A hybrid two-stage transportation and batch scheduling problem

Lixin Tang *, Hua Gong

The Logistics Institute, Northeastern University, Shenyang 110004, China

Received 1 August 2006; received in revised form 1 April 2007; accepted 17 September 2007
Available online 6 October 2007

Abstract

We study the coordinated scheduling problem of hybrid batch production on a single batching machine and two-stage transportation connecting the production, where there is a crane available in the first-stage transportation that transports jobs from the warehouse to the machine and there is a vehicle available in the second-stage transportation to deliver jobs from the machine to the customer. As the job to be carried out is big and heavy in the steel industry, it is reasonable assumed that both the crane and the vehicle have unit capacity. The batching machine processes a batch of jobs simultaneously. Each batch occur a setup cost. The objective is to minimize the sum of the makespan and the total setup cost. We prove that this problem is strongly NP-hard. A polynomial time algorithm is proposed for a case where the job transportation times are identical on the crane or the vehicle. An efficient heuristic algorithm for the general problem is constructed and its tight worst-case bound is analyzed. In order to further verify the performance of the proposed heuristics, we develop a lower bound on the optimal objective function. Computational experiments show that the heuristic algorithm performs well on randomly generated problem instances.

Keywords: Transportation; Batch scheduling; Coordination; Setup cost

1. Introduction

There are many production systems with transportation operations where semi-finished jobs are transferred from warehouse to a manufacturing facility by transporters for further processing and finished jobs are delivered to customers by transporters. The coordination of production and logistics operations can help to improve overall system performance and reduce the total operation cost.

In this paper, we study a coordinated scheduling problem of production and transportation where a batching machine and two-stage transportation connecting the production are considered (see the schematic in Fig. 1). There is a crane available in the first-stage transportation that can transport jobs from the warehouse to the batching machine, and there is a vehicle available in the second-stage transportation to deliver jobs from the machine to the customer.
the machine to the customer. The batching machine can process jobs up to the maximum number machine can hold simultaneously. The crane and vehicle can transport a job at a time. Batching is the decision about whether or not to group some jobs into a batch to be processed. The problem is to integrate batching scheduling and transportation scheduling.

As a practical example of the proposed problem, we can consider a scheduling issue for steel ingot production in iron and steel plant. Molten steel from a steelmaking furnace is poured into molds to produce steel ingots and the molds are stripped off from the ingots. Then ingots are conveyed by a crane to a soaking pit where the ingots are heated in batch to a sufficiently high temperature. The reheated ingots are transported to a blooming mill by a vehicle where ingots are rolled to a usable form of steel. As there is always some fuel consumption associated with each batch during heating processing, the objective of the scheduling concerning ingot transportation and heating is to not only maximize the machine utilization rate but also minimize the total setup cost with fuel consumption level. This is the motivation for the considered problem.

The related literatures can be divided into two major issues. The first issue deals with machine scheduling problems with transportation but without batching machine. Lee and Chen [1] comprehensively research two types of transportation. One type involves transporting a semi-finished job from one machine to another machine for further processing inside a manufacturing facility. Another type appears in the environment of delivering a finished job to the customer or warehouse. Both transportation capacity and transportation times are explicitly taken into account in their models. Chang and Lee [2] have extended Lee and Chen’s work to the situation when each job occupies a different amount of space in the vehicle. Two-type transportation models involve only regular performance measure and do not deal with the delivery cost. Pundoor and Chen [3] consider a make-to-order production–distribution system with one supplier and one or more customers that optimize an objective function involving the maximum delivery tardiness and the total distribution cost. Chen and Vairaktarakis [4] integrate production scheduling for job processing in the single machine and parallel machine with distribution scheduling and routing for delivery of completed jobs to the customers. There are enough vehicles available and each vehicle has limited capacity in the above two papers. Cheng and Kahlbacher [5], Cheng and Gordon [6], Wang and Cheng [7] address the batch delivery problems from a distribution cost point of view. Hall and Potts [8] consider various problems with an objective function combining schedule cost and delivery cost in the context of a two-stage supply chain, and without a transporter capacity constraint. Li and Ou [9] analyze a single machine scheduling model that incorporates the scheduling of jobs and the pickup and delivery arrangement of materials and finished jobs, and their model involves a capacitated pickup and delivery vehicle that travels between the machine and the storage area.

The other issue concerns single batching machine. Potts and Kovalyov [10] provide an extensive review of research in the area of scheduling with batching. Ahmadi et al. [11] study a class of scheduling problems defined in a two- or three-machine flowshop with at least one batching machine incorporated. Sung and Kim [12] have extended the two-machine flowshop problem of Ahmadi et al.’s models to the situation in which a finite number of jobs arrive dynamically at the first machine. These problems deal with the production part and do not consider the tradeoff between transportation and setup cost.

Our work differs from the above in that we study not only the transportation of semi-finished jobs and finished jobs, but also the scheduling of the batching machine. Motivated by the problem in practical production environment, we do not assume zero returning time. We note that a crane (or a vehicle) in two-stage trans-
portation may not be viewed as a "machine". It is because it is occupied but not carrying any job as the crane (the vehicle) is returning from the machine (customer) to the warehouse (machine). Hence, a crane (or a vehicle) is different from what a real machine is supposed to be in traditional scheduling problems. Thus, our problem is different from Ahmadi et al.'s three-machine flowshop with one batching machine incorporated. When we consider job production with two-stage transportation, we want to batch together a set of jobs for processing in order to reduce the total setup cost, but the decrease of total setup cost may result in greater makespan. On the other hand, a short makespan may result in the increase of cost if we create too many batches. Hence, our work studies not by solely minimizing makespan like that three-machine flowshop problem without transportation but by a coordination of makespan and the setup cost.

