mathematics

MDPI

Article

Model and Algorithm for a Two-Machine Group Scheduling Problem with Setup and Transportation Time

Yu Ni, Shufen Dai, Shuaipeng Yuan *, Bailin Wang and Zhuolun Zhang

School of Economics and Management, University of Science and Technology Beijing, No. 30, Xueyuan Road, Haidian District, Beijing 100083, China; daisf@ustb.edu.cn (S.D.); wangbl@ustb.edu.cn (B.W.); b2114136@ustb.edu.cn (Z.Z.)

* Correspondence: yuansp@ustb.edu.cn

Abstract: This paper investigates a two-machine group scheduling problem with sequence-independent setup times and round-trip transportation times, which is derived from the production management requirements of modern steel manufacturing enterprises. The objective is to minimize the makespan. Addressing limitations in prior studies, we consider a critical but largely ignored transportation method, namely round-trip transportation, and restricted transporter capacity between machines. To solve this problem, a mixed-integer programming model is first developed. Then, the problem complexity is analyzed for situations with both single and unlimited transporters. For the NP-hard case of a single transporter, we design an efficient two-stage heuristic algorithm with proven acceptable solution quality bounds. Extensive computational experiments based on steel plant data demonstrate the effectiveness of our approach in providing near-optimal solutions, and the maximum deviation between our algorithm and the optimal solution is 1.38%. This research can provide an operable optimization method that is valuable for group scheduling and transportation scheduling.

Keywords: group scheduling; transportation time; mathematical model; heuristic algorithm

MSC: 90B35; 90C59; 90-10

Citation: Ni, Y.; Dai, S.; Yuan, S.; Wang, B.; Zhang, Z. Model and Algorithm for a Two-Machine Group Scheduling Problem with Setup and Transportation Time. *Mathematics* **2024**, *12*, 888. https://doi.org/10.3390/math12060888

Academic Editors: Zsolt Tibor Kosztyán and Zoltán Kovács

Received: 19 February 2024
Revised: 11 March 2024
Accepted: 13 March 2024
Published: 18 March 2024

1. Introduction

Recent years have witnessed a surge of interest in tackling combinatorial optimization problems in various real-world industrial contexts, such as the steel manufacturing companies referenced in [1–3] and the manufacturing industry mentioned in [4,5]. This work investigates a two-machine group scheduling problem, where both the sequence independence setup times between groups and the transportation times between machines are considered. This problem is derived from the production management requirement of the tube processing workshop in iron and steel enterprises. Tube processing occurs after the hot rolling stage and includes two production stages: tube cutting and tube processing. In the cutting machine, each long tube is cut into several short tubes, and a certain setup time is required while switching tubes. Because the number of tubes in the cutting machine changes and the processing time is relatively long, the cutting process (CP) is the core link of the entire process flow. After cutting, the short tubes are trans-ported to the tube-processing stage by a transporter. Limited by the processing environment and tubes' properties, the number and capacity of transporters between machines are finite, and the transportation process is limited by the availability of the transporters. Specifically, for any tube, it can be transported only if the transporter has transported the previous tube to next machine and returned. If we take the long tubes at the cutting stage as the group of the short tubes after the CP, then the whole process can be abstracted into a two-machine group scheduling problem with setup times and round-trip transportation times. To achieve the goals of streamlined production and management, this scheduling problem needs to fully consider the constraints of transportation capacity between machines.

The round-trip transportation method and limited transportation capacity between machines make this problem significantly different to classical scheduling group problems. To the best of our knowledge, no relevant research on this problem has been published, resulting in significant limitations of existing achievements in practical applications. Therefore, we are motivated to investigate this type of problem in this work. We first develop a mixed-integer linear programming model with the objective of minimizing the makespan, and then prove that there is a polynomial optimal algorithm for the case that the number of transporters is unlimited. When there is only one transporter, it is proved that the problem is a strong NP-hard problem, and a heuristic algorithm with an upper bound of 3 is proposed. Finally, a simulation experiment based on actual production data is performed and it is verified that the algorithm has a good solution effect. We believe that our research in this work can further narrow the gap between theoretical results of group scheduling and practical applications.

The rest of this paper is organized as follows. Section 2 reviews the literature relevant to this study. Section 3 describes the studied problem and the developed model. Section 4 presents the characteristics of the problem. The designed algorithms are shown in Section 5. Computational results are presented in Section 6. Section 7 concludes our work and suggests areas for future study.

