Abstract: This paper is aimed at studying a two-machine flowshop scheduling where the processing times are linearly dependent on the waiting times of the jobs prior to processing on the second machine. That is, when a job is processed completely on the first machine, a certain delay time is required before its processing on the second machine. If we would like to reduce the actual waiting time, the processing time of the job on the second machine increases. The objective is to minimize the makespan. When the processing time is reduced, it implies that the consumption of energy is reduced. It is beneficial to environmental sustainability. We show that the proposed problem is NP-hard in the strong sense. A 0-1 mixed integer programming and a heuristic algorithm with computational experiment are proposed. Some cases solved in polynomial time are also provided.

Keywords: flowshop scheduling; sustainability; NP-hard; integer programming; heuristic algorithm

1. Introduction

In this paper, we consider a two-machine flowshop scheduling problem where the processing times are linearly dependent on the waiting times of the jobs prior to processing on the second machine. The objective is to minimize the makespan. In this problem, when a job is processed completely on the first machine, a certain delay time is required before its processing on the second machine. If we can find a way to reduce the certain delay time at the cost of extra processing time added to the processing time of the job on the second machine, the whole processing time of all jobs may be reduced. When the whole processing time is reduced, the consumption of energy is reduced. This contributes to environmental sustainability. In addition, the managerial implication of this approach is that a production scheduler has more leeway to arrange a job sequence to reach the goal. One realistic example of this kind of scheduling exists in a painting process. Generally, there are at least two layer-painting stages in a painting engineering. After the first stage, the product has to wait for some time until it is dry naturally. Then, the product goes to the second stage. If we want to reduce the waiting time, we can use a dryer to accelerate the drying process. This implies that the cost of extra processing time is added to the processing time of the job on the second machine. Another practical scheduling problem arises at the cooking–chilling stage in a food plant [1]. The chilling process must start within 30 min of the completion of cooking. Otherwise, the food has to be discarded as unfit for human consumption. If there is an advantage in terms of the makespan minimization, we can reduce the waiting time after the completion of cooking at the cost of an extra processing time of the chilling process. In such a situation, it is good for the sustainability of the earth because it involves less food...
waste. Also, the objective of the makespan is minimized. The processing time of all jobs is reduced. The consumption of the energy is reduced. It is beneficial for environmental sustainability. For some other scheduling problems related to sustainability, the reader is referred to the works of [2–8].

The two-machine, minimum makespan, flowshop scheduling problem was first solved by Johnson [9]. Mitten [10] extended this problem with time lags. Sule and Huang [11] permitted the setup and removal times to be independent of the processing time. Maggu et al. [12] considered the problem with time lags and transportation time between machines. These problems proposed above are solved by a polynomial time algorithm, which is similar to Johnson’s rule [9]. There are also many studies considering scheduling problems with sequence dependent setup time. For example, Ruiz [13] proposed an iterated greedy heuristic to solve the flowshop scheduling problem with sequence dependent setup time. Nishi and Hiranaka [14] applied the Lagrangian relaxation and cut generation method to minimize the total weighted tardiness in flowshop scheduling problems with sequence dependent setup time. Wang et al. [15] presented a hybrid local search algorithm to solve the sequence setup times flowshop scheduling problem with a makespan criterion.

For the two-machine flowshop scheduling problem with waiting time constraints (or intermediate delays), Reddi and Ramamoorthy [16] proposed a polynomial time algorithm for the problem with no-wait in process. Dell’Amico [17] showed that the two-machine flowshop scheduling problem with intermediate delays is NP-hard in the strong sense if the solution space is not restricted to permutation schedules. In addition, Yu et al. [18] showed that the two-machine flowshop machine problem with intermediate delays is NP-hard in the strong sense, even if all processing times are 1. Fiszmann and Mosheiov [19] studied the scheduling problem of minimizing total load on a proportionate flowshop. They considered position-dependent job processing times in the most general way. They showed that this problem is solved in $O(n^4)$ time, where $n$ is the number of jobs. Yang and Chern [20] considered a two-machine flowshop sequencing problem with limited waiting time constraints; they showed that the permutation scheduling problem is NP-hard and proposed a branch-and-bound algorithm. Su [21] extended the problem studied by Yang and Chern [20] and considered a two-stage flowshop with a batch processor in stage 1 and a single processor in stage 2. Each batch processor can process a batch (limited number) of jobs simultaneously. A heuristic algorithm and a mixed integer program are proposed. Sriskandarajah and Goyal [22] considered a problem in which the processing times are linearly dependent on the waiting times; they showed that the problem is NP-hard and proposed a heuristic algorithm. Yang and Chern [23] further extended the problem studied by Sriskandarajah and Goyal [22] and considered a problem in which the processing time of a job on the second machine is linearly dependent on the waiting time if the waiting time is beyond a certain range. They proposed an integer program and a heuristic algorithm to solve the problem.

Chung et al. [24] considered a two-stage hybrid flowshop scheduling problem with a waiting time constraint. They provided two algorithms to solve the makespan minimization problem. Wang et al. [25] investigated a permutation flowshop scheduling problem with a time lag between two consecutive machines. They presented a two-stage constructive heuristic to minimize the makespan.

The proposed problem is different from those studied by Sriskandarajah and Goyal [22] and Yang and Chern [23]. In the problem of Sriskandarajah and Goyal [22], when a job is processed completely on the first machine, the job can be processed immediately on the second machine. No delay time is required. If there is a delay time before a job to be processed on the second machine, the processing time of the job on the second machine increases. In the problem of Yang and Chern [23], when a job is processed completely on the first machine, there is a waiting time before its processing on the second machine. If the waiting time is beyond a certain range, the processing time of a job on the second machine increases. This problem is similar to the following problem. When a job is processed completely on the first machine, a certain delay time (i.e., a certain waiting time) is required before its processing on the second machine. However, the actual waiting time can be either decreased or increased. The cost of the actual waiting time change is the increase of the processing time of a job on the second machine. In the proposed problem, we only consider the case in which the actual waiting
time is allowed to be decreased at the cost of extra processing time added to the processing time of a job on the second machine.

The organization of the remainder of this paper is as follows. In Section 2, there is a description of the problem and its complexity. A 0-1 mixed integer programming formulation is given in Section 3 and the heuristic algorithm is presented in Section 4. Thereafter, computational experiments are reported in Section 5. Finally, in Section 6, we give the conclusions.

2. Problem Description and Complexity

The proposed two-machine makespan flowshop scheduling problem denoted $T$ with processing time linearly dependent on job waiting time is described as follows.

First, some notations are introduced in the following and additional notations will be given when needed throughout the paper.

For a given instance of partition, $\sum_{i \in N} s_i = \sum_{i \in N-E} s_i = (\sum_{i \in N} s_i)/2$? For a given instance of partition, $s_1, s_2, \ldots, s_k$, an instance of the proposed problem is constructed as follows:

For a given instance of partition, $s_1, s_2, \ldots, s_k$, an instance of the proposed problem is constructed as follows:

\begin{align*}
&\sum_{i \in N} s_i = 2S, \\
&d_{k+1} = 1, b_{k+1} = 2 + 2S^2, d_{k+2} = 0; \\
&d_{k+2} = 2, b_{k+2} = 2S^2 - S, d_{k+2} = 2S^2 + S; \\
&\alpha_i = \alpha & \text{for } i \in \{1, 2, \ldots, k+2\},
\end{align*}

where $\sum_{i \in N} s_i = 2S$. 