The rest of this paper is organized as follows. In the next section, we give problem description. In Section 3, a mixed integer programming model is presented. We also analyze the properties of the problem and prove the strong NP-hardness result. In Section 4, we generalize a polynomial time algorithm to the case with identical transportation times on the crane or the vehicle. In Section 5, we show a heuristic algorithm with tight worst-case analysis for the general problem. In Section 6, some computational experiments are conducted to test the performance of our heuristic. Finally, some conclusions are given in Section 7.

2. Problem description

We now describe our problem formally. There is a given set \( N \) of \( n \) jobs that are initially located at the warehouse. All the jobs need to be sent to the batching machine for further processing, and have to be delivered to the corresponding customers after the processing is completed. There is a crane available in the first-stage transportation that can transport jobs from the warehouse to the machine, and there is a vehicle available in the second-stage transportation to deliver jobs from the machine to the customer. For ease of exploration, we assume that loading and unloading times are included in transportation times of jobs, and then all transportation times are assumed to be job-dependent.

In the first-stage transportation, the crane is initially located at the warehouse and has unit capacity. The crane takes \( t_{j1} \) units of time for job \( j \) from the warehouse to the machine and \( t_1 \) units of time to travel back to the warehouse.

In the production part, the batching machine can process up to \( c \) jobs simultaneously, where \( (c > 1) \) is called capacity of the batching machine. Once the processing of a batch is initiated, it cannot be interrupted, nor can other jobs be introduced into the batch. Each batch to be processed occur a setup cost. We assume that the time needed to process a batch of jobs on the machine is denoted by a constant \( p \) regardless of the jobs grouped together in the batch.

In the second-stage transportation, we only assume that all customers are located in close proximity to each other. The vehicle is stationed at the machine at time 0 and delivery a job at a time. The transportation time for job \( j \) from the machine to the customer is \( t_{j2} \), and the empty moving time of the vehicle back to the machine is \( t_2 \). We use \( C_{\text{max}} \) denote makespan of the schedule, i.e., it is the arrival time of the last delivered job to the customer.

Makespan and setup cost are two major factors in this system. Makespan can measure the machine utilization rate and the total setup cost reflects fuel consumption level during the processing which is determined by the number of batches. Our objective is to find a coordinated schedule of production and transportation such that the sum of makespan and total setup cost, defined as \( Z = C_{\text{max}} + \alpha(b) \), is minimized. Here, \( b \) denotes the number of batches to be processed on the batching machine, and the total setup cost \( \alpha(b) \) is an increasing function of the number of batches. We may adjust cost function \( \alpha(b) \) to a uniform dimension with makespan.

3. Analysis of the model

In this section, in order to describe the problem clearly, the problem under consideration can be formulated as a mixed integer programming (MIP) model. We analyze the properties of the problem and show that the problem is strongly NP-hard by a reduction from 3-PARTITION, which is known to be NP-hard in the strong sense (see [13]). To define the model, we introduce the following notations:
the number of batches, \([n/c] \leq b \leq n\)

\(B_i\) the set of all jobs in the \(i\)th batch, \(i = 1, 2, \ldots, b\)

\(S_i\) the starting processing time of the batch \(B_i\) on the batching machine

\(C_i\) the completion time of the batch \(B_i\) on the vehicle, i.e., the arrival time of the last job of \(B_i\) to the customer

\(x_{ji}\) the decision variable whose value equals to 1 if job \(j\) is assigned to the \(i\)th batch to be processed, and otherwise, equals to zero, \(j = 1, 2, \ldots, n, i = 1, 2, \ldots, b\).

The MIP model is constructed as follows:

Objective function:

\[
\text{Min } Z = C_{\text{max}} + \alpha(b)
\]  \hspace{1cm} (1)

Subject to:

\[
\sum_{j=1}^{n} x_{ji} \leq c, \quad i = 1, 2, \ldots, b,
\]  \hspace{1cm} (2)

\[
\sum_{i=1}^{b} x_{ji} = 1, \quad j = 1, 2, \ldots, n,
\]  \hspace{1cm} (3)

\[
S_1 = \sum_{j=1}^{n} (t_{j1} + t_1)x_{j1} - t_1,
\]  \hspace{1cm} (4)

\[
C_1 = S_1 + p + \sum_{j=1}^{n} (t_{j2} + t_2)x_{j1} - t_2,
\]  \hspace{1cm} (5)

\[
S_i \geq \sum_{k=1}^{i} \sum_{j=1}^{n} (t_{j1} + t_1)x_{jk} - t_1, \quad i = 2, 3, \ldots, b,
\]  \hspace{1cm} (6)

\[
S_i \geq S_{i-1} + p, \quad i = 2, 3, \ldots, b,
\]  \hspace{1cm} (7)

\[
C_i \geq S_i + p + \sum_{j=1}^{n} (t_{j2} + t_2)x_{ji}, \quad i = 2, 3, \ldots, b,
\]  \hspace{1cm} (8)

\[
C_i \geq C_{i-1} + \sum_{j=1}^{n} (t_{j2} + t_2)x_{ji}, \quad i = 2, 3, \ldots, b,
\]  \hspace{1cm} (9)

\[
C_i + \alpha(i) \geq C_{i-1} + \alpha(i-1), \quad i = 1, 2, \ldots, b,
\]  \hspace{1cm} (10)

\[
C_{\text{max}} \geq C_i, \quad i = 1, 2, \ldots, b.
\]  \hspace{1cm} (11)

