0
  select neighborhood operator  $\varphi$ 
  for all solutions  $s'$  that can be generated by applying  $\varphi$  to  $s$  do
    if tardiness( $s'$ ) < tardiness( $s$ ) then
      s ←  $s'$ 
      trial(i)=0
    else
      trial(i)=trail(i)+1
    end if
    calculate Probability(i)
    if Probability(i) > rand()
      employed bees: select neighborhood operator
      for all solutions  $s'$  that can be generated by applying  $\varphi$  to  $s$  do
        if tardiness( $s'$ ) < tardiness( $s$ ) then
          s ←  $s'$ 
          trial(i) ← 0
        else
          trial(i) ← trail(i)+1
        end if
      end if
    end if
    if trail(i) > limit
      scouts: generate the new food source
      update BestSolution
      iter ← iter+1
    end if
    if iter > maxCycle
      find BestSolution
    else
      generate employed bees and repeat
    end if
  end while
end while

```

ALGORITHM 2: PSOB algorithm main steps.

As Tables 5 and 6 show, the average computation time required by CPLEX increases sharply with an increase in the number of tasks. For the instance with 2 machines, the 20-task case is already very difficult to solve; the instances with 3 machines and 4 machines are very difficult to solve with

16 tasks and 18 tasks, respectively. Analyzing Tables 5 and 6, the total number of instances is 27, and the solutions of PSOB algorithm are closer to the exact solutions obtained by CPLEX than that of PSO algorithm in all 27 instances, including 7 instances with equal solutions. Besides, the computation

TABLE 5: Computational experiments of solving small-scale instances by CPLEX, PSO, and PSOB algorithms.

Number Of Machines	Number Of Tasks	CPLEX		PSO		PSOB	
		Best Solution	CPU(s)	Best Solution	CPU(s)	Best Solution	CPU(s)
2	6	1.2	0.1	1.2	17.3	1.2	17.2
	8	17.3	9.9	18.5	25.4	18.5	21.8
	10	43.5	9.0	51.5	38.2	43.5	27.4
	12	83.5	124.3	95.9	44.9	90.8	39.3
	14	118.0	702.1	151.5	49.8	143.2	52.3
Average		53.0	169.0	64.0	35.0	59.0	32.0
3	8	10.7	2.5	10.7	68.3	10.7	43.6
	10	39.1	6.6	39.1	171.9	39.1	84.8
	12	76.2	42.6	76.5	359.8	76.3	344.9
	14	101.0	149.7	149.9	2191.7	101.0	1067.7
	16	318.3	424.3	329.7	2658.7	321.0	1499.1
Average		109.1	125.1	121.2	1090.1	109.6	608.0
4	10	39.0	9.9	39.0	222.5	39.0	199.0
	12	75.9	56.5	75.9	765.6	75.9	759.1
	14	116.0	206.8	116.8	1476.6	116.0	1376.6
	16	318.3	792.7	323.3	2462.7	318.3	2162.7
	18	324.8	2036.0	331.3	2316.0	330.0	2176.8
Average		174.8	620.4	177.3	1448.7	175.8	1334.8

TABLE 6: Computational experiments of solving large-scale instances by CPLEX, PSO, and PSOB algorithms.

Number Of Machines	Number Of Tasks	CPLEX		PSO		PSOB	
		Best Solution	CPU(s)	Best Solution	CPU(s)	Best Solution	CPU(s)
2	20	575	3600	598.0	105.0	584.0	104.0
	22	643727	3600	867.7	116.3	820.3	469.0
	25	607637	3600	1345.0	202.3	894.3	263.3
	30	16612880	3600	2218.0	347.3	1990.0	337.3
Average		-	3600	1257.2	192.7	1072.2	293.4
3	20	382432	3600	465.0	3042.0	410.7	2719.0
	22	629671	3600	708.7	2965.0	697.8	2778.9
	25	10491201	3600	1128.0	3234.3	1101.3	3008.9
	30	-	3600	2010.7	3517.0	1050.3	3379.1
		-	3600	1078.1	3189.6	815.0	2971.5
4	20	390400	3600	463.0	3354.0	424.0	3155.2
	22	635660	3600	689.7	3248.7	689.7	3145.4
	25	10341439	3600	1175.0	3440.0	1095.0	3456.0
	30	-	3600	3972.0	3549.2	2477.4	3600.0
Average		-	3600	1574.9	3398.0	1171.5	3339.2

Note. The en dash means we did not find any solution within the time limits. Two values of best solutions solved by CPLEX are the lower bound and upper bound of the corresponding instance.