2. Literature Review

Classical scheduling problems have been studied by many scholars from different perspectives. For example, Tian et al. studied a single-machine parallel-batch scheduling problem with non-identical job sizes under time-of-use electricity prices [6]. The objective was to minimize the total energy consumption. To solve the problem, they proposed three mathematical programming models and designed a column-generation algorithm. Chen et al. [7] investigated an energy-oriented scheduling problem arising from hot rolling production in the steel enterprises. The problem was first modeled as a vehicle routing problem with special constraints, and then solved via a knowledge-based NSGA-II algorithm. Zhao et al. [8] addressed an energy-efficient flowshop scheduling problem with the objective of minimizing total energy consumption and total tardiness. A hyper-heuristic Q-learning technique was developed. Tian et al. [9] studied a remanufacturing system scheduling problem with lot-streaming production mode. They formulated the problem as a multi-objective mathematical model aimed at simultaneously minimizing the total energy consumption and makespan. A hybrid optimization algorithm combing the fruit fly optimization algorithm and simulated annealing mechanism was developed. For comprehensive research on scheduling problems, the reader can refer to Dauzère-Pérès et al. [10] and Strusevich [11].

Due to their practical relevance, group scheduling problems have also received much attention in this academic field over the past decade. Lin and Ying [12] investigated a no-wait flowshop group scheduling problem with sequence-dependent setup times. They first established the problem's strong NP-hardness and subsequently formulated a linear programming model with the objective of reducing the makespan. To address this complex problem, they proposed a two-stage metaheuristic algorithm. Costa et al. [13] investigated a flowshop group scheduling with blocking constraints, and developed a parallel genetic algorithm based on adaptive and parallel computing techniques. Recently, Zhao et al. [14] extracted a flowshop group scheduling problem with batch processing characteristics from the steel industry. A mixed-integer linear programming model and a memetic algorithm were formulated to minimize the total number of delayed jobs and the total setup times. Pan et al. [15] explored a distributed flowshop group scheduling problem with the objective of minimizing the makespan. They constructed a mixed-integer linear programming model and proposed a cooperative co-evolutionary algorithm. Building upon Pan's work, Wang et al. [16] further refined the problem by focusing on tardiness. Their improved iterative greedy algorithm demonstrated promising results in minimizing total tardiness. Neufeld

et al. [17] provided a comprehensive overview of the current research status about the flow-shop group scheduling problem.

There are few related studies on the group scheduling problem considering transportation times between machines. Liou and Hsieh [18] proposed a hybrid particle swarm and genetic algorithm based on the classic flowshop environment and presented two lower bounds of the optimal solution through theoretical analysis. For the same problem, Ahmadizar and Shahmaleki [19] formulated a mixed-integer linear programming model with the goal of minimizing the total completion time. A genetic algorithm embedding problem specific rules was proposed. Yuan et al. [20] investigated a flowshop group scheduling problem with round-trip transportation times. A mixed-integer linear programming model and an improved differential evolution algorithm were developed with the objective of minimizing makespan.

Although there is a large body of work in the field of group scheduling, most of the studies have focused on the setup time constraint, and only a few studies have considered the transportation process between machines simultaneously. Even for those that have, a fundamental assumption has been that no constraint exists on the availability of transporters between machines, and only one-way transportation time was investigated. That is, once a job is processed on the current machine, it can be transported to the next machine in real time. However, this ideal scenario does not fully align with the realities of industrial settings. As a result, the existing results fall short of addressing the practical requirements of industrial applications, such as the abovementioned tube-processing method in modern iron and steel enterprises. To our knowledge, we are the first to study the group scheduling problem, where the number and capacity of the transporters between machines is limited.

3. Problem Description and Mathematical Model

3.1. Problem Description

The two-machine group scheduling problem with sequence independence setup and transportation times can be summarized as follows: There are g groups to be processed on two machines, and each group has $n_i (1 \leq i \leq g)$ jobs. Each job may be processed on at most one machine at a time, and each machine can process at most one job at a time. The jobs in the same group must be processed continuously without any interruption by the jobs of other groups. Especially, each group requires a sequence-independent setup time on both machines. A round-trip transportation time is required for moving the jobs from the first machine to the second machine. Each transporter must complete a round-trip before the next job can be transported. Different jobs might have different transportation times. In addition, it is assumed that the capacity of the transporter is 1. The objective is to minimize the makespan.

This scheduling problem needs to determine the sequence of groups, and the sequence of jobs within each group so that the makespan is the minimum. If we use GT to denote a group scheduling with intermediate transportation, where x is the number of available transporters and y is the single transport capacity per transporter, then the scheduling problem in this paper can be described as $F2|GT|V(x,1)|C_{\max}$.

3.2. Notations and Model

Notations used in this work are defined below.
g: The number of groups.
G_i: Group index, $i = 1, 2, \ldots, g$.
n_i: The number of jobs in group G_i.
J_{il}: Job index in group G_i, $l = 1, 2, \ldots, n_i$.
M_k: Machine index, $k = 1, 2$.
s_{ilk}: Start time of the job J_{il} on machine M_k.
p_{ilk}: Processing time of the job J_{il} on machine M_k.
c_{ilk}: Completion time of the job J_{il} on machine M_k.
st_{ik}: Setup time of group G_i on machine M_k.

t_1, t_2: Round-trip transportation time between M_1 and M_2, where t_1 is the time moving from M_1 to M_2, and t_2 is the returning time.