We will show that Partition has a solution if and only if the above instance has an optimal schedule with the minimum makespan \( C_{\text{max}} = 4S^2 + S + 3 \).

**Lemma 1.** For the above instance, it is sufficient to consider the schedules that have the same job processing sequence on both machines for the job \( J_i, i \in \{1, \ldots, k + 1\} \).

**Proof.** If a schedule \( \Pi \) does not have the same job processing sequence on both machines for the job \( J_i, i \in \{1, \ldots, k + 1\} \), then there are two cases:

1. There is a job \( J_i \), which directly precedes \( J_j \) on machine \( M_1 \) and follows \( J_j \) on machine \( M_2 \), \( i, j \in \{1, \ldots, k + 1\} \), perhaps with intervening jobs (Figure 1a). We may interchange the order of \( J_i \) and \( J_j \) on machine \( M_1 \) without increasing the makespan, because \( d_i = d_j = 0 \).

2. There is a sequence \( (A_1, J_i, J_{i+2}, J_j, A_2) \) on machine \( M_1 \), where \( A_1 \) and \( A_2 \) are subsequences, and \( J_i \) follows \( J_j \) on machine \( M_2 \), perhaps with intervening jobs (Figure 1b). We may interchange the processing order of \( J_i \) and \( J_j \) in \( (A_1, J_i, J_{i+2}, J_j, A_2) \) as the following sequence \( (A_1, J_{i+2}, J_j, J_i, A_2) \) on machine \( M_1 \) without increasing the makespan (Figure 1c), because \( d_i = d_j = 0 \).

This process of interchanging jobs may be repeated until a schedule \( \Pi' \) is obtained with the same order on machine \( M_1 \) as that on machine \( M_2 \) for the job \( J_i, i \in \{1, \ldots, k + 1\} \). \( \Pi' \) is clearly not worse than \( \Pi \). Therefore, Lemma 1 holds. \( \square \)

**Lemma 2.** If \( J_{k+1} \) is not processed first, then the makespan of any schedule is greater than \( 4S^2 + S + 3 \).

**Proof.** For any given schedule, if \( J_i \) (\( i \in N \)) is processed first, then

\[
C_{\text{max}} \geq a_i + \sum_{i \in N} b_i + b_{i+1} + b_{i+2} = 2S_i + 2 + 2S^2 + 2S^2 - S > 4S^2 + S + 3.
\]
Similarly, if $J_{k+2}$ is processed first, then
\[
C_{\text{max}} \geq a_{k+2} + \sum_{i \in N} b_i + b_{k+1} + b_{k+2}
\]
\[
= 2 + 2S + 2 + 2S^2 + 2S^2 - S
\]
\[
> 4S^2 + S + 3.
\]

Thus, we only consider the schedules whose $J_{k+1}$ is processed first on both machines. □

Lemma 3. If $J_{k+2}$ is not processed second on machine $M_1$, then the makespan of any schedule is greater than $4S^2 + S + 3$.

**Proof.** Let $U = \{i\}$ is processed between $J_{k+1}$ and $J_{k+2}$ on machine $M_1$. If there are jobs $J_i, i \in U$, then the following two cases are considered:

(1) If any job $J_i, i \in N$, does not create any idle-time slot on machine $M_2$, that is, the processing of these jobs on machine $M_2$ is continuous, then there are two subcases:

(i) If $2S^2 + 2S - 2S \sum_{i \in U} s_i \geq 0$ (Figure 2), then
\[
C_{\text{max}} \geq a_{k+1} + \sum_{i \in N} b_i + l + b_{k+2} + \alpha \max\{0, d_{k+2} - w_{k+2}\},
\]
\[
= 1 + 2 + 2S^2 + 2S + l + 2S^2 - S + \alpha \max\{0, 2S^2 + S - 2S^2 - 2S + 2S \sum_{i \in U} s_i - l\}
\]
\[
= 4S^2 + S + 3 + l + \alpha \max\{0, (2 \sum_{i \in U} s_i - 1)S - l\} > 4S^2 + S + 3,
\]
where $w_{k+2} = b_{k+1} + \sum_{i \in U} b_i + l - \sum_{j \in U} a_j - a_{k+2} = 2S^2 + 2S - 2S \sum_{i \in U} s_i + l$ and $l \geq 0$.

(ii) If $2S^2 + 2S - 2S \sum_{i \in U} s_i < 0$, then $J_{k+2}$ creates an idle-time slot $l_{k+2} = 2S \sum_{i \in U} s_i - 2S^2 - 2S > 0$ on machine $M_2$. From Lemma 2, if $J_{k+1}$ is scheduled first and there is no idle time on machine $M_2$, except during operation 1 of $J_{k+1}$, there is a lower bound of the $C_{\text{max}}$, which is $4S^2 + S + 3$. In this case, $J_{k+2}$ creates an idle-time slot $l_{k+2} > 0$; therefore, $C_{\text{max}} > 4S^2 + S + 3$.

(2) Similarly, if a job $J_r, r \in N$, creates an idle-time slot $l_r > 0$, then $C_{\text{max}} > 4S^2 + S + 3$.

![Figure 2. Gantt diagram for the case that job $J_i$ is processed between $J_{k+1}$ and $J_{k+2}$.](image)

Thus, we only consider the schedules whose $J_{k+2}$ is processed second on machine $M_1$. □

**Theorem 1.** For any given positive number $\alpha > 0$, $\alpha_i = \alpha, i = 1, \ldots, n$, the two-machine makespan flowshop scheduling problem $P_{1\alpha|\sum c_i}$ is NP-hard.

**Proof.** If partition has a solution, then there is an optimal schedule with the makespan $C_{\text{max}} = 4S^2 + S + 3$ (Figure 3a). We note that, in this case, the waiting time $w_i \geq d_i$ for each $i$. 
If partition has no solution, we will show that the makespan of any schedule for the above instance is greater than \( 4S^2 + S + 3 \). For given schedule in which \( J_{k+1} \) is processed first on two machines and \( J_{k+2} \) is processed second on machine \( M_1 \), we let \( E = \{ j \mid j \text{ is processed between } J_{k+1} \text{ and } J_{k+2} \text{ on machine } M_2 \} \). By the assumption that partition has no solution and \( \sum_{i \in E} s_i - S = c \neq 0 \), we consider the following two cases:

1. If \( c < 0 \) and \( l > 0 \) (Figure 3b), we have

   \[
   \max = a_{k+1} + b_{k+1} + \sum_{i \in E} b_i + l + b_{k+2} + \alpha \max\{0, d_{k+2} - w_{k+2}\} + \sum_{i \in N\setminus E} b_i \\
   = 1 + 2 + 2S^2 + 2S + l + 2S^2 - S + \alpha \max\{0, d_{k+2} - w_{k+2}\} \\
   = 4S^2 + S + 3 + l + \alpha \max\{0, 2S^2 + S - 2S^2 + S + \sum_{i \in E} s_i + l\} \\
   = 4S^2 + S + 3 + l + \alpha \max\{0, -c - l\} > 4S^2 + S + 3,
   \]

   where \( w_{k+2} = b_{k+1} + \sum_{i \in E} b_i + l - a_{k+2} = 2S^2 + \sum_{i \in E} s_i + l \).