time by the PSO algorithm is longer than that of the PSOB algorithm. Therefore, the computational experiments show that the PSOB algorithm outperforms the PSO algorithm in terms of both solution quality and computation time.

In total, the heuristic algorithms are able to find equally good solutions for small-scale instances and can find better

solutions than the CPLEX solver within a reasonable time. Tables 5 and 6 show that the PSOB algorithm's solutions are closer to CPLEX's solution than those of the PSO algorithm. Therefore, this article argues that, in the continuous-casting stage of iron and steel enterprises, the distribution of casting machines and casting molten steel tundish processing order

problems based on upstream resource constraints are more suitable to be solved by the PSOB algorithm for large-scale problems.

6. Conclusions

The steelmaking and continuous-casting problems have been extensively studied in literature and widely adopted in industry. However, most studies only consider a single machine which has a linear resource supplied. In practice, the resource supply rate and consumption rate are nonlinear, and the available time for use of supplied resource is limited. Accordingly, we assume that the upstream resource is released at regular intervals and has limited time for use. The release and consumption of the resource happen at the beginning of each resource release cycle.

This article studies task assignment for continuous-casting production in integrated iron and steel enterprises. This problem differs from other similar problems without resource constraints because the optimal solution can be influenced by the resource constraints. The distribution of casting machine and casting molten steel tundish processing order problems are important planning problems for daily operational management in iron and steel enterprises. In order to deal with these problems efficiently, we present a mathematical model minimizing total tardiness and an improved heuristic algorithm to generate solutions automatically. We design a new encoding and decoding method and use the CPLEX, PSO algorithm, and proposed PSOB algorithm to solve the problem. From the computational experiments, we can find that the PSOB algorithm is able to find equally good solutions for small-scale instances and can find better solutions than the CPLEX solver and the PSO algorithm within a reasonable time.

Throughout this article, we assume that the resource is supplied at regular intervals and has a limited time for use. While these assumptions are essential and practical, the heuristic approach can easily be adapted to be used for different scenarios with arbitrary resource constraints. Therefore, the proposed approach can be used for a variety of industry production processes in which schedulers must take into account resource constraints.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [grant numbers 71771143 and 71771180], Innovation Program of Shanghai Municipal Education Commission [No. 14YZ111], Shanghai Young Eastern Scholar

Programme [QD2015041], Shanghai Pu Jiang Program (No. 13PJJC066), and Human Arts & Social Science Foundation of Ministry of Education of China (No. 16YJC630112).

References

- [1] L. Chen, B. Yang, X. Shen, Z. Xie, and F. Sun, "Thermodynamic optimization opportunities for the recovery and utilization of residual energy and heat in China's iron and steel industry: A case study," *Applied Thermal Engineering*, vol. 86, pp. 151–160, 2015.
- [2] J. Mori and V. Mahalec, "Planning and scheduling of steel plates production. Part II: scheduling of continuous casting," *Computers & Chemical Engineering*, vol. 6, pp. 438–446, 2016.
- [3] Q.-K. Pan, "An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling," *European Journal of Operational Research*, vol. 250, no. 3, pp. 702–714, 2016.
- [4] L. Tang, J. Liu, A. Rong, and Z. Yang, "Mathematical programming model for scheduling steelmaking-continuous casting production," *European Journal of Operational Research*, vol. 120, no. 2, pp. 423–435, 2000.
- [5] L. Tang, P. B. Luh, J. Liu, and L. Fang, "Steel-making process scheduling using Lagrangian relaxation," *International Journal of Production Research*, vol. 40, no. 1, pp. 55–70, 2002.
- [6] L. X. Tang, Y. Zhao, and J. Y. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steel-making-continuous casting production," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 209–225, 2014.
- [7] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.
- [8] R. K. Cheung, D. D. Hang, and N. Shi, "A labeling method for dynamic driver-task assignment with uncertain task durations," *Operations Research Letters*, vol. 33, no. 4, pp. 411–420, 2005.
