Two-machine flowshop scheduling problems involving a batching machine with transportation or deterioration consideration

Lixin Tang *, Peng Liu

The Logistics Institute, Northeastern University, Shenyang 110004, China

Received 19 November 2006; received in revised form 15 January 2008; accepted 17 January 2008
Available online 2 February 2008

Abstract

This paper considers two scheduling problems for a two-machine flowshop where a single machine is followed by a batching machine. The first problem is that there is a transporter to carry the jobs between machines. The second problem is that there are deteriorating jobs to be processed on the single machine. For the first problem with minimizing the makespan, we formulate it as a mixed integer programming model and then prove that it is strongly NP-hard. A heuristic algorithm is proposed for solving this problem and its worst case performance is analyzed. The computational experiments are carried out and the numerical results show that the heuristic algorithm is effective. For the second problem, we derive the optimal algorithms with polynomial time for minimizing the makespan, the total completion time and the maximum lateness, respectively.

© 2008 Elsevier Inc. All rights reserved.

Keywords: Flowshop scheduling; Batching machine; Transportation; Deterioration; Heuristic

1. Introduction

Flowshop scheduling has been a popular topic of scheduling research for many years. For the flowshop, there are \( n \) jobs to be processed on \( m \) machines in series, and each job has \( m \) operations, the first of which requires processing on machine 1, the second requires processing on machine 2, and so on, and the last requires processing on machine \( m \). In the flowshop, all jobs follow the same machine sequence and each job has exactly one operation on each machine. In this paper, we study two-machine flowshop scheduling where a single machine is followed by a batching machine which can process a batch of jobs simultaneously. There are two situations to be considered. The first is that there is a transporter to carry the jobs between machines. Most of machine scheduling models in the published literature assume that either transportation capacity of transporters for delivering jobs is unlimited or transportation times from one location to another...
are negligible. There is limited research on machine scheduling problem with explicitly transportation considerations. The second is that there are deteriorating jobs to be processed on the single machine. Unlike classical scheduling problems, deterioration of a job means that its processing time is a function of its starting time instead of constant.

This scheduling problem arises from the ingot teeming and heating process in the steel plant (see Fig. 1). Molten steel in the ladle is poured into moulds to produce steel ingots, and this operation is called teeming. After the ingots are stripped from the moulds, they are appropriately segregated into batches and next transported to the soaking pit. The soaking pit reheats several ingots throughout to a sufficiently high temperature and can be regarded as a batching machine. Ingots are required for heating to the desirable temperature before rolling on the rough mill. The temperature of the ingots will drop as waiting time increases. If ingots have inevitable drop in the temperature, then ingots are required more additional heating time to reach the rolling temperature. In the process described above, there exists transportation as well as deterioration phenomenon.

Scheduling problems with batching, transportation and deterioration have been studied separately in the most of previous literatures. We consider not only the two-machine flowshop scheduling problem with batching on the second machine, but also the scheduling problem with transportation times between machines or with deteriorating jobs on the first machine. We will present a brief review on the related scheduling problems as follows. There are reviews of models which combine scheduling with batching by Webster and Baker [1] and Potts and Kovalyov [2]. Ahmadi et al. [3] investigate a class of scheduling problems defined in a two- or three-machine flowshop with at least one batch processor incorporated. They study the complexity analysis for two problem instances with the objective of makespan and total completion time. Sung and Kim [4] extend the two-machine flowshop problem with makespan measure of Ahmadi et al. [3] to the situation of allowing dynamic arrival times at the discrete processor. They show this problem is strongly NP-hard. Our work differs from the models of Ahmadi et al. [3] and Sung and Kim [4] in that we consider the scheduling problem with a transporter to carry the jobs between machines or with deteriorating jobs to be processed on the first machine.

A few related machine scheduling models with transportation considerations have been studied in the literatures. Lee and Chen [5] and Hurink and Knust [6] study machine scheduling problems with the transportation of jobs between machines in a flowshop environment where jobs are transported from one machine to another for further processing. Both transportation capacity and transportation times are taken into account in their models. The literature concerning the scheduling model with transportation considerations focusing on the transportation of finished jobs to customers can be found in Lee and Chen [5], Soukhal et al. [7], and Li et al. [8]. Li and Ou [9] study a single machine scheduling model that incorporates the scheduling of jobs and the pickup and delivery arrangements of the materials and finished jobs. The literatures mentioned above cover the scheduling with transportation, however, they do not consider the scheduling with batching.

In another related paper, machine scheduling problems with deteriorating processing times have received increasing attention in recent years. An extensive survey of different models and problems concerning starting time-dependent job processing times can be found in Alidaee and Womer [10] and Cheng et al. [11]. Mosheiov [12] considers the single machine scheduling problems of simple linear deterioration of processing times. For this case, he is able to find the optimal solution. Mosheiov [13] shows that Johnson’s rule (Johnson [14]) can be adapted to the two machine flowshop minimizing the makespan scheduling problem under simple linear deterioration of processing times. Zhao et al. [15] consider the scheduling problems under increasing linear deterioration of processing times. Some optimal algorithms are presented for single machine scheduling problem. For the two-machine flowshop scheduling problem to minimize the makespan, they prove that the optimal solution can also be obtained by Johnson’ rule. Eren and Guner [16] consider the total tardiness problem in a single machine that job processing times are dependent of their position in the scheduling sequence.