2. If \( c > 0 \) (\( c \geq 1/2 \), because \( s_1, s_2, \ldots, s_k \) are positive integers), without loss of generality, we let \( J_j, j \in E \), be the last processed job in \( E \) (Figure 3c), then there are two cases:

   (i) If any job \( J_i, i \in E \setminus \{j\} \), does not create any idle-time slot, then we will show that job \( J_j \) creates an idle-time slot \( l_j > 0 \) on machine \( M_2 \) (Figure 3c). The idle-time slot is

   \[
   l_j = \max\{0, a_{k+2} + \sum_{i \in E} a_i - b_{k+1} - \sum_{i \in E \setminus \{j\}} b_i\} \\
   = \max\{0, 2 + 2S\sum_{i \in E} s_i - 2S^2 - \sum_{i \in E} s_i + s_j\} \\
   = \max\{0, 2S(S + c) - 2S^2 - S - c + s_j\} \\
   = \max\{0, 2S(S - c + s_j)\} = \max\{0, 2(S - 1/2)(c - 1/2 - 1/2 + s_j)\} \\
   = 2(S - 1/2)(c - 1/2) - 1/2 + s_j > 0, \text{ since } S \geq 1, c \geq 1/2 \text{ and } s_j \geq 1.
   \]
Hence, $C_{\text{max}} > 4S^2 + S + 3$.

(ii) If a job $J_r, r \in E-\{j\}$, creates an idle-time slot $I_r > 0$, then $C_{\text{max}} > 4S^2 + S + 3$.

Thus, the makespan is greater than $4S^2 + S + 3$ if partition has no solution. It follows that partition has a solution if and only if the optimal schedule of the above instance has the minimum makespan $C_{\text{max}} = 4S^2 + S + 3$. □

3. A 0-1 Mixed Integer Programming Formulation

According to the similar analysis in Yang and Chern [23], a 0-1 mixed integer programming of the problem is developed as follows. First, we denote that

$A_i =$ the starting time of job $i$ on machine $M_1$;
$B_i =$ the starting time of job $i$ on machine $M_2$;
$w_i =$ the actual waiting time of job $i$ before its processing on machine $M_2 = B_i - A_i - a_i$;
$W_i = \max \{d_i - w_i, 0\} = \max \{d_i - B_i + A_i + a_i, 0\}$;
$y_{ijk} = 1,$ if job $i$ precedes job $j$ on machine $M_k$;
$y_{ijk} = 0,$ otherwise;
$C_{\text{max}} =$ the makespan.

For each job $i$, it is clear to have

$$B_i \geq A_i + a_i, \quad 1 \leq i \leq n. \quad (1)$$

In addition, it is necessary to assure that no two operations can be processed simultaneously by the same machine. Suppose, for example, that job $j$ precedes job $i$ on machine 1, then it is necessary to have

$$A_i \geq A_j + a_j.$$

On the other hand, if job $i$ precedes job $j$ on machine 1, then it is necessary to have

$$A_j \geq A_i + a_i.$$

These inequalities are called disjunctive constraints, because one and only one of these inequalities must hold. In order to accommodate these constraints in the formulation, these disjunctive constraints can be rewritten as follows:

$$A_i + My_{ij1} \geq A_j + a_j, 1 \leq i < j \leq n, \quad (2)$$
$$A_j + M(1 - y_{ij1}) \geq A_i + a_i, 1 \leq i < j \leq n, \quad (3)$$

where $M$ represents a sufficiently large positive number.

By the same way, these disjunctive constraints of job $i$ and job $j$ processed on machine 2 can be expressed as follows:

$$B_i + My_{ij2} \geq B_j + b_j + a_j W_i, 1 \leq i < j \leq n, \quad (4)$$
$$B_j + M(1 - y_{ij2}) \geq B_i + b_i + a_i W_i, 1 \leq i < j \leq n. \quad (5)$$

We note that

$$W_i = \max \{d_i - B_i + A_i + a_i, 0\},$$

then it is necessary to have $W_i \geq 0$ and

$$W_i \geq d_i - B_i + A_i + a_i, 1 \leq i \leq n. \quad (6)$$
For the makespan problem, it is necessary to have
\[ C_{\text{max}} \geq B_{i} + b_{i} + \alpha_{i}W_{i}, \quad 1 \leq i \leq n. \] 

Then, a disjunctive integer programming formulation of the proposed problem can be given by

\[
\begin{align*}
\text{minimize} & \quad C_{\text{max}} \\
\text{subject to} & \quad C_{\text{max}} \geq B_{i} + b_{i} + \alpha_{i}W_{i}, \quad 1 \leq i \leq n, \\
& \quad B_{i} \geq A_{i} + a_{i}, \quad 1 \leq i \leq n, \\
& \quad A_{i} + M_{ij(i)} \geq A_{j} + a_{j}, \quad 1 \leq i < j \leq n, \\
& \quad A_{i} + M_{ij(i)} \geq A_{j} + b_{i} + \alpha_{i}W_{i}, \quad 1 \leq i < j \leq n, \\
& \quad B_{i} + M_{ij(2)} \geq B_{j} + b_{i} + \alpha_{i}W_{i}, \quad 1 \leq i < j \leq n, \\
& \quad W_{i} \geq b_{i} - B_{i} + A_{i} + a_{i}, \quad 1 \leq i \leq n, \\
& \quad C_{\text{max}}, A_{i}, B_{i}, W_{i} \geq 0, \quad 1 \leq i \leq n, W_{ijk} = 0 \text{ or } 1, \quad 1 \leq i \leq j \leq n, 1 \leq k \leq 2.
\end{align*}
\]

The total number of type (1), (6), and (7) constraints is equal to 3n. The total number of type (2) and (3) constraints is equal to n(n - 1). The total number of type (4) and (5) constraints is equal to n(n - 1). Hence, the total number of constraints is n(2n + 1). There are 3n + 1 nonnegative variables of \( C_{\text{max}}, A_{i}, B_{i}, \) and \( W_{i}. \) We note that if \( y_{ijk} \) is in the formulation, then \( y_{ijk} \) needs to be defined. Hence, there are \( n(n - 1)/2 \) 0 - 1 integer variables of \( y_{ij} \) and the same number of \( y_{ij(2)}. \) The total number of variables is thus \( n(n + 2) + 1. \)

4. Heuristic Algorithm and Its Worst-Case Performance

As a reducible waiting time is considered in the proposed problem, somehow, the proposed problem is similar to the two-machine flowshop scheduling problem with start and stop lags. Therefore, we first use the Maggu and Das’s algorithm [27] to determine a sequence \( A \) (Algorithm 1).