- [9] S. Yan and C.-H. Tang, "A heuristic approach for airport gate assignments for stochastic flight delays," *European Journal of Operational Research*, vol. 180, no. 2, pp. 547–567, 2007.
- [10] L. Zhen, "Task assignment under uncertainty: Stochastic programming and robust optimisation approaches," *International Journal of Production Research*, vol. 53, no. 5, pp. 1487–1502, 2015.
- [11] L. Zhen, D. Zhuge, and S.-L. Zhu, "Production stage allocation problem in large corporations," *OMEGA - The International Journal of Management Science*, vol. 73, pp. 60–78, 2017.
- [12] C.-F. Chien and C.-H. Chen, "Using genetic algorithms (GA) and a coloured timed Petri net (CTPN) for modelling the optimization-based schedule generator of a generic production scheduling system," *International Journal of Production Research*, vol. 45, no. 8, pp. 1763–1789, 2007.
- [13] W. Kaczmarczyk, "Proportional lot-sizing and scheduling problem with identical parallel machines," *International Journal of Production Research*, vol. 49, no. 9, pp. 2605–2623, 2011.
- [14] D. Shabtay and M. Kaspi, "Parallel machine scheduling with a convex resource consumption function," *European Journal of Operational Research*, vol. 173, no. 1, pp. 92–107, 2006.
- [15] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *European Journal of Operational Research*, vol. 47, no. 3, pp. 271–292, 1990.
- [16] A. Allahverdi, J. N. D. Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations," *Omega*, vol. 27, no. 2, pp. 219–239, 1999.

- [17] E. Mokotoff, "Parallel machine scheduling problems: a survey," *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, pp. 193–242, 2001.
- [18] M. Pfund, J. W. Fowler, and J. N. D. Gupta, "A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems," *Journal of the Chinese Institute of Engineers*, vol. 21, no. 3, pp. 230–241, 2004.
- [19] T. Sen, J. M. Sulek, and P. Dileepan, "Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey," *International Journal of Production Economics*, vol. 83, no. 1, pp. 1–12, 2003.
- [20] C. Koulamas, "The single-machine total tardiness scheduling problem: review and extensions," *European Journal of Operational Research*, vol. 202, no. 1, pp. 1–7, 2010.
- [21] H. Emmons, "One-machine sequencing to minimize certain functions of job tardiness," *Operations Research*, vol. 17, no. 4, pp. 701–715, 1969.
- [22] J. Du and J. Y. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Research*, vol. 15, no. 3, pp. 483–495, 1990.
- [23] E. L. Lawler, "A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness," *Annals of Discrete Mathematics*, vol. 1, no. 8, pp. 331–342, 1977.
- [24] S.-O. Shim and Y.-D. Kim, "Scheduling on parallel identical machines to minimize total tardiness," *European Journal of Operational Research*, vol. 177, no. 1, pp. 135–146, 2007.
- [25] J.-Y. Lee and Y.-D. Kim, "A branch and bound algorithm to minimize total tardiness of jobs in a two identical-parallel-machine scheduling problem with a machine availability constraint," *Journal of the Operational Research Society*, vol. 66, no. 9, pp. 1542–1554, 2015.
- [26] J. Blazewicz, J. K. Lenstra, and A. H. Rinnooy Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11–24, 1983.
- [27] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Handbook on Scheduling*, Springer, 2007.
- [28] D. Briskorn, B.-C. Choi, K. Lee, J. Leung, and M. Pinedo, "Complexity of single machine scheduling subject to nonnegative inventory constraints," *European Journal of Operational Research*, vol. 207, no. 2, pp. 605–619, 2010.
- [29] D. Briskorn, F. Jaehn, and E. Pesch, "Exact algorithms for inventory constrained scheduling on a single machine," *Journal of Scheduling*, vol. 16, no. 1, pp. 105–115, 2013.
- [30] P. Györfyi and T. Kis, "Approximability of scheduling problems with resource consuming jobs," *Annals of Operations Research*, vol. 235, pp. 319–336, 2015.
- [31] S. Wang and M. Liu, "Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning," *Journal of Manufacturing Systems*, vol. 37, pp. 182–192, 2015.