The objective function (1) represents the sum of the schedule length and the total cost. Constraints (2) ensure that the number of jobs scheduled in each batch cannot exceed the capacity of the batching machine. Constraints (3) guarantee that each job must be scheduled exactly once. Constraints (4) and (5) define the starting time of the first batch on the machine and the completion time of the first batch to the customer. Constraints (6) and (7) indicate that the batching machine may start to process one batch after the jobs of this batch have arrived at the machine and the previous batch has completed. Constrains (8) and (9) define the arrival time of each batch to the customer. They indicate the vehicle may start to transport the jobs of one batch after this batch is completed on the machine and the vehicle finishes transportation of the previous batch. Constraints (10) and (11) define the properties of decision variables \(C_{\text{max}}, C_i\) and \(\alpha(i)\).

We present some straightforward optimality properties to our problem.

**Lemma 1.** There exists an optimal schedule for the problem that satisfies the following conditions:

1. There is no idle time on the crane in the first-stage transportation.
2. Jobs processed in the same batch are transported consecutively without idle time on the vehicle, i.e., if there are jobs that still need to be delivered, then the vehicle will depart from the machine.
3. The sequence of jobs on the crane is the same as that on the vehicle.
Proof

(1) If jobs are transported on the crane with idle time, we can always move the subsequent jobs earlier without increasing the objective value.

(2) If jobs in the same batch are not transported consecutively on the vehicle, then we can always move the jobs of the same batch on the vehicle such that these jobs are transported consecutively and the completion time of the batch is not changed. The objective value does not increase as a result these moves. It is easy to see that the vehicle can consecutively delivery all jobs of the same batch without idle time.

(3) If the job sequence on the vehicle is not the same as that on the crane, then we can re-sequence jobs on the vehicle to be in the same order as those on the crane without increasing the objective value.

The following theorem states the computational complexity of the problem. Problem reduction is a principal concept in complexity theory. If one problem can be reduced to another problem, this means that for any instance of the later problem an equivalent instance of the prior problem can be constructed. If a problem which is known strongly NP-hard reduces to another problem, then the later problem is also proved strongly NP-hard. In order to prove the NP-hardness of our problem, we need verify whether 3-PARTITION problem reduces to our problem or not. If we can choose an equivalent instance for our problem which can be transformed in polynomial time to a given instance of the 3-PARTITION problem, then our problem is strongly NP-hard.

3-PARTITION problem. Given a set of 3h items $H = \{1, 2, \ldots, 3h\}$ and an integer $b$ such that each item $j \in H$ has an integer size $a_j$ satisfying $b/4 < a_j < b/2$ and $\sum_{j=1}^{3h} a_j = hb$. Do there exist $h$ disjoint subsets $H_1, H_2, \ldots, H_h$ of $H$ such that each subset contains exactly three items and its total size $\sum_{j \in H_i} a_j = b$ for $i = 1, 2, \ldots, h$?

**Theorem 1.** The problem is NP-hard in the strong sense.

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

- **Number of jobs:** $n = 3h + 3$, $N = H \cup \{3h + 1, 3h + 2, 3h + 3\}$;
- **Transportation times:**
  - $t_{j1} = t_{j2} = a_j$, $j = 1, 2, \ldots, 3h$;
  - $t_{3h+1,1} = 1$, $t_{3h+1,2} = a + 2$,
  - $t_{3h+2,1} = t_{3h+2,2} = t_{3h+3,1} = t_{3h+3,2} = 0$;
  - $t_1 = t_2 = 1$;
- **Processing times:** $p_j = a + 3$, $j = 1, 2, \ldots, 3h + 3$;
- **Machine capacity:** $c = 3$;
- **Setup cost:** $\alpha(b) = 0$;
- **Makespan threshold value:** $y = (h + 2)(a + 3)$.

Now we prove that the problem instance has a minimum objective value of $y$ if and only if 3-PARTITION problem has a solution. This will then imply that our problem is strongly NP-hard.

→ If there is a solution to 3-PARTITION problem instance, 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 instance, $H_1, H_2, \ldots, H_h$, we construct a schedule for our problem as shown in Fig. 2.

![Fig. 2. The schedule in the proof of Theorem 1.](image-url)
In this schedule, job \(3h + 1\) is first transported by the crane, and the crane starts transporting the jobs of \(H_i\), one by one at time \(1 + i(a + 3)\) from warehouse to the batching machine, for \(i = 1, 2, \ldots, h\). The machine processes three jobs as a batch at each time point \(1 + i(a + 3)\), for \(i = 0, 1, \ldots, h\). The vehicle delivers job \(3h + 1\) to the customer at time \(a + 4\), and begins delivering the jobs of \(H_i\) one by one at time \(1 + (i + 1)(a + 3)\) from machine to customer, for \(i = 1, 2, \ldots, h\). It is easy to see that the above schedule (see Fig. 2) is feasible and the makespan is \(y\).

Conversely, suppose that there exists a schedule for the constructed instance of our problem with a makespan of no more than \(y\). We can obtain that there is exactly \(h + 1\) batches on the batching machine. The smallest possible number of batches is \(h + 1\) on the machine due to the limited capacity of the machine. If there are more than \(h + 1\) batches in this schedule, without loss of generality, we assume that there are \(h + 2\) batches. Then the corresponding total processing time on the batching machine is \((h + 2)(a + 3) = y\). This is a contradiction because all jobs need to be considered transportation.