**Algorithm 1.** Maggu and Das’s algorithm

1. \( \text{Step 1.} \) Let \( J = \{J_{1}, J_{2}, \ldots, J_{n}\}. \)
2. \( \text{Step 2.} \) Determine the job processing order in the following way:
   1. Decompose set \( J \) into the following two sets:
      \[ U = \{J_{i} \mid a_{i} \leq b_{i}\} \text{ and } V = \{J_{i} \mid a_{i} > b_{i}\}. \]
   2. Arrange the members of set \( U \) in nondecreasing order of \( a_{i} + d_{i}, \) and arrange the members of set \( V \) in nonincreasing order of \( b_{i} + d_{i}. \)
   3. A sequence \( A \) is the ordered set \( U \) followed by the ordered set \( V. \)

Some solvable cases are described in the following.

First, if \( \min_{1 \leq i \leq n} \{a_{i}\} \geq 1, \) the proposed problem is the same as that in Dell’Amico [17]. Therefore, if \( \max_{1 \leq i \leq n} \{d_{i}\} \leq \min_{1 \leq i \leq n} \{b_{i} + d_{i}\} \) or \( \max_{1 \leq i \leq n} \{d_{i}\} \leq \min_{1 \leq i \leq n} \{a_{i} + d_{i}\}, \) an optimal schedule is given by Maggu and Das’s Algorithm.

**Theorem 2** ([23]). In the problem, the case considered here is with \( \max_{1 \leq i \leq n} \{a_{i}\} \leq 1, \) \( \max_{1 \leq i \leq n} \{a_{i} + d_{i}\} \leq \min_{1 \leq i \leq n} \{b_{i} + d_{i}\}, \) and \( a_{c} + \alpha_{c}d_{c} = \min_{1 \leq i \leq n} \{a_{i} + \alpha_{i}d_{i}\}. \) If there is a job \( J_{j}, \) such that \( a_{j} + d_{j} \leq b_{c} + \alpha_{c}d_{c}, \) then \( (J_{i}, J_{j}, B) \) is an optimal schedule, where \( B \) is an arbitrary subsequence of the jobs without \( J_{i}, J_{j}. \)

**Proof.** Please see the proof in Appendix A. \( \Box \)

**Theorem 3** ([23]). If \( \max_{1 \leq i \leq n} \{a_{i}\} \leq 1, \) \( \min_{1 \leq i \leq n} \{a_{i}\} \geq \max_{1 \leq i \leq n} \{b_{i} + \alpha_{i}d_{i}\} \) and \( b_{c} + \alpha_{c}d_{c} = \min_{1 \leq i \leq n} \{b_{i} + \alpha_{i}d_{i}\}, \) then \( (B, J_{i}, J_{j}) \) is an optimal schedule, where \( B \) is an arbitrary subsequence of the jobs without \( J_{i}, \) and we schedule the jobs with no-wait manner as shown in Figure 4.
It is clear that a lower bound on the makespan is
\[
\sum_{i=1}^{n} a_i + \min_{1 \leq i \leq n} \{b_i + \alpha_i d_i\}.
\]
If \(\max_{1 \leq i \leq n} \{\alpha_i\} \leq 1\), then it is useless to reduce the waiting time. However, if \(0 < \alpha_i < 1\), then it is useful to reduce the waiting time as much as possible. A stepwise description of the algorithm is given as follows (Algorithm 2):

**Algorithm 2. Heuristic algorithm**

Step 0. (Initialization) Check those conditions stated in Theorem 2 and Theorem 3. If any one of conditions holds, the optimal schedule is obtained. Otherwise, generate a heuristic solution in the following steps.

Determine a sequence \(\pi\) by using Maggu and Das’s algorithm. Set \(k = 1\) and \(C_{[0],1} = C_{[0],2} = 0\).

Step 1. (Determining the reduced period of waiting time and the additional time on machine \(M_2\) for each job)

Set \(C_{[k],1} = C_{[k-1],1} + a_i\).

If \(C_{[k-1],2} - C_{[k-1],1} < d_i\) (Figure 5a), then go to Step 1.1.

If \(a_i \leq C_{[k-1],2} - C_{[k-1],1} \leq a_i + d_i\) (Figure 5b), then go to Step 1.2.

If \(a_i + d_i < C_{[k-1],2} - C_{[k-1],1}\) (Figure 5c), then go to Step 1.3.

1.1 If \(0 < \alpha_i < 1\), then \(C_{[k],2} = C_{[k],1} + b_i + \alpha_i d_i\).

If \(1 \leq \alpha_i\), then \(C_{[k],2} = C_{[k],1} + b_i\).

Go to Step 2.

1.2 If \(0 < \alpha_i < 1\), then \(C_{[k],2} = C_{[k-1],2} + b_i + \alpha_i (C_{[k],1} + d_i) - C_{[k-1],2}\).

If \(1 \leq \alpha_i\), then \(C_{[k],2} = C_{[k],1} + d_i + b_i\).

Go to Step 2.

1.3 \(C_{[k],2} = C_{[k-1],2} + b_i\). Go to Step 2.

Step 2. Set \(k = k + 1\)

If \(k \leq n\), go to Step 1. Otherwise, stop.

In the heuristic algorithm, Step 0 first determines a sequence \(\pi\) by using Maggu and Das’s algorithm. Step 1 adjusts the reduced period of waiting time and the additional time on machine 2 for each job. In Step 1.1 and Step 1.2, if \(\alpha_i \geq 1\), then it is useless to reduce the waiting time. However, if \(1 > \alpha_i > 0\), then it is useful to reduce the waiting time as much as possible and the calculation of the
Algorithm, Step 0 first determines a sequence \( \{0\} \), then it is useful to reduce the waiting time as much as possible. Because all the jobs have to be processed simultaneously on both machines, the completion times on both machines is given. Step 1.3 depicts that if \( a_k \geq 1 \) or \( 1 > \alpha_k > 0 \), then it is useless to reduce the waiting time either \( \alpha_k \). If \( 0 < \alpha_k < 1 \), then it is useful to reduce the waiting time either \( \alpha_k \). If \( \alpha_k \geq 1 \) or \( \alpha_k = 1 \), then it is useless to reduce the waiting time either \( \alpha_k \).

![Figure 5. Gantt diagrams for three cases in the heuristic algorithm.](image)

In the following, an example of five jobs is given to illustrate the heuristic algorithm.

**Example 1.** There are five jobs; that is, \( J_1, J_2, J_3, J_4, \) and \( J_5 \), to be processed on machine \( M_1 \) and machine \( M_2 \). The processing times on these jobs on machine \( M_1 \) are \( a_1 = 1, a_2 = 3, a_3 = 2, a_4 = 3, \) and \( a_5 = 2 \), respectively. The processing times on machine \( M_2 \) are \( b_1 = 5, b_2 = 4, b_3 = 1, b_4 = 2, \) and \( b_5 = 3 \), respectively. The delay times required before its processing on machine \( M_2 \) are \( d_1 = 2, d_2 = 3, d_3 = 1, d_4 = 2, \) and \( d_5 = 5 \), respectively. The cost indices are \( \alpha_1 = 0.2, \alpha_2 = 0.3, \alpha_3 = 0.1, \alpha_4 = 0.5, \) and \( \alpha_5 = 1.2, \) respectively.