U: A very large number.

X_{ij}: Decision variables, equal to 1 if and only if the group G_j is processed immediately after the group G_i.

$x_{i,l'l}$: Decision variables, equal to 1 if and only if job J_{il} is processed immediately after the job $J_{il'}$.

b_{il}: Start transportation time of job J_{il} on machine M_1.

C_{ik}: Completion time of group G_i on machine M_k.

Objective functions and constraints of the model are formulated as follows.

$$\min C_{\max} = \max\{C_{i2} | 1 \le i \le g\} \tag{1}$$

$$\sum_{i=0}^{g} X_{ij} = 1,\ 1 \le j \le g \tag{2}$$

$$\sum_{i=1}^{g} X_{ji} = 1,\ 1 \le j \le g \tag{3}$$

$$X_{ij} + X_{ji} \le 1,\ 1 \le i, j \le g \tag{4}$$

$$\sum_{l'=0}^{n_i} x_{i,l'l} = 1,\ 1 \le i \le g,\ 1 \le l \le n_i \tag{5}$$

$$\sum_{l'=1}^{n_i} x_{i,ll'} \le 1,\ 1 \le i \le g,\ 1 \le l \le n_i \tag{6}$$

$$x_{i,ll'} + x_{i,l'l} \le 1,\ 1 \le i \le g,\ 1 \le l, l' \le n_i \tag{7}$$

$$s_{jlk} - c_{il'k} - st_{ik} + U(1 - X_{ij}) \ge 0, 1 \le i, j \le g, l \le n_j, l' \le n_i \tag{8}$$

$$s_{ilk} - c_{il'k} + U(1 - x_{i,l'l}) \ge 0, 1 \le i \le g, 1 \le l', l \le n_i \tag{9}$$

$$s_{il2} - b_{il} - t_1 \ge 0,\ 1 \le i \le g,\ 1 \le l \le n_i \tag{10}$$

$$b_{il} \ge c_{il1},\ 1 \le i \le g,\ l = 1, \ldots, n_i \tag{11}$$

$$C_{i2} \ge c_{il2},\ 1 \le i \le g,\ 1 \le l \le n_i \tag{12}$$

$$X_{ij}, x_{i,l'l} \in \{0,\ 1\},\ 1 \le i, j \le g, 1 \le l', l \le n_i \tag{13}$$

Objective (1) indicates the minimum completion time of all jobs. Constraints (2)–(4) indicate that each group should have exactly one predecessor (containing dummy group G_0) and at most one successor, and these groups cannot be the same. Similarly, Constraints (5)–(7) indicate that, within the same group, each job has exactly one immediate predecessor and at most one immediate successor, with the condition that these two cannot be the same. Constraints (8) and (9) ensure that each machine can process at most one job at a time; Constraint (10) indicates that the start processing time of J_{il} on M_2 cannot be earlier than its arrival time. Constraint (11) shows that the start transportation time of a job should be no less than its completion time on M_1. Constraint (12) defines the makespan. Constraint (13) defines the binary decision variables.

It should be noted that the correctness of this model has been verified by a well-known off-the-shelf solver called CPLEX. Detailed experimental results are provided in Section 6.2.

4. Problem Analysis

This section discusses the complexity of a problem where the number of transporters is unlimited and when there is one. For the convenience of the following description, the problem studied is defined as P.

(1) Unlimited number of transporters:

When the number of transporters is unlimited, or at least greater than the number of all jobs to be scheduled, there is always a transporter available in real time for any jobs that are completed on machine M_1. In this case, the transportation time between machines can be viewed as a component of the processing time. This problem is equivalent to $F2|GT|C_{\max}$. Logendran et al. [21] has proved that there is a polynomial optimization algorithm for such problem.

(2) One transporter:

When the number of transporters is only one, the start time of different jobs in machine M_2 will be subject to the arrival time of the transporter. The nature of the problem undergoes significant changes, and the algorithm for an infinite number of transporters will no longer be applicable. The following theorem proves that the problem is strongly NP-hard in this case.

Theorem 1. *The scheduling problem P has strong NP-hard characteristics when* $x = 1$.

Proof. We prove the NP-hardness of this problem by a reduction from the 3-partition problem, which is described as follows: Given $3t$ items $T = \{a_1, a_2, \ldots, a_{3t}\}$ and a positive integer b, each item $a_j \in T$ satisfies $b/4 < a_j < b/2$, and $\sum_{j=1}^{3t} a_j = tb$. The question asks whether there are t disjoint subsets T_1, T_2, ..., T_t of T such that each subset contains exactly three items and their total size is equal to b?