From the above discussion, we see that the schedule consists of exactly \(h + 1\) batches, say \(B_1, B_2, \ldots, B_{h+1}\), each of which contains exactly three jobs. Hence, job \(3h + 1\) is the first job scheduled. We know that the first batch contains three jobs \(3h + 1, 3h + 2, 3h + 3\) and is processed at time 1. Furthermore, the earliest possible starting time of delivering jobs on the vehicle is \(1 + p_{3h+1} = a + 4 = y - [(h + 1)(a + 3) - 1]\). The corresponding total transportation time of all jobs on the vehicle is \(ha + a + 3h + 2\). This implies that there is no idle time on the vehicle after it starts transporting the first job \(3h + 1\) at time \(a + 4\).

Denote the starting times of \(h + 1\) batches on the machine by \(S_1, S_2, \ldots, S_{h+1}\), respectively. Let the completion times of these batches on the machine be denoted by \(c_1, c_2, \ldots, c_{h+1}\), and the corresponding arrival times at the customer be denoted by \(C_1, C_2, \ldots, C_{h+1}\), respectively. Obviously, \(c_i = S_i + p\) for all \(i\). It is easy to see that \(S_1 = 1, c_1 = a + 4, \) and \(C_1 = 2a + 6\). The vehicle returns to the machine from the customer at time \(C_1 + 1 = 2a + 7\). Thus, the second batch \(B_2\) must be completed on the machine at time no later than \(2a + 7\). Since \(p = a + 3\), then \(S_2 > S_1 + a + 3\) and \(c_2 > S_2 + p = 2a + 7\). In other words, it indicates that \(c_2 = 2a + 7\) and \(S_2 = a + 4\).

Now we prove that \(\sum_{j \in B_i} t_{j1} = a\). If \(\sum_{j \in B_i} t_{j1} < a\), then \(\sum_{j \in B_i} t_{j2} < a\), and hence \(C_2 < 3a + 9\). On the other hand, it can be seen that \(c_3 > c_2 + p = 3a + 10\) and the arrival time of the vehicle at the machine is \(C_2 + 1 < 3a + 10\). As a result, there is idle time after batch \(B_1\) on the vehicle, which is a contradiction. This in turn implies that the batch \(B_i\) must start being processed at time \(1 + i(a + 3)\) and completed at time \(1 + (i + 1)(a + 3)\), for \(i = 3, 4, \ldots, h + 1\). We can obtain that \(\sum_{j \in B_i} t_{j1} = a\) by the same discussion. Then it is easy to see that the \(h\) batches \(B_2, B_3, \ldots, B_{h+1}\) form a solution to the 3-PARTITION problem instance.

Combining the “if” part and the “only if” part, we have proved the theorem.

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. Therefore, developing fast heuristic algorithm for yielding near-optimal solutions is of great interest.

4. Special case

In this section, we construct a dynamic programming algorithm with polynomial time for the case with identical transportation times on the crane or the vehicle.

For the case \(t_{j1} = t\), index the jobs such that \(t_{j2} > t_{j2} > \cdots > t_{n2}\). It suffices to consider one job sequence and apply it to the transportation from the warehouse to the machine, the processing on the machine, and the delivery from the machine to the customer. So the starting time of each batch on the machine needs to be decided, and this can be done by dynamic programming. These jobs are transported in the order \((1, 2, \ldots, n)\) by the crane without idle time. The arrival time of job \(j\) at the machine is \(r_j = j(t + t_1) - t_1\), for \(j = 1, 2, \ldots, n\). Once the machine finishes one batch, it either processes a new batch immediately or waits until the last job of new batch arrives at the machine. In the first situation, the starting time of a new batch is \(x + p\) where the starting time of current last batch on the machine is \(x\). In the second situation, the starting time is \(r_j\), for some job \(j\). We notice that \(x\) can be tracked in the first situation and the starting time can be expressed as \(r_j + ap\) for some \(q < n - j\) and some job \(j\). Hence, the possible starting times of the machine can be
earliest time is

Consider a 5-jobs problem such that the dynamic programming algorithm finds an optimal schedule for the problem with \( t_{j1} = t \) in \( \mathcal{O}(cn^3) \) time.

Proof. The proof of optimality follows from the above description and analysis. By definition, there are at most \( n^2 \) candidate starting time points for \( s(h) \). Given \( k \) and \( j \), there are at most \( \mathcal{O}(c) \) possible such \( s(h) \)'s because \( k - j + 1 \leq c \). Thus, there are a total of \( \mathcal{O}(cn^2) \) possible combinations of \( s(h) \) and \( s(l) \) in the algorithm. For each combination, there are a total of \( \mathcal{O}(c^2n) \) possibilities of \((k,j,i)\). Therefore, this algorithm has the complexity \( \mathcal{O}(cn^3) \).

For illustration, Example 1 is now solved.

Example 1. Consider a 5-jobs problem such that \( c = 2 \), \( p = 5 \), \( t_{j1} = t = 2 \), \( t_1 = t_2 = 1 \), \( \alpha(b) = 8b \), and the following transportation time on the vehicle:

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{j2} )</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

\( r_j, r_j + p, \ldots, r_j + qp \), for some \( q \leq n - j \), and \( j = 1, 2, \ldots, n \). Define \( s_m(h) \) as the starting time of the batch \( B_m \) corresponding to time point \( h \), for \( h = r_j, r_j + p, \ldots, r_j + qp, j = 1, 2, \ldots, b \), where \( \lfloor n/c \rfloor \leq b \leq n \) and \( s(0) = 0 \).