According to the above heuristic algorithm, we obtain that the makespan of these five jobs is 16.6. Please see the details in Appendix B.

**Lower bound**

First, we assume that processing on each machine may be continuous, \( D_1 = \{ i \mid \alpha_i \geq 1 \} \) and \( D_2 = N - D_1 \). We can see that, if \( \alpha_i \geq 1 \), then it is useless to reduce the waiting time. However, if \( 1 > \alpha_i > 0 \), then it is useful to reduce the waiting time as much as possible. Because all the jobs have to be processed on machine \( M_1 \) and \( M_2 \), if the delay time is short relative to the corresponding processing time, we have an immediate lower bound \( LBI = \max \{ \sum_{i=1}^{n} a_i + \min_{i \in D_1} \{ \min \{d_i + b_i\}, \min_{i \in D_2} \{\alpha_i d_i + b_i\}\}, \min_{i \in D_1} \{\alpha_i d_i + b_i\} + \sum_{i=1}^{n} b_i\} \). On the other hand, if one of the delay times is long enough relative to
In this case, \( q_i \geq 1 \) and \( C_i + v_i = b_i \), where \( v_i < a_i \).

In this case, \( q_i < 1 \) and \( C_i + v_i = b_i \), where \( v_i < a_i \).

In this case, \( q_i < 1 \) and \( C_i + v_i = b_i \), where \( v_i = 0 \).

Figure 6. Gantt diagrams for three cases of the critical job.

In the following, we will find the worst case of the heuristic algorithm. First, some notations are defined. Given a solution of the proposed problem, we define the critical job with maximum starting time on machine \( M_i \) as \( J_i \), such that \( \max_d d_i = b_i \). \( d_i \) and \( b_i \) are the starting and the completion time of \( J_i \) on machine \( M_i \). Then, considering the schedule in which the jobs are arranged by Maggu and Das's algorithm, we have the following results.

Case 1. \( 0 < q_i < 1 \) and \( C_i < C_i < 1 \). Hence, a lower bound is calculated as follows:

\[
LB_1 = \frac{1}{\rho} \left( \sum_{i} a_i + \sum_{j=1}^{n} b_j \right)
\]

Figure 6. Gantt diagrams for three cases of the critical job.
Theorem 5. Let \( \pi^* \) be an optimal schedule and \( \pi \) be the Maggu and Das’s schedule for the heuristic algorithm. If \( 0 < \alpha_c < 1 \) and \( C_{i-1,2} \geq C_{i-1,2} \), then \( \rho = \frac{C_{\text{max}}(\pi)}{C_{\text{max}}(\pi^*)} \leq 2 \) and the bound is tight.

Proof. Considering the schedule in which the jobs are arranged by Maggu and Das’s algorithm, if \( 0 < \alpha_c < 1 \) and \( C_{i-1,2} \geq C_{i-1,2} \), then the makespan is as follows:

\[
C_{\text{max}}(\pi) = \sum_{i\in F} a_i + a_c + \sum_{i\in F} b_i
\]

Therefore, if \( a_c \leq b_c \) the jobs \( J_i \in F \) have the property of \( a_i \leq b_i \). Then, we have the following results:

\[
C_{\text{max}}(\pi) \leq (\alpha_c d_c + b_c) + \sum_{i=1}^{n} b_i \leq \max\{a + \alpha_c d_c + b_i\} + \min\{a + d_i\},
\]

\[
\min\{a + \alpha_c d_i\} + \sum_{i=1}^{n} b_i \leq LB1 + LB2 \leq 2C_{\text{low}}.
\]

Similarly, if \( a_c > b_c \), the jobs \( J_i \in F \) have the property of \( a_i > b_i \). Thus,

\[
C_{\text{max}}(\pi) \leq (a_c + \alpha_c d_c) + \sum_{i=1}^{n} a_i \leq \max\{a + \alpha_c d_i + b_i\} + \sum_{i=1}^{n} a_i + \min\{d_i\}
\]

\[
+ b_i, \min\{a + d_i + b_j\} \leq LB2 + LB1 \leq 2C_{\text{low}}.
\]

Hence, \( \rho = \frac{C_{\text{max}}(\pi)}{C_{\text{max}}(\pi^*)} \leq C_{\text{max}}(\pi) / C_{\text{low}} \leq 2 \).

To prove the tightness of the bound, consider \( J_1 \) with \( a_1 = 1, b_1 = 1, d_1 = n, \) and \( 0 < \alpha_1 < 1 \), and \( J_i (i = 2, 3, \ldots, n) \) with \( a_i = 1, b_i = 1, d_i = 0, \) and \( \alpha_i = 0.5 \). By applying Maggu and Das’s algorithm, the job sequence is \( J_2, J_3, \ldots, J_n, J_1 \) on both machine \( M_1 \) and \( M_2 \). Then, \( C_{\text{max}}(\pi) = n + \alpha_1 n + 1 = (1 + \alpha_1)n + 1 < 2n + 1 \).

On the other hand, an optimal schedule can be obtained by arranging the job sequence \( J_1, J_2, J_3, \ldots, J_n \) on machine \( M_1 \) and the job sequence \( J_2, J_3, \ldots, J_n, J_1 \) on machine \( M_2 \). Then, \( C_{\text{max}}(\pi^*) = n + 2 \). Hence, \( \rho = \frac{C_{\text{max}}(\pi)}{C_{\text{max}}(\pi^*)} = (2n + 1) / (n + 2) \), which tends to 2 as \( n \to \infty \). \( \square \)

Theorem 5. Let \( \pi^* \) be an optimal schedule and \( \pi \) be the Maggu and Das’s schedule for the heuristic algorithm. If \( \alpha_c \geq 1 \), then \( \rho = \frac{C_{\text{max}}(\pi)}{C_{\text{max}}(\pi^*)} \leq 2 \) and the bound is tight.

Proof. The proof is the similar to that of Theorem 4. Thus, we omit it here. \( \square \)

5. Computational Experiments

Although we find the worst case of the heuristic algorithm under some certain situations (case 1 and case 3), the upper bound of case 2 is still unknown. Therefore, in order to evaluate the overall efficiency of the heuristic algorithm, we generate several groups of random problems as follows:
In the computational experiment, a total of 720 test problems are generated. The computation times of algorithms for all the test problems are within one second. For each of these random problems, the percentage of the error $e = (C_h - C_{\text{low}}) \times 100/C_{\text{low}}$ is computed, where $C_h$ is the makespan of the heuristic solution and $C_{\text{low}}$ is the lower bound on the makespan.