Given a 3-partition problem, we construct an instance for our problem as follows:

$$
\begin{aligned}
&g = t,\ n_i = 5, 1 \le i \le g;\\
&p_{i11} = b/4, p_{i21} = p_{i31} = p_{i41} = b/3, p_{i51} = 3b;\\
&p_{i12} = 3b,\ p_{i22} = a_{3i-2}, p_{i32} = a_{3i-1}, p_{i42} = a_{3i},\ p_{i52} = b;\\
&t_1 = t_2 = b/2;\\
&st_{i1} = st_{i2} = 3b/4;
\end{aligned}
$$

Threshold value $z = 5tb + 3b/4$.

Sufficiency: If there is a solution to the 3-partition problem, then we can construct a scheduling instance as shown in Figure 1, whose makespan is equal to z.

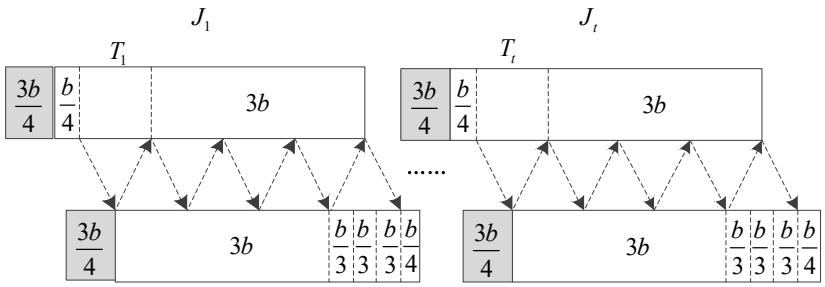

Figure 1. Schedule instance.

Necessity: The completion time of each group is not less than $st_{i1} + \min(p_{il1}) + 5t_1 + 4t_2 + \min(p_{il2})$, and the $C_{\max}$ is not less than $st_{i1} + \min(p_{il1}) + 5t(t_1 + t_2) - t_2 + \min(p_{il2})$, that is $C_{\max} \ge 5tb{+}3b/4$. The equality of $C_{\max} = 5tb{+}3b/4$ is satisfied if and only the following two conditions hold true.

(1) In any group G_i, the job J_{i1} must be processed at the first position, and the job J_{i5} must be processed at the last position.

(2) The transporter must operate continuously without any idle time until the last job reaches machine M_2.

It can be easily demonstrated that the $C_{\max}$ will be greater than z if any of the aforementioned conditions are not met. Suppose that the 3-partition problem has no solution, which implies that there exists a T_i such that $\sum_{a_j \in T_i} a_j \neq b$, or, equivalently, a T_w such that $\sum_{a_j \in T_w} a_j > b$. Consequently, $C_{w1} > 17b/4$, indicating that there is some idle time for transporter while transporting jobs in group G_w, which violates the above condition 2. As previously discussed, this situation is untenable. □

5. Solving Algorithm

5.1. Optimality Analysis

Because the problem $F2|GT|V(1,1)|C_{\max}$ has strong NP-hard characteristics, we can only construct heuristic algorithm by analyzing the characteristics of the problem. In this section, the properties of the optimal solution are analyzed first, and then a heuristic algorithm with upper bound of 3 is proposed.

Lemma 1. *In any optimal scheduling, the sequence of jobs on the two machines is identical, and the order of transportation is also the same.*

Proof. The above lemma can be proved by standard theory of interchange of jobs; there is no further detail to provide. □

From the above lemma, it can be observed that the two sub-problems of the groups sequence and the jobs sequence within each group satisfy the characteristics of the permutation flowshop. For the first sub-problem, it is equivalent to $TF2|prmu|V(1,1)|C_{\max}$, and for the other one, due to the overlapping of the processing times between the groups, it is necessary to introduce the following two definitions.

Definition 1 (Extension). *The difference between the completion time of the group G_i on the two machines is called extension, denoted as B_i. The calculation formula is $B_i = C_{i2} - C_{i1}$.*

Definition 2 (Indentation). *The maximum extension of a group can be provided for its tight front group without changing its own completion time. and the calculation formula is $A_i = max(C_{i1} + t_1, c_{i(n_i-1)2}) - \sum_{l=1}^{n_i-1} p_{il2} - st_{i2}$. It is shown as the shadow part of Figure 2.*

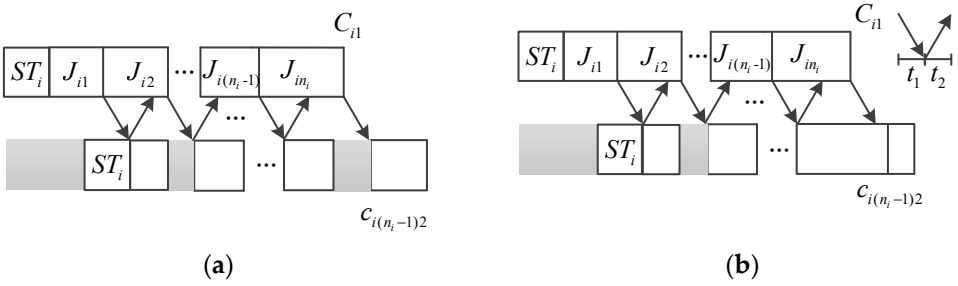

Figure 2. Classification of the indentation for (**a**) $C_{i1} + t_1 > c_{i(n_i-1)2}$ and (**b**) $C_{i1} + t_1 \leq c_{i(n_i-1)2}$.