Fig. 1. The process of ingot teeming and heating.
Different from the above models, we are in addition to taking into account scheduling with batching on the second machine in the two-machine flowshop environment.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed problems. In Sections 3 and 4, we study the first and second problem, respectively. Section 5 gives some concluding remarks.

2. Problem description

We now describe our problems formally. We consider two scheduling problems for a two-machine flowshop where a single machine is followed by a batching machine. The first problem is that there is a transporter to carry the jobs between machines. The second problem is that there are deteriorating jobs to be processed on the single machine. The two problems share the same machine environment which is described as follows.

The single machine processes one job at a time and the batching machine can process a batch of jobs simultaneously. There is a set of jobs \( J = \{1, 2, \ldots, N\} \) of \( N \) independent and non-preemptive jobs. Once processing of a batch is initiated, no job can be released from the batching machine until the entire batch is processed. The processing time of a batch on the batching machine is a constant \( C_j \), which denotes the basic processing time of job \( j \) on the single machine. After the job is processed on the single machine, it is transported to the batching machine by only one transporter. There is a transporter that can carry jobs back and forth between machines. The capacity of the transporter is equal to the capacity \( c \) of the batching machine, i.e., it can carry up to \( c \) jobs in one shipment. All the jobs delivered together in one shipment are defined as a delivery batch. We assume that a round trip time of the transporter is a constant \( T \), and two one-way times are identical, i.e., each one-way time is \( T/2 \), where \( T \) is a round trip time. The transportation time of the \( k \)th delivery batch is equal to \( T \).

For the first problem, let \( p_j \) denote the processing time of job \( j \) on the single machine. After the job is processed on the single machine, it is transported to the batching machine by only one transporter. There is a transporter that can carry jobs back and forth between machines. The capacity of the transporter is equal to the capacity \( c \) of the batching machine, i.e., it can carry up to \( c \) jobs in one shipment. All the jobs delivered together in one shipment are defined as a delivery batch. We assume that a round trip time of the transporter is a constant \( T \), and two one-way times are identical, i.e., each one-way time is \( T/2 \), where \( T \) is a round trip time. The transportation time of the \( k \)th delivery batch is equal to \( T \).

Concerning the second problem, the processing time of deteriorating job \( j \) on the single machine is given as a linear function dependent on its starting time \( t \): \( p_j(t) = a_j(a + bt) \), where \( a \) and \( b \) are positive constants and \( a_j \) denotes the basic processing time of job \( j \). Each job completed on the single machine moves to the batching machine immediately. The batching machine processes a number of jobs in a batch so that all jobs in each batch start together, and also finish their processing at the same time. The second problem can be denoted as \( F2|b_1 = 1, b_2 = c, t_k = T|C_{max} \), which denotes the minimization of the makespan in a two-machine flowshop where the first machine is a single machine and the second is a batching machine with capacity \( c \), and the transportation time of the \( k \)th delivery batch is equal to \( T \).

Concerning the second problem, the processing time of deteriorating job \( j \) on the single machine is given as a linear function dependent on its starting time \( t \): \( p_j(t) = a_j(a + bt) \), where \( a \) and \( b \) are positive constants and \( a_j \) denotes the basic processing time of job \( j \). Each job completed on the single machine moves to the batching machine immediately. The batching machine processes a number of jobs in a batch so that all jobs in each batch start together, and also finish their processing at the same time. The second problem can be denoted as \( F2|b_1 = 1, b_2 = c, t_k = T|C_{max} \), which denotes the minimization of the makespan in a two-machine flowshop where the first machine is a single machine and the second is a batching machine with capacity \( c \), and the transportation time of the \( k \)th delivery batch is equal to \( T \).

3. Scheduling problem with transportation

3.1. Model

In this subsection, the problem \( F2|b_1 = 1, b_2 = c, t_k = T|C_{max} \) can be formulated as a mixed integer programming model. Before the model is presented, the parameters and variables used in the model are firstly described below.

**Parameters**

- \( J \): The set of all jobs, \( J = \{1, 2, \ldots, N\} \), where \( N \) is the total number of jobs.
- \( c \): The capacity of the batching machine.
- \( T \): The round trip time of the transporter.
- \( P \): The batch processing time.
The number of batches, \( B_k \)

Decision variables

- \( x_{ijk} \): 1, if job \( j \) is the \( i \)th job in the sequence on the single machine and belongs to the \( k \)th batch on the batching machine; 0, otherwise.
- \( C_{ij} \): The completion time of job \( j \) on the single machine.
- \( R_k \): The maximum completion time of jobs belonged to batch \( B_k \) on the single machine, i.e., \( R_k = \max_{j \in B_k} \{ C_{ij} \} \).
- \( C_{2k} \): The completion time of the \( k \)th batch on the batching machine.
- \( I_k \): The idle time of the batching machine immediately before the \( k \)th batch starts.