The result is given in Table 1. There are 20 test problems for each problem type. To evaluate the overall performance of the heuristic algorithm, we compute the mean of all the average percentage errors reported in Table 1. The mean value is 1.97%, which suggests that the heuristic algorithm, on average, finds schedules that are within 1.97% of optimality. From Theorem 4 and 5, we can see that the upper bound of the heuristic algorithm is $2 \cdot C_{\text{low}}$; therefore, the performance of the heuristic algorithm is quite satisfactory. From Figure 7, the larger the value of $d_i$, the greater the percentage of the error. Because the proposed heuristic algorithm is restricted to searching a near-optimal permutation schedule, it may imply that the optimal schedule is likely to be a non-permutation schedule when the values of delay times of jobs are larger. Therefore, the performance of the heuristic algorithm is better when $d_i$ is smaller. We can also see that the average percentage of errors decreases as the job number $n$ increases for different values of $d_i$. Especially, when the job number $n$ is more than 100, the mean value of the average percentage of errors is less than 1%. In view of the NP-hardness of the problem, this result is quite encouraging as it provides efficient procedures for solving large-sized problems.

![Figure 7. The average percentages of errors in the computational experiments.](image-url)
Table 1. Computational results for Algorithm 1.

| n   | d_i | Minimum Percentage Error (%) | Average Percentage Error (%) | Maximum Percentage Error (%) |
|-----|-----|-------------------------------|------------------------------|------------------------------|
|     |     | Observed in 20 Test Problems  |                              |                              |
| 10  | [0,50] | 0                             | 3.8172                       | 10.6325                      |
|     | [50,100] | 0                             | 5.9864                       | 11.8965                      |
|     | [100,200] | 0                             | 8.1236                       | 19.3263                      |
| 20  | [0,50] | 0                             | 2.7115                       | 5.3269                       |
|     | [50,100] | 0                             | 2.9658                       | 10.1268                      |
|     | [100,200] | 1.2547                       | 5.1579                       | 9.8852                       |
| 30  | [0,50] | 0                             | 1.2363                       | 3.9986                       |
|     | [50,100] | 0                             | 2.6636                       | 6.2355                       |
|     | [100,200] | 3.6278                       | 5.6986                       | 12.3023                      |
| 40  | [0,50] | 0.1321                       | 2.1506                       | 5.3330                       |
|     | [50,100] | 0.1056                       | 1.5367                       | 3.7795                       |
|     | [100,200] | 1.2233                       | 3.9688                       | 7.4862                       |
| 50  | [0,50] | 0                             | 0.9865                       | 1.2296                       |
|     | [50,100] | 0.1007                       | 1.2233                       | 3.0125                       |
|     | [100,200] | 1.0023                       | 3.0559                       | 6.7756                       |
| 60  | [0,50] | 0.0553                       | 1.0021                       | 2.9140                       |
|     | [50,100] | 0.3054                       | 0.9553                       | 1.2235                       |
|     | [100,200] | 0.9562                       | 2.0013                       | 3.9240                       |
| 70  | [0,50] | 0.0342                       | 1.0782                       | 2.6835                       |
|     | [50,100] | 0.1602                       | 0.9866                       | 2.2012                       |
|     | [100,200] | 0.6058                       | 2.6158                       | 3.9671                       |
| 80  | [0,50] | 0                             | 0.2212                       | 1.2369                       |
|     | [50,100] | 0.2365                       | 0.9696                       | 2.0123                       |
|     | [100,200] | 0.5523                       | 1.2633                       | 2.1825                       |
| 90  | [0,50] | 0.0327                       | 0.3223                       | 1.0023                       |
|     | [50,100] | 0.2305                       | 0.6696                       | 1.2354                       |
|     | [100,200] | 0.6813                       | 2.0362                       | 3.7182                       |
| 100 | [0,50] | 0.0173                       | 0.1211                       | 1.2310                       |
|     | [50,100] | 0.0235                       | 0.9071                       | 2.3261                       |
|     | [100,200] | 0.2103                       | 1.5981                       | 3.6682                       |
| 150 | [0,50] | 0                             | 0.4335                       | 0.8908                       |
|     | [50,100] | 0.0636                       | 0.3693                       | 1.0032                       |
|     | [100,200] | 0.1015                       | 1.1036                       | 1.9936                       |
| 200 | [0,50] | 0                             | 0.1036                       | 0.3679                       |
|     | [50,100] | 0.3012                       | 0.4250                       | 0.5587                       |
|     | [100,200] | 0.2963                       | 0.3972                       | 0.8572                       |

6. Conclusions

In this paper, we investigate a two-machine flowshop scheduling problem in which the processing times of the second operations are linearly dependent on the waiting times of the jobs. This problem is shown to be NP-hard. A 0-1 integer programming and an efficient heuristic algorithm with computational experiment are also proposed. In addition, the worst case of the heuristic algorithm under some situations is provided. From the computational experiments, the overall performance of the proposed algorithm is quite satisfactory, especially when the job number is large. Some cases solved in polynomial time are also provided.

There are some limitations in this study. For example, the proposed heuristic algorithm only searches a near-optimal permutation schedule. In the future research, to develop a heuristic algorithm to find both permutation and non-permutation schedules may improve this scheduling performance (makespan). In addition, another performance measure, such as total tardiness or maximum lateness, may be considered for the proposed problem. Actually, in real situations, the cost of the reduction of
the waiting time may be non-linear or the compression of the waiting time may be limited. Therefore, future research may focus on issues, too.

**Author Contributions:** Conceptualization, D.-L.Y.; Methodology, D.-L.Y. and W.-H.K.; Software, W.-H.K.; Validation, D.-L.Y. and W.-H.K.; Formal Analysis, D.-L.Y. and W.-H.K.; Investigation, W.-H.K.; Resources, D.-L.Y.; Data Curation, W.-H.K.; Writing-Original Draft Preparation, W.-H.K.; Writing-Review & Editing, W.-H.K.; Visualization, D.-L.Y.; Supervision, D.-L.Y.; Project Administration, D.-L.Y.; Funding Acquisition, D.-L.Y.

**Funding:** This research was partially supported by the Ministry of Science and Technology of the Republic of China under grant MOST 103-2221-E-150-028.

**Acknowledgments:** The authors wish to thank the anonymous reviewers for their helpful comments on an earlier version of this paper.

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

---

**Appendix A**

**Theorem 2.** In the problem, the case considered here is with \( \max \{\alpha_i\} \leq 1 \), \( \max \{a_i + d_i\} \leq \min \{b_i + d_i\} \), and \( a_i + \alpha_i d_i = \min \{a_i + \alpha_i d_i\} \). If there is a job \( J_i \), such that \( a_i + d_i \leq b_{i^*} + \alpha_i d_i \), then \( (J_{i^*}, J_i, B) \) is an optimal schedule, where \( B \) is an arbitrary subsequence of the jobs without \( J_{i^*}, J_i \).

**Proof.** Without loss of the generality, let \( J_{i^*} = J_1 \) and \( J_i = J_2 \). It is clear that a lower bound on the makespan is \( \min \{a_i + \alpha_i d_i\} + \sum_{i=1}^{n} b_i \). In the following, we will show that if \( \max \{\alpha_i\} \leq 1 \), \( \max \{a_i + d_i\} \leq \min \{b_i + d_i\} \), and \( a_i + d_i \leq b_{i^*} + \alpha_i d_i \), then the makespan of \( (J_{i^*}, J_i, B) \), as shown in Figure A1, is equal to the lower bound \( a_{i^*} + \alpha_{i^*} d_{i^*} + \sum_{i=1}^{n} b_i \). Hence, \( (J_{i^*}, J_i, B) \) is an optimal schedule.