Let \( f(k,j,s_m(h)) \) denote the minimal makespan to schedule the first \( k \) jobs \( 1, 2, \ldots, k \), provided that the current last batch \( B_m \) contains jobs \( j, j + 1, \ldots, k \) which are processed from times \( s_m(h) \) where \( k - j + 1 \leq c \) and \( s_m(h) \geq r_k \). It is easy to see that \( \lfloor k/c \rfloor \leq m \leq k \) due to the limited capacity of the machine. Note that if the starting time of the batch \( B_m \) on the batching machine and the available time of vehicle can be derived, then the increase of the makespan due to jobs \( j, j + 1, \ldots, k \) is exactly total transportation time of these jobs. The earliest time is \( f(j - 1, i, s_{m-1}(l)) \) when the vehicle becomes available. Namely, \( f(j - 1, i, s_{m-1}(l)) \) for jobs \( i, i + 1, j - 1 \) that satisfies the following three properties.

(i) \( 0 < j - i \leq c \);
(ii) \( s_{m-1}(l) = r_j, r_j + p, \ldots, r_j + qp \), and \( s_m(h) - s_{m-1}(l) \geq p \);
(iii) \( \lfloor (j - 1)/c \rfloor \leq m - 1 \leq j - 1 \).

Otherwise, \( f(j - 1, i, s_{m-1}(l)) = \infty \).

At first, we denote initial conditions: \( f(0, 0, 0) = 0 \). Then, the induction formulas can be expressed as follows:

\[
f(k, j, s_m(h)) = \min \begin{cases} 
s_m(h) + p + \sum_{u=j}^{k} t_{u2} + t_2 - t_2, & \text{if } f(j - 1, i, s_{m-1}(l)) + t_2 \leq s_m(h) + p \text{ and } j > 1 \\
f(j - 1, i, s_{m-1}(l)) + \sum_{u=j}^{k} t_{u2} + t_2, & \text{if } f(j - 1, i, s_{m-1}(l)) + t_2 > s_m(h) + p \text{ and } j > 1 \\
s_1(r_k) + p + \sum_{u=1}^{k} t_{u2} + t_2 - t_2, & \text{if } j = 1 
\end{cases}
\]

where \( j, i, s_{m-1}(l) \) satisfy the conditions (i), (ii) and (iii) described above, and recursive equations satisfy the following conditions: \( 0 < k - j + 1 \leq c \), \( \lfloor k/c \rfloor \leq m \leq k \) and \( s_m(h) = r_j, r_j + p, \ldots, r_j + qp \), for \( k = 1, 2, \ldots, n \).

Define \( F(k) \) as the minimal sum of makespan and total setup cost to schedule the first \( k \) jobs \( 1, 2, \ldots, k \) where \( F(0) = 0 \). Then the function can be derived as

\[
F(k) = \min \{ f(k, j, s_m(h)) + \alpha(m) \} \text{ all possible states } (j, s_m(h))\}.
\]

Thus, the optimal solution is obtained after the induction process, and it is in form of

\[
F(n) = \min \{ f(n, j, s_b(h)) + \alpha(b) \}.
\]

By recording all the necessary information in the above process, an optimal schedule can be calculated. From the above description and analysis, the following theorem is true.

Theorem 2. The dynamic programming algorithm finds an optimal schedule for the problem with \( t_{j1} = t \) in \( \mathcal{O}(cn^3) \).

Proof. The proof of optimality follows from the above description and analysis. By definition, there are at most \( n^2 \) candidate starting time points for \( s(h) \). Given \( k \) and \( j \), there are at most \( \mathcal{O}(c) \) possible such \( s(h) \)'s because \( k - j + 1 \leq c \). Thus, there are a total of \( \mathcal{O}(cn^2) \) possible combinations of \( s(h) \) and \( s(l) \) in the algorithm. For each combination, there are a total of \( \mathcal{O}(c^2n) \) possibilities of \((k,j,i)\). Therefore, this algorithm has the complexity \( \mathcal{O}(cn^3) \).
If the jobs are indexed in the non-increasing order of $t_{j2}$, then we obtain the job sequence $\pi = \{J_4, J_3, J_5, J_1, J_2\}$. Then arrival time of job $j$ at the machine is $r_j = j(t + t_1) - t_1$, for $j = 1, 2, \ldots, n$. Now, by the following the procedure of the proposed dynamic programming algorithm, the optimal value of $f(k, j, s_m(h))$ is calculated:

$$f(1, 1, s_1(h)) = r_1 + p + t_{12} = 12;$$
$$F(1) = f(1, 1, s_1(h)) + \alpha(1) = 20.$$  
$$f(2, 1, s_1(h)) = r_2 + p + t_{42} + t_2 + t_{12} = 20;$$
$$f(2, 2, s_2(h)) = f(1, 1, s_1(h)) + t_2 + t_{32} = 17;$$
$$F(2) = \min\{f(2, 1, s_1(h)) + \alpha(1), f(2, 2, s_2(h)) + \alpha(2)\} = 28.$$  
$$f(3, 2, s_2(h)) = f(1, 1, s_1(h)) + t_{32} + 2t_2 + t_{12} = 21;$$
$$f(3, 3, s_2(h)) = f(2, 1, s_1(h)) + t_2 + t_{52} = 24;$$
$$f(3, 3, s_3(h)) = f(2, 2, s_2(h)) + t_2 + t_{32} = 21;$$
$$F(3) = \min\{f(3, 2, s_1(h)) + \alpha(2), f(3, 3, s_2(h)) + \alpha(2), f(3, 3, s_3(h)) + \alpha(3)\} = 37.$$  
$$f(4, 3, s_2(h)) = f(2, 1, s_1(h)) + t_{32} + 2t_2 + t_{12} = 27;$$
$$f(4, 3, s_3(h)) = f(2, 2, s_2(h)) + t_2 + t_{52} + t_{12} = 24;$$
$$f(4, 4, s_3(h)) = \min\{f(3, 2, s_2(h)) + t_2 + t_{12} = 24;$$
$$f(4, 4, s_4(h)) = s_3(h) + p + t_{12} = 24;$$
$$F(4) = \min\{f(4, j, s_m(h)) + \alpha(m)\} \text{ all possible states } (j, s_m(h)) = 43.$$  
$$f(5, 4, s_3(h)) = \min\{f(3, 2, s_2(h)) + 2t_2 + t_{12} + t_{32} = 26;$$
$$f(5, 4, s_4(h)) = s_4(h) + p + t_2 + t_{12} + t_{32} = 26;$$
$$f(5, 5, s_3(h)) = f(4, 3, s_2(h)) + t_2 + t_{32} = 29;$$
$$f(5, 5, s_4(h)) = \min\{f(4, 3, s_3(h)) + t_2 + t_{32} = 26;$$
$$f(5, 5, s_5(h)) = f(4, 4, s_3(h)) + t_2 + t_{32} = 26;$$
$$F(5) = \min\{f(5, j, s_m(h)) + \alpha(m)\} \text{ all possible states } (j, s_m(h)) = 50.$$  

The optimal schedule of this example is finally found as $\pi = \{J_4, J_3, J_5, J_1, J_2\}$ with the optimal value of 50.

Index the jobs such that $t_{11} \leq t_{21} \leq \cdots \leq t_{n1}$ for the case with $t_{j2} = t$. A similar dynamic programming algorithm with the same complexity can be easily constructed. We omit the details of the algorithm.

5. Heuristic for the general problem

In this section, we present a heuristic solution method for our problem, which is based on Johnson’s algorithm and first-only-empty $(n, c)$ (FOE$(n, c)$) algorithm [12]. FOE$(n, c)$ batching denotes the batching in which the first batch contains the first $[n – ([n/c] – 1)c]$ jobs and the next $([n/c] – 1)$ batches are all full of jobs up to the machine capacity (up to $c$ jobs).

Algorithm H

Step 1. Partition the jobs into two sets, with Set $N_1$ containing all the jobs with $t_{j1} + t_1 \leq t_{j2} + t_2$ and Set $N_2$ all the jobs with $t_{j1} + t_1 > t_{j2} + t_2$. 

Step 2. Form a permutation $\pi^H$ in which all jobs in Set $N_1$ precede each of those in Set $N_2$: the jobs in Set $N_1$ are sequenced in non-decreasing order of $t_1$ (shortest transportation time); while the jobs in Set $N_2$ are sequenced in non-increasing order of $t_2$ (longest transportation time). Then the entire job schedule $\pi^H$ is determined.

Step 3. Apply algorithm FOE($n,c$) to schedule the jobs on the batching machine.

Step 4. Evaluate $C_{\text{max}}(\pi^H)$ and $Z^H(\pi^H) = C_{\text{max}}(\pi^H) + \alpha(b^H)$ with respect to the coordinated problem.

Remark

(1) Jobs are transported by the crane one by one without idle time from Lemma 1.
(2) The running time of Algorithm $H$ is $O(n^2 \log n)$, since the application of Johnson’s algorithm and FOE algorithm dominates the other computations.
(3) We can obtain $b^H = b^* = \lceil n/c \rceil$ from algorithm FOE($n,c$).

Thus, we compare it with a lower bound on the optimal value in order to evaluate the performance of the heuristic. The following bounds are used.

One lower bound can be determined from the optimal solution with the case when $t_2 = 0$ for all $j$. Ignoring the transportation of the second-stage this also implies $t_2 = 0$. That is the optimal schedule can be obtained by STT–FOE when we ignore the transportation of the second-stage. The STT–FOE schedule is a permutation schedule that transports the jobs according to a shortest transportation time sequence on the crane, and processes $\lceil n/c \rceil$ batches on the machine. Let $C^2_{\text{max}}$ denote the minimum makespan for the problem when $t_2 = 0$ for all $j$. Thus the first lower bound is derived as

$$lb_1 = C^1_{\text{max}} + \min_{j \in \mathbb{N}} \{t_{j2}\}.$$  \hfill (12)

The problem when $t_1 = 0$ for all $j$ is equivalent to the problem with $t_2 = 0$ of minimizing makespan when the direction of time is reversed. Ignoring the transportation of the first-stage this also implies $t_1 = 0$. Last-only-empty($n,c$) (LOE($n,c$)) \cite{12} batching denotes the batching in which the first ($\lceil n/c \rceil − 1$) batches are all full of jobs up to the machine capacity (up to $c$ jobs) and the last batch contains the remaining $\lfloor n − (\lceil n/c \rceil − 1)c \rfloor$ jobs. It remains to determine the job sequence to be delivered on the vehicle. Suppose that we deliver the jobs according to the LTT (longest transportation time first) schedule. Notice that the LOE–LTT is an optimal schedule that avoids unnecessary machine idle time on the machine. Let $C^2_{\text{max}}$ denote the minimum makespan for the problem when $t_1 = 0$ for all $j$. Hence, the second lower bound is derived as

$$lb_2 = \min_{j \in \mathbb{N}} \{t_{j1}\} + C^2_{\text{max}}.$$  \hfill (13)