Mixed integer programming model:

Objective function:

\[
\text{Min } C_{\text{max}} = \sum_{k=1}^{B} I_k + BP
\]  

Subject to

1. \( \sum_{j=1}^{N} \sum_{k=1}^{B} x_{ijk} = 1, \ i = 1, 2, \ldots, N, \)  
2. \( \sum_{i=1}^{N} \sum_{k=1}^{B} x_{ijk} = 1, \ j = 1, 2, \ldots, N, \)  
3. \( \sum_{j=1}^{N} \sum_{i=1}^{N} x_{ijk} \leq c, \ k = 1, 2, \ldots, B, \)  
4. \( R_1 + T/2 + P = C_{21}, \)  
5. \( R_k + T/2 \leq C_{2k} - P, \ k = 2, 3, \ldots, B, \)  
6. \( C_{2,k-1} \leq C_{2k} - P, \ k = 2, 3, \ldots, B, \)  
7. \( I_1 = R_1 + T/2, \)  
8. \( I_k \geq R_k + T/2 - C_{2,k-1}, \ k = 2, 3, \ldots, B, \)  
9. \( C_{2k} \leq C_{\text{max}}, \ k = 1, 2, \ldots, B, \)  
10. \( x_{ijk} \in \{0, 1\}, \ i, j = 1, 2, \ldots, N; \ k = 1, 2, \ldots, B, \)  
11. \( C_{ij}, C_{2k}, I_k \geq 0, \ i, j = 1, 2, \ldots, N; \ k = 1, 2, \ldots, B, \)

The objective function (1) minimizes the makespan, i.e., the maximum completion time of all jobs, which is equal to the sum of the total batch processing time and the total idle time on the batching machine. Constraint (2) specifies that only one job has to be assigned to position \( i \) in the sequence for any \( i \). Constraint (3) ensures that each job must be scheduled exactly once. Constraint (4) guarantees that the number of jobs scheduled in one batch cannot exceed the capacity of the batching machine. Constraint (5) indicates the completion time of the first batch on the batching machine. Constraints (6) and (7) define the property of the completion time of the batch on the batching machine. They indicate that the batching machine may start to process one batch after the jobs of this batch have arrived at the batching machine and the previous batch has completed. Constraints (8) and (9) define the idle time on the batching machine. Constraint (10) defines the properties of decision variables \( C_{\text{max}} \) and \( C_{2k} \). Constraints (11) and (12) define the range of the variables.

3.2. Strong NP-hardness

We now prove that the corresponding problem with the makespan as the scheduling objective is strongly NP-hard by a reduction from 3-PARTITION problem, which is known to be NP-hard in the strong sense.
In order to prove the NP-hardness of our problem, we need to construct an instance for our problem. If this constructed instance can be transformed polynomially to a given instance of the 3-PARTITION problem, then our problem is strongly NP-hard. In the proof of NP-hardness, as long as we are capable of constructing an instance of the problem and can prove that 3-PARTITION problem has a solution if and only if there is a feasible solution to the constructed instance, we can conclude that the problem is strongly NP-hard. As can be seen, in the constructed instance, the capacity $c$ of the batching machine may be arbitrarily chosen if only it can satisfy the above reduction condition. In our NP-hardness proof, we construct an instance with capacity $c = 3$ which can be reduced to 3-PARTITION problem.

3-PARTITION problem: Given $3h$ items, $H = \{1, 2, \ldots, 3h\}$, each item $j \in H$ has a positive integer size $a_j$ satisfying $a/4 < a_j < a/2$, and $\sum_{j=1}^{3h} a_j = ha$, for some integer $a$, do there exist $h$ disjoint subsets $H_1, H_2, \ldots, H_h$ of $H$ such that $|H_i| = 3$ and $\sum_{j \in H_i} a_j = a, i = 1, 2, \ldots, h$?

The following theorem states the computational complexity of the problem.

**Theorem 1.** Problem $F2|b_1 = 1, b_2 = c, t_k = T|C_{\max}$ is NP-hard in the strong sense.

**Proof.** The proof is based on the following transformation by reduction from 3-PARTITION problem. Given an arbitrary instance of 3-PARTITION problem, we construct the following instance of our scheduling problem:

- Number of jobs: $N = 3h + 3$.
- Capacity of the batching machine: $c = 3$.
- Transportation time: $T = P = a$.
- Processing time of the job on the single machine: $p_j = a_j, j = 1, 2, \ldots, 3h$.
  - $p_j = 0; j = 3h + 1, 3h + 2, 3h + 3$.
- Threshold value: $y = (h + 3/2)a$.