5.2. Heuristic Algorithm

Figure 2 indicates that the overlap time changes with the location change in the scheduling sequence. Therefore, we can propose the following two-stage heuristic algorithm (Algorithm 1) based on the value of extension and indentation.

Algorithm 1: Two-stage heuristic algorithm

for $i = 1$ **to** g
  for $l = 1$ **to** n_i
    $N_1 \leftarrow \{J_{il} | p_{il1} <= t_1 + t_2\}$.
    $N_2 \leftarrow \{J_{il} | p_{il1} > t_1 + t_2\}$.
  end
  $S_1 \leftarrow$ Sort the sequence according to the SPT rule of p_{il2} for jobs in N_1.
  $S_2 \leftarrow$ Sort the sequence according to the SPT rule of p_{il1}, p_{il2} for jobs in N_2.
  $\pi_i \leftarrow [S_1,\ S_2]$ /* π_i is the job sequence in group G_i */
  /* Calculate A_i and B_i according to the π_i */
  $A_i \leftarrow \max(C_{i1} + t_1, c_{i(n_i-1)2}) - \sum_{l=1}^{n_i-1} p_{il2} - st_{i2}$.
  $B_i \leftarrow C_{i2} - C_{i1}$.
end
$N_3 \leftarrow \{G_i | A_i \leq B_i\}$.
$N_4 \leftarrow \{G_i | A_i > B_i\}$.
$S_3 \leftarrow$ Sort the groups in N_3 according to the SPT rule of A_i.
$S_4 \leftarrow$ Sort the groups in N_4 according to the LPT rule of B_i.
$\Pi \leftarrow [S_3,\ S_4]$ /* Π is the group sequence */

The following Lemma 2 gives the upper bound of the algorithm.

Lemma 2. *The above algorithm provides a bound of no more than 3 for the problem.*

Proof. Assume that the target value obtained by the above algorithm is C, and the optimal solution to this problem is C^*. Hence, the upper bound of C must be less than $\sum_{i=1}^{g} p_{i1} + \sum_{i=1}^{g} st_{i1} + \sum_{i=1}^{g} \sum_{j=1}^{n_i} (t) - t_2 + \sum_{i=1}^{g} p_{i2}$. In addition, we can see that C^* has the following three lower bounds:

$$LB_1 = \sum_{i=1}^{g} p_{i1} + \sum_{i=1}^{g} st_{i1} + t_1 + \min(p_{il2}), \tag{14}$$

$$LB_2 = \min(st_{ik}) + \min(p_{il1}) + t_1 + \sum_{i=1}^{g} p_{i2}, \tag{15}$$

$$LB_3 = \min(st_{i1}) + \min(p_{il1}) + (\sum_{i=1}^{n} \sum_{l=1}^{n_i} (t_1 + t_2)) - t_2 + \min(p_{il2}). \tag{16}$$

Therefore, the upper bound of the algorithm can be derived from the following equation.

$$\frac{C}{C^*} = \frac{3C}{3C^*} \leq \frac{3C}{LB_1 + LB_2 + LB_3} \leq \frac{3\sum_{i=1}^{n} p_{i1} + 3((\sum_{i=1}^{n} \sum_{l=1}^{n_i} (t_1 + t_2)) - t_2) + 3\sum_{i=1}^{n} p_{i2} + 3\sum_{i=1}^{n} ST_i}{\sum_{i=1}^{n} p_{i1} + \sum_{i=1}^{n} p_{i2} + \sum_{i=1}^{n} ST_i + (\sum_{i=1}^{n} \sum_{l=1}^{n_i} (t_1 + t_2)) - t_2} = 3 \tag{17}$$

□

6. Experimental Results and Discuss

In this section, computational experiments are presented to evaluate the performance of the proposed model and the heuristic algorithm. First, we use small-scale instances to verify the correctness of the model and the deviation between the algorithm and the optimal solution solved by model. Then, the algorithm was compared with the three lower bounds of the optimal solution using large-scale instances. The algorithm is implemented using Matlab language (Matlab R2015a), and the model is solved by CPLEX 12.10. The code-running environment is i5-12500H Intel CPU with 16.00 GB RAM hardware environment.