Consider the case when $p = 0$. Then the coordinated scheduling problem of production and transportation is reduced to the two-stage transportation problem. Applying Step 1 and Step 2 of Algorithm $H$, it follows that it is optimal for the problem. Let $C^3_{\text{max}}$ denote the minimum makespan for the problem when $p = 0$. $C^3_{\text{max}}$ can be written as

$$C^3_{\text{max}} = \max_{u \in \mathbb{N}} \left\{ \sum_{j=1}^{u} (t_{j1} + t_1) - t_1 + \sum_{j=u}^{n} (t_{j2} + t_2) - t_2 \right\}. \hfill (14)$$

The third lower bound is derived as

$$lb_3 = C^3_{\text{max}} + p.$$  \hfill (15)

The fourth lower bound is derived as

$$lb_4 = \max \left\{ \sum_{j=1}^{n} (t_{j1} + t_1) - t_1 + p + \min\{t_{j2}\}, \min\{t_{j1}\} + \lceil n/c \rceil p + \min\{t_{j2}\}, \min\{t_{j1}\} + \sum_{j=1}^{n} (t_{j2} + t_2) - t_2 + p \right\}. \hfill (16)$$
Based on the four lower bounds derives above, the overall lower bound can be derived as 
\[ LB = \max\{lb_1, lb_2, lb_3, lb_4\}. \]

The following theorem provides a performance guarantee of the Algorithm H.

**Theorem 3.** If \( \pi^H \) is the permutation generated by Algorithm H, then \( Z(\pi^H)/Z(\pi^*) \leq 2 \). Furthermore, the bound of 2 is tight.

**Proof.** Let \( C_{\text{max}}^* \) denote the makespan of the optimal schedule. For the permutation \( \pi^H \) found in Algorithm H, we can write the makespan for \( \pi^H \) as

\[ C_{\text{max}}(\pi^H) = \sum_{j=1}^s (t_{j1} + t_1) - t_1 + kp + \sum_{j=w}^n (t_{j2} + t_2) - t_2. \quad (17) \]

We refer to \( J_s \) and \( J_w \) as critical jobs in \( \pi^H \). \( J_s \) is defined by the last job of the batch containing \( J_s \). Once \( J_s \) arrives at the batching machine, the batch can start processing immediately. \( J_w \) is the first job of a sequence of jobs \{\( J_w, \ldots, J_n \)\} continuously delivered by the vehicle without time gap between adjacent jobs. \( k \) is the number of batches consecutively processed on the batching machine, dependent on \( J_s \) and \( J_w \).

We prove the theorem by considering the different cases that arise through the various possible choices of \( s \) and \( w \).

**Case 1.** \( s = w = 1 \).

In this case, \( J_s \) is processed immediately after it arrives at the machine, and \( J_s \) is delivered immediately after it is processed. We know the first batch is not full based on FOE(\( n, c \)) algorithm. So \( s = w = 1 \). We can rewrite (17) as

\[ C_{\text{max}}(\pi^H) = t_{s1} + p + \sum_{j=1}^n (t_{j2} + t_2) - t_2 \leq C_{\text{max}}^* + C_{\text{max}}^* = 2C_{\text{max}}^*, \]

Obviously, where \( \sum_{j=1}^n (t_{j2} + t_2) - t_2 < C_{\text{max}}^* \).

**Case 2.** \( s < w \).

Applying (14) and (16), we have

\[ C_{\text{max}}(\pi^H) = \sum_{j=1}^s (t_{j1} + t_1) - t_1 + kp + \sum_{j=w}^n (t_{j2} + t_2) - t_2 \leq \sum_{j=1}^w (t_{j1} + t_1) - t_1 + \sum_{j=w}^n (t_{j2} + t_2) - t_2 + [n/c]p \]

\[ \leq C_{\text{max}}^* + C_{\text{max}}^* \leq 2C_{\text{max}}^*. \]

**Case 3.** \( s > w \).

We know that \( J_w \) and \( J_s \) are in the same batch. Applying (16), we can rewrite (17) as

\[ C_{\text{max}}(\pi^H) = \sum_{j=1}^s (t_{j1} + t_1) - t_1 + p + \sum_{j=w}^n (t_{j2} + t_2) - t_2 \leq 2C_{\text{max}}^*. \]

Since no other cases are possible, the theorem is proved. We can obtain that

\[ C_{\text{max}}(\pi^H)/C_{\text{max}}^* \leq 2. \]

Hence,

\[ \frac{Z(\pi^H)}{Z(\pi^*)} \leq \frac{C_{\text{max}}(\pi^H) + z(b^*)}{C_{\text{max}}^* + z(b^*)} \leq 2, \]

where the last inequality comes from \( \frac{x+y}{y+z} \leq \frac{x}{z} \), for any \( z \geq 0 \) and \( 0 < y \leq x \).
We now prove the tightness of the upper bound of the heuristic algorithm. If an instance for the Algorithm H can be found such that the bound of the performance of the algorithm is exactly (or asymptotically) reached, then this bound is tight where the better legible bounds cannot be found. It is easy to find an instance for which the bound is asymptotically researched.