We will show that there exists a solution to 3-PARTITION problem if and only if there is a feasible solution to the constructed scheduling problem with a makespan of no greater than $y$.

- If there is a solution to 3-PARTITION problem, we show that there is a schedule to our problem with a makespan of no more than $y$. Given a solution to 3-PARTITION problem, $H_1, H_2, \ldots, H_h$, we construct a schedule for our problem as shown in Fig. 2.

  In this schedule, the first delivery trip carries three jobs $\{J_{3h+1}, J_{3h+2}, J_{3h+3}\}$ and departs at time 0, and the transporter delivers the jobs in $\{J_j| j \in H_i\}$ as a batch at a time, $i = 1, 2, \ldots, h$. It is easy to see that the above schedule is feasible and the makespan is $(h + 3/2)a = y$.

  Conversely, suppose that there exists a schedule for the constructed instance of our problem with a makespan of no greater than $y$. Since the makespan is always greater than or equal to $(h + 3/2)a$ for the $N = 3h + 3$ jobs, we obtain that the makespan is $(h + 3/2)a$. Since the total processing time of jobs on the batching
machine equals to \((h+1)a\), the earliest possible starting time of processing jobs on the batching machine is \(T/2 = a/2 = y - (h+1)a\). Therefore, we can see that for \(C_{\text{max}} = (h+3/2)a\), (1) jobs \(\{J_{3h+1}, J_{3h+2}, J_{3h+3}\}\) are the first jobs scheduled; (2) the single machine processes job in the interval \([0, ha]\) without idle time; (3) the transporter carries three jobs at a time in the interval \([0, (h+1/2)a]\) without idle time.

The proof can be done by contradiction. Suppose that there does not exist any feasible partition to the 3-PARTITION problem. Accordingly, let \(H_k\) be the first batch such that \(\sum_{j \in H_k} p_j \neq a\).

**Case 1.** \(\sum_{j \in H_k} p_j < a\). It is easy to check from Fig. 2 that the batch \(H_k\) will create idle time on the single machine, and this contradicts (2).

**Case 2.** \(\sum_{j \in H_k} p_j > a\). It is easy to check from Fig. 2 that the batch \(H_k\) will create idle time on the transporter, and this contradicts (3).

This implies the existence of a solution to 3-PARTITION problem.

Combining the “if” part and the “only if” part, we have proved the theorem. \(\square\)

When the round trip time of the transporter is variable, we have the following corollary from Theorem 1.

**Corollary 1.** Problem \(F2|b_1 = 1, b_2 = c, t_k|C_{\text{max}}\) is strongly NP-hard.

### 3.3. Heuristic for the problem \(F2|b_1 = 1, b_2 = c, t_k|C_{\text{max}}\)

Although the mixed integer programming model provides the optimal solution, variables and constraints increase drastically when the number of jobs increases. Theorem 1 indicates that the existence of a polynomial time algorithm to solve our scheduling problem is unlikely. Because of the inherent intractability of the scheduling problem, developing fast heuristic algorithm for yielding near-optimal solutions is justifiable. In this subsection, we present a heuristic algorithm and establish an upper bound on the worst case performance ratio of the heuristic algorithm. In the heuristic algorithm, we create \(L = \lceil N/c \rceil\) batches of jobs \(\{B_1, B_2, \ldots, B_L\}\) basing on the capacity \(c\) of the batching machine.

**Heuristic \(H\):**

**Step 1:** Divide \(N\) jobs into \(L\) batches. Arrange the jobs in non-increasing order of the processing times \(p_j\) so that \(p_1 \geq p_2 \geq \cdots \geq p_N\). Create empty batch \(B_h\) and set \(P_l = 0\), where \(P_l\) denotes the total processing time of the jobs assigned to the batch \(B_l\), \(l = 1, 2, \ldots, L\). Assign job \(J_l\) to batch \(B_k\), where \(k = \arg \min_{l=1,2,\ldots,L}(\{P_l | B_l\ \text{does not contain } c\ \text{jobs}\})\) with ties broken arbitrarily, and set \(P_k = P_k + p_j, j = 1, 2, \ldots, N\).

**Step 2:** Sequence the \(L\) batches and find the batch \(B_0\). Arrange the batches in non-decreasing order of the total processing time \(P_h\) and re-index the batches so that \(P_1 \leq P_2 \leq \cdots \leq P_L\). Let \(\theta = \max\{2, \min\{l | P_l \geq T\}\}\), and set \(P_{L+1} = +\infty\). Arrange the batches of jobs in the order of \(B_1, B_2, \ldots, B_0, B_\theta, \ldots, B_L\), and sequence the jobs within each batch in the longest processing time first order.