6.1. Experiment Design

As mentioned before, a difference exists between our studied problem and those in the literature, and no existing benchmark data can be used directly. Therefore, a number of test instances are generated based on the real situations from a large steel plant in China. The following combinations of g and n_i ($1 \leq i \leq g$) are set: $g \in \{3, 5, 10, 20, 50, 100, 150\}$ and $n_i \in \{3, 5, 10, 15\}$. Those data are divided into 6 small-scale classes ($g \in \{3, 5, 10\}$ and $n_i \in \{3, 5\}$) and 12 large-scale classes ($g \in \{20, 50, 100, 150\}$ and $n_i \in \{5, 10, 15\}$). One instance is generated for each combination, which results in 18 instances in total. Other production data for the instances are generated as follows: $p_{ilk} \in DU(5,30)$, $t_1, t_2 = DU[5,15]$; $st_{ik} = DU[5,10]$. Here, $DU(a,b)$ means a discrete uniform distribution between a and b.

6.2. Optimality Test

This section conducts experiments on the aforementioned six small-scale instances to verify the correctness of the model, and to observe the deviation of $C_{\max}$ obtained by our designed algorithm from the optimal one. For each instance, the time limit of CPLEX solver is set to 3600 s. The relative percentage increase (RPI) is used as the evaluation index; RPI is calculated by

$$\mathrm{RPI}(A) = (C_{\max}(A) - C^*_{\max}) \,/\, C^*_{\max} \times 100, \tag{18}$$

where $C_{\max}(A)$ is the objective value obtained by our algorithm, and $C^*_{\max}$ is the optimal target value obtained by CPLEX. The experimental results are shown in Table 1, where $LB_{\max}$ represents the maximum value of three lower bounds.

Table 1. Results for the model and designed algorithm.

$g \times n_i$	LB_{max}	$C_{max}(CPLEX)$	$C_{max}(A)$	RPI
3×3	217	222	222	0.000
3×5	298	298	298	0.000
5×3	434	451	455	0.887
5×5	528	539	542	0.557
10×3	610	612	619	0.814
10×5	915	917	931	1.086
Average	500	506	511	0.557

As shown in Table 1, when the problem scale is very small ($g = 3$ & $n_i = 3$), both CPLEX and our algorithm yield the same $C_{\max}$ value, confirming the correctness of the model. The $LB_{\max}$ values of the six instances are all smaller than the optimal $C_{\max}$ obtained by CPLEX, and the differences between them are extremely small, with an average deviation ratio of 1.2%. This indicates that the lower bound values are tight. Overall, the designed algorithm achieves a maximum RPI of 1.086% and an average RPI of 0.557%, which shows that the deviations in the solutions obtained by the algorithm from the optimal ones are very small. Therefore, we can conclude that our designed algorithm has a good ability to construct high-quality solutions for small-scale instances.

6.3. Results for Large-Scale Instances

To further validate the performance of our algorithm, we conducted experiments using the above 12 large-scale problem instances. Since these instances are beyond the solvability range of CPLEX, the $LB_{\max}$ is used as the benchmark for each instance. The results are shown in Table 2, where T represents the computing time of algorithms.

Table 2. Experimental results.

$g \times n_i$	LB_{max}	C	RPI	T
20 × 5	2307	2309	0.22	0.09
20 × 10	4673	4684	0.24	0.12
20 × 15	7254	7254	0.00	0.12
50 × 5	5263	5298	0.66	0.18
50 × 10	10,511	10,569	0.55	0.12
50 × 15	11,000	11,053	0.48	0.17
100 × 5	11,000	11,024	0.21	0.21
100 × 10	22,000	22,028	0.12	0.19
100 × 15	33,000	33,051	0.15	0.28
150 × 5	15,134	15,245	0.73	0.33
150 × 10	30,023	30,347	1.07	0.50
150 × 15	46,348	46,991	1.38	0.58
Average	16,543	16,654	0.48	0.24

It can be seen from Table 2 that, as the problem size increases, the RPI value also increases, indicating that the difficulty of solving the problem is positively related to the problem size. For the RPI indicator, there are 7 instances with RPI values less than 0.5%, and 10 instances with RPI values less than 1%. The maximum RPI value between the proposed algorithm and the maximum lower bound ($LB_{\max}$) is 1.38%, and the minimum RPI value is zero. Overall, the average RPI value for all instances is just 0.48%, which indicates that the solution obtained by our algorithm is very close to the lower bound of the optimal solution.

The average solution time of all instances for our algorithm is just 0.24 s. Even if the problem scales up to 150 × 15, it can be completed in 0.58 s, showing that the algorithm has high efficiency.

To analyze the performance of the algorithms in solving instances with different numbers of total jobs, a series of statistical analyses based on means and LSD intervals for IGD values are performed. The results are shown in Figure 3.