![Figure A1. Optimal schedule for the case that \( \max \{\alpha_i\} \leq 1 \), \( \max \{a_i + d_i\} \geq \min \{b_i + d_i\} \), \( a_i + b_i \leq b_{i^*} + \alpha_{i^*} d_{i^*} \), and \( B = \{J_1, \ldots, J_n\} - \{J_{i^*} - J_i\} \).](image)

The constraint of \( \max \{\alpha_i\} \leq 1 \) implies that it is worth reducing the delay time of a job if the finishing time of the job on machine \( M_2 \) can be earlier. However, in such a case, only the delay time of the first job in the sequence is worthy to be reduced. Therefore, the constraint of \( a_i + d_i \leq b_{i^*} + \alpha_{i^*} d_{i^*} \), guarantees that the finishing time of \( J_i \) on machine \( M_1 \) plus the delay time of the job is earlier than the finishing time of \( J_{i^*} \) (the first job) on machine \( M_2 \). The constraint of \( \max \{a_i + d_i\} \leq \min \{b_i + d_i\} \) guarantees the finishing times of jobs (except for \( J_{i^*} \) and \( J_i \)) on machine \( M_1 \), plus the delay times of these jobs are earlier than the finishing time of their previous jobs on machine \( M_2 \). It also implies that \( a_i \leq b_i \) for \( i = 1, 2, \ldots, n \).

We proceed by induction. First, we can obtain that the finishing time of \( J_{i^*} \) on machine \( M_2 \) is \( (a_{i^*} + b_{i^*} + \alpha_{i^*} d_{i^*}) \). The finishing time of \( J_i \) (the second job) on machine \( M_1 \) plus the delay time of the job is \( (a_{i^*} + a_i + d_i) \). If \( (a_i + d_i) \leq (b_{i^*} + \alpha_{i^*} d_{i^*}) \), then \( (a_{i^*} + a_i + d_i) \leq (a_{i^*} + b_{i^*} + \alpha_{i^*} d_{i^*}) \). This means that the finishing time of \( J_i \) (the second job) on machine \( M_1 \) plus the delay time of the job is earlier than the finishing time of \( J_{i^*} \) on machine \( M_2 \). There is no room for the reduction of the actual waiting time of \( J_i \). Therefore, job \( J_i \) can be processed on machine \( M_2 \) immediately after job \( J_{i^*} \) is finished on machine \( M_2 \). Then, the
completion time of job $J_i$ on machine $M_2$ is $a_i + \alpha_i d_i + b_i + b_j$. Therefore, the case $m = 2$ ($J_j = J_2$) is true.

If the $m$th case is assumed to be true, that is, $(a_i + a_j + \sum_{k=3}^{m} a_k + d_m) \leq (a_i + b_i + a_i d_i + b_j + \sum_{k=3}^{m-1} b_k)$, and the completion time of the $m$th job on machine $M_2$ is $(a_i + b_i + a_i d_i + b_j + \sum_{k=3}^{m-1} b_k + b_m)$, then we show that the $(m+1)$st case is also true.

The finishing time of the $(m+1)$st job (say $J_{m+1}$) on machine $M_1$ plus the delay time of the job is

$$ A = (a_i + a_j + \sum_{k=3}^{m} a_k + d_m) + (a_{m+1} + d_{m+1}) \leq (a_i + a_j + \sum_{k=3}^{m} a_k + d_m + a_j + d_j) + (a_{m+1} + d_{m+1}) = (a_i + b_i + a_i d_i + b_j + \sum_{k=3}^{m} b_k + (b_{m+1} + d_{m+1} + d_{m+1}) = (a_i + b_i + a_i d_i + b_j + \sum_{k=3}^{m} b_k).$$

This completes the proof.

$$\square$$

**Appendix B**

**Example.** There are five jobs; that is, $J_1, J_2, J_3, J_4$, and $J_5$, to be processed on machine $M_1$ and machine $M_2$. The processing times of these jobs on machine $M_1$ are $a_1 = 1, a_2 = 3, a_3 = 2, a_4 = 3$, and $a_5 = 2$, respectively. The processing times on machine $M_2$ are $b_1 = 5, b_2 = 4, b_3 = 1, b_4 = 2$, and $b_5 = 3$, respectively. The delay times required before its processing on machine $M_2$ are $d_1 = 2, d_2 = 3, d_3 = 1, d_4 = 2$, and $d_5 = 5$, respectively. The cost indices are $\alpha_1 = 0.2, \alpha_2 = 0.3, \alpha_3 = 0.1, \alpha_4 = 0.5$, and $\alpha_5 = 1.2$, respectively.

**Step 0.** Because $\max_{1 \leq i \leq 5} a_i = \alpha_5 = 1.2 > 1$, Theorem 2 and Theorem 3 cannot apply to this example. A sequence $\pi$ using Maggu and Das’s algorithm is determined as follows: $U = \{J_1 J_2 J_3\}$ and $V = \{J_4 J_5\}$. Therefore, sequence $\pi = (J_1 J_2 J_3 J_4 J_5) = (J_1 J_2 J_3 J_4 J_5)$. Set $k = 1$ and $C_{[0],1} = C_{[0],2} = 0.$

**Step 1.** ($k = 1$)

Set $C_{[1],1} = C_{[1],1} + a_1 = C_{[0],1} + a_1 = 0 + 1 = 1.$

Set $C_{[1],2} = C_{[1],1} + a_1 = 0 < a_1 = 1.$ Go to Step 1.1.

**Step 1.1.** $0 < a_1 < 1.$

Set $C_{[2],1} = C_{[1],1} + b_1 = C_{[1],1} + b_1 + \alpha_1 d_1 = C_{[2],1} + b_1 + \alpha_1 d_1 = 1 + 5 + (0.2)(2) = 6.4.$

Go to Step 2.

**Step 2.** Set $k = k + 1 = 2 \leq 5$. Go to Step 1.

**Step 1.** ($k = 2$)

Set $C_{[2],1} = C_{[1],1} + a_2 = 1 + 3 = 4.$

Set $C_{[2],2} = C_{[1],2} + b_2 = 6.4 - 1 = 5.4 < a_2 = 3.$ Go to Step 1.1.

**Step 1.1.** $0 < a_2 < 1.$

Set $C_{[3],1} = C_{[2],1} + b_2 + \alpha_2 d_2 = 4 + 4 + (0.3)(3) = 8.9.$
Go to Step 2.

**Step 2.** Set $k = k + 1 = 3 \leq 5$. Go to Step 1.

**Step 1.** ($k = 3$)

Set $C_{[3],1} = C_{[3],1} + a_{[3]} = 4 + 2 = 6$.

$a_{[3]} = 2 < C_{[2],2} - C_{[2],1} = 8.9 - 4 = 4.9 < a_{[3]} + d_{[3]} = 2 + 5 = 7$. Go to Step 1.2.

**Step 1.2.** $\alpha_{[3]} = 1.2 > 1$.

$C_{[3],2} = C_{[3],1} + d_{[3]} + b_{[3]} = 6 + 5 + 3 = 14$.