**Step 3:** Schedule the batches of jobs on the machines and the transporter. The single machine processes the jobs according to the order obtained in Step 2. The second batch of jobs is immediately processed on the single machine when the first batch of jobs has processed completely. From the second batch, the next batch of jobs is started processing on the single machine only when the transporter comes back the single machine. The transporter carries one batch at a time according to the order obtained in Step 2. And the batching machine also processes the batch of jobs according to the order obtained in Step 2.

**Remarks.** 1. Since there is no more than \(O(N^2)\) time for each of these steps in heuristic \(H\), the computational complexity of heuristic \(H\) is \(O(N^2)\).

2. The role of \(\theta\) in heuristic \(H\) is convenient for analyzing the worst case performance of heuristic \(H\).

Next, we are ready to evaluate the worst case performance of heuristic \(H\). Note that \(T \geq P, L = \lceil N/c \rceil, P_1 \leq P_2 \leq \cdots \leq P_{L-1} \leq T \leq P_0 \leq \cdots \leq P_L \leq P_{L+1}, 2 \leq \theta \leq L + 1, \) and \(P_{L+1} = +\infty\).

**Theorem 2.** The worst case performance ratio of heuristic \(H\) is less than or equal to 2.

**Proof.** Let \(C_{\text{max}}^{(H)}\) denote the makespan of the schedule generated by heuristic \(H\), it is easy to see that \(C_{\text{max}}^{(H)} = P_1 + (\theta - 2)T + \sum_{l=0}^{L} P_l + T/2 + P\).
Let $C_{\text{max}}(\sigma^*)$ denote the minimum makespan for problem $F2|b_1 = 1$, $b_2 = c$, $t_k = T|C_{\text{max}}$, we have $C_{\text{max}}(\sigma^*) \geq \max\{\sum_{i=1}^LP_i + T/2 + P, P_i + (L-1)T\}$.

Hence,

$$
\frac{C_{\text{max}}(\sigma^H)}{C_{\text{max}}(\sigma^*)} \leq \frac{P_1 + (0 - 2)T + \sum_{i=1}^LP_i + T/2 + P}{\max\{\sum_{i=1}^LP_i + T/2 + P, P_i + (L-1)T\}} \\
\leq \frac{\sum_{i=1}^LP_i + T/2 + P}{\sum_{i=1}^LP_i + T/2 + P} + \frac{(0 - 2)T + P_1}{(L-1)T + P_1} \leq 2. \quad (13)
$$

whether this error bound is tight or can be improved remains an interesting open question.

### 3.4. Computational experiments

In this subsection, we carry out computational experiments to verify the effectiveness of heuristic $H$. The heuristic algorithm is coded in Visual C++ language and implemented on the computer with 512MB RAM and 256KB L2 cache. The test problems are generated randomly by considering the following parameters:

- **Numbers of jobs ($N$):** 30, 80, 100, 160.
- **Capacity of the batching machine ($c$):** 5, 10, 15.
- **Processing times of jobs on the single machine ($p_j$):** generated from the discrete uniform distribution with range $[1, 30]$, $[1, 50]$, $[1, 100]$, $[1, 150]$.

We generate randomly the values of $T$ and $P$ basing on the different range of the discrete uniform distribution of processing times of jobs on the single machine, where the values of $T$ and $P$ satisfy $P_1 \leq T \leq P_L$ and $T \geq P$.

In order to evaluate the performance of heuristic $H$, we propose two lower bounds of the makespan. The first lower bound can be derived as the sum of the time required to complete all the jobs on the single machine, the one-way transportation time of the last batch, and the processing time of the last batch on the batching machine. That is, $l_{b_1} = \sum_{i=1}^LP_i + T/2 + P$, where we assume that there is no additional idle time on the single machine.

The second lower bound can be derived as the sum of the completion time of the first batch on the single machine, the total transportation time of $L$ batches, and the processing time of the last batch on the batching machine. That is, $l_{b_2} = P_1 + (L - 1)T + T/2 + P$, where we assume that there is no additional idle time on the transporter.

Based on the two lower bounds derived above, the overall lower bound can be derived as $LB = \max\{l_{b_1}, l_{b_2}\}$. The error ratio is defined as $ER = 100 \times (\text{Heu} - \text{LB})/\text{LB}$, where Heu denotes the makespan of the schedule generated by heuristic $H$. The average error ratio (Avg.ER) and the maximum error ratio (Max.ER) measured over the derived lower bound of the makespan are used for the performance test.

For each parameter combination, 50 random problems are generated for the performance test of the heuristic algorithm. The results of the evaluation are reported in Tables 1–3.