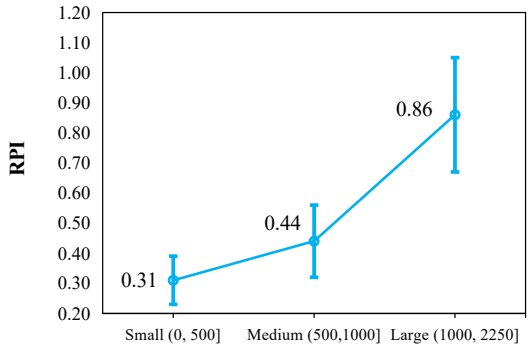

Figure 3. Results of RPI for different number of job sizes.

From Figure 3, we can observe that as the number of jobs increases, the RPI value of our algorithm shows a slight increase trend. Specifically, when the total number of jobs is in the small-scale range (0, 500], the overall RPI value is 0.31. It increases to 0.41 while the total number of jobs is in the medium-scale range (500, 1000], and further increased to 0.86 when the job size is in the large-scale range (1000, 2250]. The above results indicate that the number of jobs has a certain impact on the performance of the algorithm. No matter what the situation, the RPI value from our algorithm is always at a lower level, which confirms the effectiveness of our algorithm in solving the performance problem.

6.4. Comparison with a Real-World Heuristic

In the actual industrial production process, heuristic algorithms based on professional knowledge and human experience are usually used to solve the corresponding scheduling

problems. For the problem studied in this work, the most commonly used method by enterprises is the largest processing time (LPT) rule, in which the job (group) with a longer processing time is given a higher priority. To be specific, the job scheduling sub-problem is solved first. For each group, sort the jobs within it in decreasing order of the total processing times on both the two machines ($p_{il1} + p_{il2}$), and generated a job sequence. Then, the completion time of each group should be calculated according to the job sequence within it; also, all of the groups should be arranged in decreasing order of their completion time.

The above heuristic (defined as A_1) is compared with our heuristic on the generated 12 instances. The experimental results are presented in Table 3. From Table 3, it can be seen that our algorithm achieves much better results than the real-world heuristic on all considered instances. The minimum RPI value between those two algorithms is 0.09%, the maximum value is 2.44%, and the average value is 0.74%, further confirming the effectiveness of our algorithm. The 0.74% improvement in makespan may seem modest, but it has a significant impact in industrial applications. It not only enhances production efficiency, but also facilitates energy conservation and emission reduction for energy-intensive manufacturing enterprises. In fact, our designed algorithm has been implemented in a large steel company in China, with satisfactory outcomes.

Table 3. Comparison for our algorithm and a real-world heuristic.

$g \times n_i$	A	A_1	$Gap(A_1 - A)$	RPI
20×5	2309	2333	24	1.04
20×10	4684	4739	55	1.17
20×15	7254	7267	13	0.18
50×5	5298	5320	22	0.42
50×10	10,569	10,827	258	2.44
50×15	11,053	11,064	11	0.10
100×5	11,024	11,115	91	0.83
100×10	22,028	22,053	25	0.11
100×15	33,051	33,088	37	0.11
150×5	15,245	15,404	159	1.04
150×10	30,347	30,375	28	0.09
150×15	46,991	47,613	622	1.32
Average	16,654	16,766	112	0.74

Therefore, it can be concluded that the designed heuristic algorithm is very effective in solving the studied problem.

7. Conclusions

In this work, a two-machine group scheduling problem arising from the real-world steel production is studied. This study makes both practical and theoretical contributions to the field by incorporating round-trip transportation times and limited transportation capacity constraints. To solve the problem, a mixed-integer programming formulation is first established to minimize the makespan. The problem's complexity is examined from two angles. First, when the number of transporters between machines is infinite, we demonstrate that the problem can be solved using a polynomial optimization algorithm. Second, in scenarios with only one transporter, it is proved that the problem is a strongly NP-hard problem. A two-stage heuristic algorithm with an upper bound of 3 is proposed. The proposed heuristic maintains solution quality within 1.38% of lower bounds for large problem instances. Overall, this study significantly advances our understanding of how to combine coordinated scheduling decisions with restricted material flows, which is essential to many multi-stage manufacturing settings. Practical implementations leveraging this research have the potential to yield reduced steel production timelines and improved just-in-time capabilities. From a theoretical perspective, the analysis and algorithms presented open rich avenues for better incorporating logistic realities like finite transporter into classical scheduling models.

Although this paper has made some progress, the following limitations remain. First, only a two-machine case is considered. However, there are often more than two machines in actual industrial environments. Second, the case of multiple transporters is not considered. Third, the two-stage heuristic algorithm designed in this work has certain limitations in the solution quality when solving some large-scale instances. Therefore, an area for further work is extending the problem to environments with more machines and transportation networks. Moreover, using intelligent optimization and reinforcement learning techniques to design more efficient algorithms is also an effective research direction.

Author Contributions: Conceptualization, Y.N. and S.D.; software, validation, investigation, and data curation, S.Y., B.W. and Z.Z.; methodology, formal analysis, and writing—original draft preparation, S.Y.; writing—review and editing and funding acquisition, Y.N. and S.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 72301026, Ministry of education of Humanities and Social Science project, grant number 23YJA630090.