Consider an instance such that \( n = 3, \ t_1 = t_2 = 0, \ c = 2, \ \alpha(b) = 0, \ p = \varepsilon \) and the transportation times of jobs are given as

\[
\begin{align*}
    t_{11} &= 0, \quad t_{21} = \varepsilon, \quad t_{31} = 1 + \varepsilon, \quad t_{12} = \varepsilon, \quad t_{22} = 1 + \varepsilon, \quad t_{32} = 0,
\end{align*}
\]

where \( \varepsilon \) is a very small number. The optimal solution is to process \( J_1 \) in the first batch, \( J_2 \) in the second batch and then \( J_3 \) in the last batch on the batching machine, resulting \( Z^* = 1 + 3\varepsilon \). Algorithm H may process \( J_1 \) in the first batch and \( \{J_2, J_3\} \) in the second batch, resulting \( Z^H = 2 + 4\varepsilon \). Therefore, \( Z^H/Z^* = (2 + 4\varepsilon)/(1 + 3\varepsilon) \) and the ratio goes to 2 as \( \varepsilon \) approaches zero.

The worst-case analysis we presented, provides an upper bound on the performance ratio of the heuristic algorithm. However, from an actual point of view it may be too pessimistic, since the worst-case may be rarely occur.

6. Computational results

In this section, the computational experiments are conducted to test the performance of the proposed heuristic algorithm, and the computational results are reported to demonstrate the practical effectiveness of the algorithm. The algorithm is coded in Visual C++ language and implemented on the computer with 512MB RAM and 256KB L2 cache. The following parameters are considered to generate the random problem instances for testing:

- Number of jobs (\( n \)): 50, 100, 200, 500 and 1000;
- Capacity of the batch processing machine (\( c \)): 5, 10 and 15;
- Transportation times of jobs (\( t_{j1} \) and \( t_{j2} \)): being generated from the uniform distribution in the interval \([1,10]\) and \([1,50]\);
- Batch processing time (\( p \)): being generated from the uniform distribution in the interval \([1,10]\) and \([1,100]\);
- Returning time (\( t_1 \) and \( t_2 \)): being generated from the uniform distribution in the interval \([1,10]\);
- The setup cost \( \alpha(b) \): \( \alpha(b) = \beta b \), \( \beta \) is generated from the uniform distribution in the interval \([1,10]\) and \([1,100]\).

| \( c \) | \( n = 50 \) | \( n = 100 \) | \( n = 200 \) | \( n = 500 \) | \( n = 1000 \) | \( c = 5 \) | \( n = 50 \) | \( n = 100 \) | \( n = 200 \) | \( n = 500 \) | \( n = 1000 \) | \( c = 10 \) | \( n = 50 \) | \( n = 100 \) | \( n = 200 \) | \( n = 500 \) | \( n = 1000 \) | \( c = 15 \) | \( n = 50 \) | \( n = 100 \) | \( n = 200 \) | \( n = 500 \) | \( n = 1000 \) |
| \( t_{j1}, t_{j2}, p, \beta \in [1,10] \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) | \( \text{avg}(r) \) | \( \text{max}(r) \) |
| \( c = 5 \) | 3.10 | 5.29 | 1.45 | 2.49 | 7.05 | 11.45 | 4.22 | 6.76 | 3.49 | 5.46 | 2.66 | 4.24 | 1.39 | 2.87 | 0.54 | 1.83 | 0.67 | 1.53 | 0.16 | 0.32 | 0.41 | 0.65 | 0.12 | 0.24 |
For each combination of number of jobs and capacity of batching machine, we randomly generate 10 problem instances. Total 150 instances are generated with 15 combinations. For each instance, we compute $C_{\text{max}}^H$ and $LB$, where $C_{\text{max}}^H$ denotes the makespan of the schedule generated by the Algorithm H. We can obtain $b = b^H = b^{LB}$ from algorithm FOE(n, c) and LOE(n, c). The ratio error is defined as $r = 100* (Z^H - Z^{LB})/Z^{LB}$, where $Z^H = C_{\text{max}}^H + a(b)$ and $Z^{LB} = LB + a(b)$.

Table 1 summarizes the results of the computational experiments. In the table, the average and maximum values of $r$ which are denoted by $\text{avg}(r)$ and $\text{max}(r)$, respectively, are used to further evaluate the performance of the heuristic.

These computational results demonstrate that the Algorithm H is capable of generating near-optimal solutions within a reasonable amount of CPU time. As seen in the table, the ratio errors appear in a decreasing trend as the value of $n$ increases. One of its reasons may be that the lower bound increases as $n$ increases, but the difference between the objective value generated by the Algorithm H and the lower bound may not increase in proportion to the size of $n$. However, for the problem special instance when $n$ is close to $c$, the algorithm gets worse. This can be explained that the vehicle becomes the bottleneck facility when the jobs processed on the batching machine at a time increase. The computational experiments indicate that the proposed approach to the coordination scheduling is propitious to achieve a balance between the makespan and the total setup cost.

7. Conclusions

The coordination of production and transportation is an important issue in manufacturing and logistics management. In this paper, we have tackled a coordinated batch scheduling problem with two-stage transportation connecting production where a crane transports the jobs from the warehouse to the batching machine, and a vehicle delivers the jobs from the machine to the customer. Our goal is to optimize a combined objective function that considered makespan and total setup cost. In order to clearly describe the problem, a mixed integer programming model is presented. We show that the general problem is strongly NP-hard. For the case where the transportation times are equal on the vehicle or the crane, it is shown that the optimal schedule can be found in $O((cn)^3)$ time. An efficient heuristic for the general problem along with a tight worst-case error bound is presented and analyzed. We have also derived lower bound as comparison for the problem to evaluate the performance of the proposed heuristic. Furthermore, our numerical experiments show that the heuristic is effective in practice.

There are several possible extensions to this research. One future research is needed to analyze to include multiple cranes, multiple vehicles and multiple customer locations. Another future research may consider problems with other objective functions such as minimizing total completion time or minimizing maximum job tardiness and earliness.

Acknowledgements

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