### Table 1

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 30$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg.ER</td>
<td>0.093</td>
<td>0.101</td>
<td>0.102</td>
<td>0.091</td>
</tr>
<tr>
<td>Max.ER</td>
<td>0.686</td>
<td>0.562</td>
<td>0.407</td>
<td>0.434</td>
</tr>
<tr>
<td>$n = 80$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg.ER</td>
<td>0.071</td>
<td>0.111</td>
<td>0.147</td>
<td>0.121</td>
</tr>
<tr>
<td>Max.ER</td>
<td>0.504</td>
<td>0.844</td>
<td>0.618</td>
<td>0.605</td>
</tr>
<tr>
<td>$n = 100$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg.ER</td>
<td>0.103</td>
<td>0.121</td>
<td>0.133</td>
<td>0.131</td>
</tr>
<tr>
<td>Max.ER</td>
<td>0.805</td>
<td>0.738</td>
<td>0.471</td>
<td>0.574</td>
</tr>
<tr>
<td>$n = 160$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg.ER</td>
<td>0.136</td>
<td>0.099</td>
<td>0.165</td>
<td>0.179</td>
</tr>
<tr>
<td>Max.ER</td>
<td>0.727</td>
<td>0.596</td>
<td>0.755</td>
<td>0.800</td>
</tr>
</tbody>
</table>
The computational results reveal that the performance of heuristic $H$ is effective in obtaining near-optimal solutions. They also indicate that the lower bound is close to the optimal solutions. From the computational results of average error ratios, we observe that the error ratio decreases as the given capacity of the batching machine increases. We also observe that the error ratio appears in an increasing trend as the variation range of the job processing times on the single machine increases. It may be interpreted that the larger processing time variances of jobs may incur the worse solutions.

4. Scheduling problem with deteriorating jobs

In this section, we consider the second problem that there are deteriorating jobs to be processed on the single machine. We develop the polynomial time algorithms for three different objective functions: the makespan, the total completion time and the maximum lateness, respectively. Given the considered flowshop environment and the set of jobs to be processed, the scheduling problem is to determine the sequence of all the jobs to be processed on the single machine, the composition of each batch, and the sequence of all the batches to be processed on the batching machine.

4.1. Minimizing makespan

In this subsection, we present a polynomial time algorithm for the problem $F2|b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt)|C_{\text{max}}$. Our first result establishes that we can focus on processing jobs in the permutation sequence. That is, the sequence of jobs processed on the single machine is also kept on for the batching machine.

**Lemma 1.** For the given scheduling problem, there exists an optimal schedule satisfying the permutation sequence.

**Proof.** It can be proved by a pairwise job interchange argument. Consider a non-permutation schedule $\sigma$, which implies that somewhere in $\sigma$, there must be a pair of jobs $i$ and $j$ in different orders on the single machine and the batching machine. Without loss of generality, if the job sequence on the batching machine is not the same as that on the single machine, then we can re-sequence jobs $i$ and $j$ to be in the same order as those on the
single machine without increasing the objective value. Similarly, by repeating the argument, a permutation schedule can be generated which is not worse than the schedule \( \sigma \).

Ikura and Gimple [19] have given the result that the FOE\((N, c)\) rule minimizes the makespan for a single batching machine problem, where the FOE\((N, c)\) rule is defined as the batching rule in which the first batch contains \( \lceil \frac{N}{c} \rceil - 1 \) jobs but the next \( \lceil \frac{N}{c} \rceil - 1 \) batches are all full of jobs up to the machine capacity (i.e., containing \( c \) jobs). Accordingly, it is stated in Lemma 2 that the FOE\((N, c)\) rule is the optimal batching on the batching machine.

Lemma 2. The FOE\((N, c)\) rule is optimal for the batching machine.

The remaining issue is how optimally to determine the jobs on the single machine. We can determine that the makespan is minimized by sequencing the jobs in the rule of shortest basic processing time first (SBPT). The SBPT rule is defined as sequencing the jobs in non-decreasing order of their basic processing times. As a consequence, this is stated in Lemma 3.

Lemma 3. The SBPT rule is optimal for the single machine.

Proof. It can be easy proved by a pairwise job interchange argument. We omit the details. \( \square \)

Consequently, we have the following theorem.

Theorem 3. Algorithm SBPT–FOE\((N, c)\) finds an optimal schedule for the problem \( F2|b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt)C_{max} \) in \( O(N\log N) \) time.

Proof. The proof of optimality is straightforward from the results of Lemmas 1–3. We now turn to time complexity. The computational complexity of the SBPT rule and the FOE\((N, c)\) rule is \( O(N\log N) \) and \( O(N) \), respectively. Accordingly, the computational complexity of the SBPT–FOE\((N, c)\) algorithm is \( O(N\log N) \). \( \square \)

4.2. Minimizing total completion time

In this subsection and the next subsection, we establish the processing order of the jobs in an optimal schedule on the single machine, and then use dynamic programming to make batching decision.

It is easy to use an interchange argument to show that Lemmas 1 and 3 still hold for the problem with the objective of the total completion time. Since the total completion time is non-decreasing in the completion times, there is no convenience in keeping the single machine idle, and each job on the single machine is started as soon as the previous job in the sequence is completed. We describe Lemma 4 as follows.

Lemma 4. It is optimal to process the jobs according to the SBPT rule on the single machine.