Go to Step 2.

**Step 2.** Set $k = k + 1 = 4 \leq 5$. Go to Step 1.

**Step 1.** ($k = 4$)

Set $C_{[4],1} = C_{[4],1} + a_{[4]} = 6 + 3 = 9$.

$C_{[3],2} - C_{[3],1} = 14 - 6 = 8 > a_{[4]} = 3$. Go to Step 1.3.

**Step 1.3.** $C_{[4],2} = C_{[3],2} + b_{[4]} = 14 + 2 = 16$.

Go to Step 2.

**Step 2.** Set $k = k + 1 = 5 \leq 5$. Go to Step 1.

**Step 1.** ($k = 5$)

Set $C_{[5],1} = C_{[4],1} + a_{[5]} = 9 + 2 = 11$.

$a_{[5]} = 2 < C_{[4],2} - C_{[4],1} = 16 - 9 = 7$. Go to Step 1.2.

**Step 1.2.** $0 < \alpha_{[5]} = 0.1 < 1$.

$C_{[5],2} = C_{[4],2} + b_{[5]} + \alpha_{[5]} \cdot (C_{[5],1} + d_{[5]} - C_{[4],2}) = 16 + 1 + 0.1(11 + 1 - 16) = 16.6$.

Go to Step 2.

**Step 2.** Set $k = k + 1 = 6 > 5$. Stop

Therefore, the makespan of these five jobs is $C_{[5],2} = 16.6$.

**References**

1. Hodson, A.; Muhlemann, A.; Price, D. A microcomputer based solution to a practical scheduling problem. *J. Oper. Res. Soc.* 1985, 36, 903–914. [CrossRef]
2. Chen, S.-H.; Liou, Y.-C.; Chen, Y.-H.; Wang, K.-C. Order Acceptance and Scheduling Problem with Carbon Emission Reduction and Electricity Tariffs on a Single Machine. *Sustainability* 2019, 11, 5432. [CrossRef]
3. Giret, A.; Trentesaux, D.; Prabhu, V. Sustainability in manufacturing operations scheduling: A state of the art review. *J. Manuf. Syst.* 2015, 37, 126–140. [CrossRef]
4. Gong, L.; Li, Y.; Xu, D. Combinational Scheduling Model Considering Multiple Vehicle Sizes. *Sustainability* 2019, 11, 5144. [CrossRef]
5. Liu, Y.; Liao, X.; Zhang, R. An Enhanced MOPSO Algorithm for Energy-Efficient Single-Machine Production Scheduling. *Sustainability* 2019, 11, 5381. [CrossRef]
6. Shuue, F-J.; Zheng, M.-C.; Lee, H.-Y.; Khitam, A.F.K.; Li, P.-Y. Renovation Construction Process Scheduling for Long-Term Performance of Buildings: An Application Case of University Campus. *Sustainability* 2019, 11, 5542. [CrossRef]
7. Theophilus, O.; Dulebenets, M.A.; Pasha, J.; Abiye, O.F.; Kavoosi, M. Truck Scheduling at Cross-Docking Terminals: A Follow-Up State-Of-The-Art Review. *Sustainability* 2019, 11, 5245. [CrossRef]
8. Torkjazi, M.; Huynh, N. Effectiveness of Dynamic Insertion Scheduling Strategy for Demand-Responsive Paratransit Vehicles Using Agent-Based Simulation. *Sustainability* 2019, 11, 5391. [CrossRef]
9. Johnson, S.M. Optimal two-and three-stage production schedules with setup times included. *Nav. Res. Logist. Q.* 1954, 1, 61–68. [CrossRef]
10. Mitten, L. Sequencing n jobs on two machines with arbitrary time lags. *Manag. Sci.* 1959, 5, 293–298. [CrossRef]

11. Sule, D.R.; Huang, K.Y. Sequence on two and three machines with setup, processing and removal times separated. *Int. J. Prod. Res.* 1983, 21, 723–732. [CrossRef]

12. Maggu, P.L.; Smghal, M.L.; Mohammad, N.; Yadav, S.K. On n-job, 2-machine flow-shop scheduling problem with arbitrary time lags and transportation times of jobs. *J. Oper. Res. Soc. Jpn.* 1982, 25, 219–227. [CrossRef]

13. Ruiz, R.; Stützle, T. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *Eur. J. Oper. Res.* 2008, 187, 1143–1159. [CrossRef]

14. Nishi, T.; Hirakawa, Y. Lagrangian relaxation and cut generation for sequence-dependent setup time flowshop scheduling problems to minimise the total weighted tardiness. *Int. J. Prod. Res.* 2013, 51, 4778–4796. [CrossRef]

15. Wang, Y.; Li, X.; Ma, Z. A Hybrid Local Search Algorithm for the Sequence Dependent Setup Times Flowshop Scheduling Problem with Makespan Criterion. *Sustainability* 2017, 9, 2318. [CrossRef]

16. Reddi, S.; Ramamoorthy, C. On the flow-shop sequencing problem with no wait in process. *J. Oper. Res. Soc.* 1972, 23, 323–331. [CrossRef]

17. Dell’Amico, M. Shop problems with two machines and time lags. *Oper. Res.* 1996, 44, 777–787. [CrossRef]

18. Yu, W.; Hoogeveen, H.; Lenstra, J.K. Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard. *J. Sched.* 2004, 7, 333–348. [CrossRef]

19. Fiszman, S.; Mosheiov, G. Minimizing total load on a proportionate flowshop with position-dependent processing times and job-rejection. *Inf. Process. Lett.* 2018, 132, 39–43. [CrossRef]

20. Yang, D.-L.; Chern, M.-S. A two-machine flowshop scheduling problem with limited waiting time constraints. *Comput. Ind. Eng.* 1995, 28, 63–70. [CrossRef]

21. Su, L.-H. A hybrid two-stage flowshop with limited waiting time constraints. *Comput. Ind. Eng.* 2003, 44, 409–424. [CrossRef]

22. Sriskandarajah, C.; Goyal, S. Scheduling of a two-machine flowshop with processing time linearly dependent on job waiting-time. *J. Oper. Res. Soc.* 1989, 40, 907–921. [CrossRef]

23. Yang, D.-L.; Chern, M.-S. A generalized two-machine flowshop scheduling problem with processing time linearly dependent on job waiting-time. *Comput. Ind. Eng.* 1999, 36, 365–378. [CrossRef]

24. Chung, T.-P.; Sun, H.; Liao, C.-J. Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint. *Comput. Ind. Eng.* 2017, 113, 859–870. [CrossRef]

25. Wang, B.; Huang, K.; Li, T. Permutation flowshop scheduling with time lag constraints and makespan criterion. *Comput. Ind. Eng.* 2018, 120, 1–14. [CrossRef]

26. Johnson, D.; Garey, M. A Guide to the Theory of NP-Completeness. *Computers and Intractability*, WH Freeman and Company: New York, NY, USA, 1979.

27. Maggu, P.L.; Das, G. On 2xn sequencing problem with transportation times of jobs. *Pure Appl. Math. Sci.* 1980, 12, 1–6.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).