Data Availability Statement: The datasets used in the study are available from the corresponding author on reasonable request.

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

References

1. Özgür, A.; Uygun, Y.; Hütt, M. A review of planning and scheduling methods for hot rolling mills in steel production. *Comput. Ind. Eng.* **2021**, *151*, 106606. [CrossRef]
2. Soares, L.C.R.; Carvalho, M.A.M. Application of a hybrid evolutionary algorithm to resource-constrained parallel machine scheduling with setup times. *Comput. Oper. Res.* **2022**, *139*, 105637. [CrossRef]
3. Cao, J.; Pan, R.; Xia, X.; Shao, X.; Wang, X. An efficient scheduling approach for an iron-steel plant equipped with self-generation equipment under time-of-use electricity tariffs. *Swarm Evol. Comput.* **2021**, *60*, 100764. [CrossRef]
4. Chen, C.; Fathi, M.; Khakifirooz, M.; Wu, K. Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. *Comput. Ind. Eng.* **2022**, *165*, 107915. [CrossRef]
5. Bitar, A.; Dauzère-Pérès, S.; Yugma, C.; Roussel, R. A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. *J. Sched.* **2016**, *19*, 367–376. [CrossRef]
6. Tian, Z.; Zheng, L. Single machine parallel-batch scheduling under time-of-use electricity prices: New formulations and optimisation approaches. *Eur. J. Oper. Res.* **2024**, *312*, 512–524. [CrossRef]
7. Chen, L.; Cao, L.; Wen, Y.; Chen, H.; Jiang, S. A knowledge-based NSGA-II algorithm for multi-objective hot rolling production scheduling under flexible time-of-use electricity pricing. *J. Manuf. Syst.* **2023**, *69*, 255–270. [CrossRef]
8. Zhao, F.; Di, S.; Wang, L. A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. *IEEE Trans. Cybern.* **2023**, *53*, 3337–3350. [CrossRef]
9. Tian, G.; Wang, W.; Zhang, H.; Zhou, X.; Zhang, C.; Li, Z. Multi-objective optimization of energy-efficient remanufacturing system scheduling problem with lot-streaming production mode. *Expert Syst. Appl.* **2024**, *237*, 121309. [CrossRef]
10. Dauzère-Pérès, S.; Ding, J.; Shen, L.; Tamssaouet, K. The flexible job shop scheduling problem: A review. *Eur. J. Oper. Res.* **2024**, *314*, 409–432. [CrossRef]
11. Strusevich, V.A. Complexity and approximation of open shop scheduling to minimize the makespan: A review of models and approaches. *Comput. Oper. Res.* **2022**, *144*, 105732. [CrossRef]
12. Lin, S.; Ying, K. Makespan optimization in a no-wait flowline manufacturing cell with sequence-dependent family setup times. *Comput. Ind. Eng.* **2019**, *128*, 1–7. [CrossRef]
13. Costa, A.; Cappadonna, F.V.; Fichera, S. Minimizing makespan in a flow shop sequence dependent group scheduling problem with blocking constraint. *Eng. Appl. Artif. Intell.* **2020**, *89*, 103413. [CrossRef]
14. Zhao, Z.; Liu, S.; Zhou, M.; Abusorrah, A. Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1199–1209. [CrossRef]
15. Pan, Q.; Gao, L.; Wang, L. An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. *IEEE Trans. Cybern.* **2022**, *52*, 5999–6012. [CrossRef]
16. Wang, Z.; Pan, Q.; Gao, L.; Wang, Y. An effective two-stage iterated greedy algorithm to minimize total tardiness for the distributed flowshop group scheduling problem. *Swarm Evol. Comput.* **2022**, *74*, 101143. [CrossRef]
17. Neufeld, J.S.; Gupta, J.N.D.; Buscher, U. A comprehensive review of flowshop group scheduling literature. *Comput. Oper. Res.* **2016**, *70*, 56–74. [CrossRef]
18. Liou, C.; Hsieh, Y. A hybrid algorithm for the multi-stage flow shop group scheduling with sequence-dependent setup and transportation times. *Int. J. Prod. Econ.* **2015**, *170*, 258–267. [CrossRef]

19. Ahmadizar, F.; Shahmaleki, P. Group-shop scheduling with sequence-dependent set-up and transportation times. *Appl. Math. Model.* **2014**, *38*, 5080–5091. [CrossRef]
20. Yuan, S.; Li, T.; Wang, B. A discrete differential evolution algorithm for flow shop group scheduling problem with sequence-dependent setup and transportation times. *J. Intell. Manuf.* **2020**, *32*, 427–439. [CrossRef]
21. Logendran, R.; Salmasi, N.; Sriskandarajah, C. Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. *Comput. Oper. Res.* **2006**, *33*, 158–180. [CrossRef]