The problem \( F2 | b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt) \) can be solved by the proposed SBPT-DP algorithm, where the DP algorithm is developed to deal with a given set of deteriorating jobs by referencing Ahmadi et al. [3]. Before proposing the SBPT-DP algorithm, we first present the following lemma.

Lemma 5. It is optimal to batch and schedule the jobs according to the SBPT rule and it is sufficient to consider initiating a batch when a new job becomes available.

Proof. By Lemma 4, we know that the SBPT rule is optimal, and permutation schedule dominates other schedule from Lemma 1. \( \square \)

Now, we propose the SBPT-DP algorithm. The jobs are scheduled on the single machine by the SBPT order. Let \( r_n \) denote the time of job \( n \) that is available for processing on the batching machine, where \( r_n = \sum_{j=1}^{n} a_j(a + bt) \) for \( n = 1, 2, \ldots, N \). Let \( w(v) \) be the number of available jobs that have been already processed on the single machine and that are waiting to be processed on the batching machine at time \( v \). Clearly, \( 0 \leq w(v) \leq N \). Suppose that we schedule \( h \) continuous batches which start at time \( r_n \), then the starting time \( S_k \)
and the size $N_k$ of the $k$th batch in the sequence of $h$ continuous batches are $S_k = r_n + (k - 1)P$ and $N_k = \min\{w(S_k), c\}$ for $k = 1, 2, \ldots, h$, respectively. Note that $w(S_k) = \max\{w(S_{k-1}) - c, 0\} + \{\{j : S_{k-1} < r_j \leq S_k\}\}$, where $|X|$ denotes the number of elements in set $X$.

Define $f(r_n, w(r_n))$ as the minimum total completion times for the state $(r_n, w(r_n))$, then the recurrence relation is

$$f(r_n, w(r_n)) = \min_{1 \leq h \leq N} \left\{ f(r_{n+1}, w(r_n) + 1), \min_{1 \leq k \leq h} \left\{ N_k(S_k + P) + f(r_{n(h)}, w(r_{n(h)})) \right\} \right\},$$

where $n(h)$ returns the index of the first job to arrive after the $h$ continuous batches complete. The boundary condition is $f(r_{n+1}, x) = \left\{ \begin{array}{ll} 0, & x \leq 1 \\ +\infty, & x > 1 \end{array} \right.$, and the optimum is $f(r_1, 1)$.

**Theorem 4.** Algorithm SBPT-DP finds an optimal schedule for the problem $F2 \mid b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt) \mid \sum C_j$ in $O(N^3)$ time.

**Proof.** The proof of optimality follows from Lemmas 1, 4 and 5. The computational complexity of the SBPT rule is $O(N \log N)$, which does not affect the overall time complexity of our algorithm. The total number of states is calculated in $O(N^2)$ time because the number of initiating a batch is at most $N$ and $w(r_n) \leq N$, and the computation effort for each state requires $O(N)$. Therefore, the overall time complexity of the SBPT-DP algorithm is $O(N^3)$. □

Furthermore, to illustrate the application of Algorithm SBPT-DP, we consider the following numerical example.

**Example 1.** Consider the instance with $J = \{J_1, J_2, J_3, J_4\}$, such that $c = 2, P = 5, p_j(t) = a_j(1 + t), a_1 = 1, a_2 = 1/2, a_3 = 2, a_4 = 4/3$.

According to the SBPT-DP algorithm, the jobs are sequenced in the non-decreasing order of $a_j$, then we obtain the job sequence $\sigma = (J_2, J_1, J_4, J_3)$. We re-index the job sequence $\sigma$ such that $\sigma' = (J'_1, J'_2, J'_3, J'_4) = \sigma = (J_2, J_1, J_4, J_3)$. Hence, on the single machine we have $p'_1 = 1/2, C'_1 = 1/2 = r'_1; p'_2 = 3/2, C'_2 = 2 = r'_2; p'_3 = 4, C'_3 = 6 = r'_3; and p'_4 = 14, C'_4 = 20 = r'_4$. By using algorithm SBPT-DP to solve the example, we have the following result:

$$f(r'_4, 1) = \min \{f(r'_4, 2), (r'_4 + P) + f(r'_4, 1), w(r'_4, 1))\} = \min \{f(r'_4, 2) = +\infty, (20 + 5) + f(r'_4, 1) = 25\} = 25.$$

Similarly, $f(r'_4, 2) = (2r'_4 + P) = 50, f(r'_4, 3) = (2r'_4 + P) + (r'_4 + 2P) = 80, f(r'_4, 4) = 2(r'_4 + P) + 2(r'_4 + 2P) = 110, f(r'_3, 1) = \min \{f(r'_4, 2), (r'_3 + P) + f(r'_3, 1, w(r'_4, 1))\}

$$= \min \{f(r'_4, 2) = 50, (6 + 5) + f(r'_4, 1) = 36\} = 36,$$

$$f(r'_3, 2) = \min \{f(r'_4, 3), \min \{2(r'_3 + P) + f(r'_3, 1, w(r'_4, 1))\}, (r'_3 + P) + f(r'_3, 1, w(r'_4, 1))\} = \min \{f(r'_4, 3) = 80, min \{(6 + 5) + f(r'_4, 1) = 47\}\} = 47.$$
Hence, we obtain an optimal schedule $\sigma^* = ([J_2], \{J_1\}, \{J_4\}, \{J_3\})$, which has the total completion time of 51.

### 4.3. Minimizing maximum lateness

In this subsection, we consider the objective to minimize the maximum lateness. It is assumed that the processing times and due dates of the jobs are agreeable, i.e., if $p_i(t) \leq p_j(t)$, then $d_i \leq d_j$, for $1 \leq i, j \leq N$. We assume that the due date $d_i$ is greater than or equal to the system lead time during which all the jobs are completed on the single machine, i.e., $d_i \geq \sum_{j \in S} p_j(t), n = 1, 2, \ldots, N$.

Since the processing times and due dates of the jobs are agreeable, Lemma 1 still hold for the objective to minimize the maximum lateness. We can determine that the maximum lateness is minimized by sequencing the jobs in the rule of earliest due date first (EDD) for the single machine. Accordingly, this is stated in Lemma 6.

**Lemma 6.** It is optimal to process the jobs according to the EDD rule on the single machine.

**Proof.** It can be easy proved by a pairwise job interchange argument. We omit the details. $\square$
We propose an EDD-DP algorithm for the problem $F2|b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt)|L_{\text{max}}$, where the DP algorithm is based on Webster and Baker [1]. We give the following lemma that is similar to Lemma 5.

**Lemma 7.** It is optimal to batch and schedule the jobs according to the EDD rule and it is sufficient to consider initiating a batch when a new job becomes available.

We now propose the EDD-DP algorithm. Let $\sigma = (1, 2, \ldots, N)$ denote the EDD order of jobs to be processed on the single machine. Let $r_n$ denote the time at which job $n$ is available for processing on the batching machine, where $r_n = \sum_{j=1}^{n} a_j(a + bt)$ for $n = 1, 2, \ldots, N$. Let $L(n, h, w(r_n))$ be the minimum maximum lateness for the jobs contained in $h$ continuous batches beginning with the state $(r_n, w(r_n))$, i.e.,

$L(n, h, w(r_n)) = \max_{k < h} \{r_n + kP - d'\}$,

where $k' = \sum_{j=1}^{n-1} N_j + | \{ j : r_j < r_n \} | - w(r_n) + 1$, represents the index of the jobs with the EDD order in the $k$th batch.

Define $g(r_n, w(r_n))$ as the minimum maximum lateness for the state $(r_n, w(r_n))$, then the recurrence relation is

$$g(r_n, w(r_n)) = \min \left\{ g(r_{n+1}, w(r_n) + 1), \min_{1 \leq h \leq N} \left\{ \max \left\{ L(n, h, w(r_n)), g(r_n, w(r_n)) \right\} \right\} \right\},$$

where $n(h)$ returns the index of the first job to arrive after the $h$ continuous batches complete. The boundary condition is $g(r_{N+1}, x) = \left\{ \begin{array}{ll} -\infty, & x \leq 1 \\
+\infty, & x > 1 \end{array} \right.$, and the optimum is $g(r_1, 1)$.

Similarly, we have the following theorem.

**Theorem 5.** Algorithm EDD-DP finds an optimal schedule for the problem $F2|b_1 = 1, b_2 = c, p_j(t) = a_j(a + bt)|L_{\text{max}}$ in $O(N^3)$ time.

5. Conclusions

In this paper, we investigate two scheduling problems for a two-machine flowshop where a single machine is followed by a batching machine. For one scheduling problem where there is a transporter to carry the jobs between machines, we formulate the problem with minimizing the makespan as a mixed integer programming model and prove that it is strongly NP-hard. We propose a heuristic algorithm and analyze its worst case performance. The computational experiments are carried out and the numerical results show that the performance of the heuristic algorithm is effective in solving this scheduling problem. For the other scheduling problem where there are deteriorating jobs to be processed on the single machine, we show that optimal schedules can be found for the objective to minimize the makespan, the total completion time and the maximum lateness, respectively.

Many interesting topics remain for future exploration. Firstly, we may assume that the processing time of a batch is the maximum processing time of the jobs grouped together in the batch. Secondly, the objective to minimize the number of tardy jobs can be researched in the second scheduling problem. Finally, we can study a two-machine flowshop problem where a batching machine is followed by a single machine.

Acknowledgements

The authors would like to thank two anonymous referees for their helpful comments and suggestions. This research is partly supported by National Natural Science Foundation for Distinguished Young Scholars of China (Grant No.70425003), National 863 High-Tech Research and Development Program of China through approved No. 2006AA04Z174 and National Natural Science Foundation of China (Grant No. 60674084).

References