Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

The coordination of transportation and batching scheduling

Lixin Tang *, Hua Gong

Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang 110004, China

ARTICLE INFO

Article history:
Received 27 August 2009
Received in revised form 21 December 2009
Accepted 12 January 2009
Available online 20 January 2009

Keywords:
Production scheduling
Transportation
Dynamic programming
Fully polynomial time approximation scheme

ABSTRACT

We study a coordinated scheduling problem of production and transportation in which each job is transported to a single batching machine for further processing. There are m vehicles that transport jobs from the holding area to the batching machine. Each vehicle can transport only one job at a time. The batching machine can process a batch of jobs simultaneously where there is an upper limit on the batch size. Each batch to be processed occurs a processing cost. The problem is to find a joint schedule of production and transportation such that the sum of the total completion time and the total processing cost is optimized. For a special case of the problem where the job assignment to the vehicles is predetermined, we provide a polynomial time algorithm. For the general problem, we prove that it is NP-hard (in the ordinary sense) and present a pseudo-polynomial time algorithm. A fully polynomial time approximation scheme for the general problem is obtained by converting an especially designed pseudo-polynomial dynamic programming algorithm.

1. Introduction

The coordination of production scheduling and transportation has recently received a lot of attention in logistics and manufacturing management research. Semi-finished jobs are transported from a holding area to a manufacturing facility for further processing by transporters in many manufacturing systems. Another motivation arises in many industries where the coordination of production and transportation can help to save energy and reduce fuel consumption. This is particularly true in the iron and steel industry. In the ingot production system, the ingots stripped are transported by some vehicles to a soaking pit. The soaking pit which is used to heat ingots can accommodate several ingots for processing at a time. Each batch of ingots to be heated in the soaking pit needs energy consumption. Due to the requirement of temperature and high energy consumption, the efficient coordination scheduling of production and transportation in the ingots system can improve working-in-processing inventories and reduce the total energy consumption.

In this paper, motivated by applications in the iron and steel industry, we study a coordinated scheduling problem of production and transportation. The jobs located at a holding area need to be transported by some vehicles to a batching machine for further processing. Each vehicle can transported one job at a time, and the batching machine can process several jobs at a time. Each batch of jobs to be processed on the batching machine occurs a processing cost. The problem is to find a joint schedule of production and transportation such that the objective is to minimize the sum of total completion time and total processing cost.

This problem integrates production scheduling for job processing on the batching machine and transportation scheduling for before processing. We briefly review batching machine scheduling problems and coordination scheduling problems of production and transportation, respectively.

* Corresponding author. Tel./fax: +86 24 83680169.
E-mail addresses: lixintang@mail.neu.edu.cn (L. Tang), gonghua1018@sina.com (H. Gong).

The batching machine scheduling is an important research topic. Recent reviews of batch scheduling research are provided by Potts and Kovalyov [1] and Brucker et al. [2]. The scheduling problems on the batching machine can be divided into two categories according to batch processing time pattern. In the first category, the processing time of a batch is dependent on the jobs grouped together in the batch. For the problem of minimizing the total completion time, the complexity of the problem is still open, but Chandru et al. [3] provide an optimal branch and bound algorithm and some efficient heuristics on single batching machine, Hochbaum and Landy [4] present a two-approximation algorithm, which is later improved by Cai et al. [5] to a polynomial time approximation scheme. For scheduling an unbounded batch machine to minimize the total weighted completion time with job release dates, Deng et al. [6] pinpoint the difficulty by first proving the problem with unbounded batch size to be NP-complete, and give a polynomial time approximation. Liu and Cheng [7] present a polynomial time approximation scheme for the total completion time with job release dates, and also present a fully polynomial time approximation scheme for scheduling an unbounded batch machine to minimize the total weighted completion time with job release dates. In the second category, the processing time of each batch is assumed to be fixed regardless of the jobs grouped together in the batch. Ahmadi et al. [8] have introduced two-machine flowshop problems with a discrete machine and a batching machine. Sung et al. [9], and Sung and Kim [10] have extended the two-machine flowshop scheduling problems of Ahmadi et al. These problems deal with the production part on the single batching machine with fixed batch processing time, and do not consider the coordination between transportation and production.

Another scheduling problems related to ours are found in joint production and transportation problems. Lee and Chen [11] review two types of transportation situations. The first type involves transporting a semi-finished job from one machine to another for further processing. The second type involves transporting a finished job to the customer or warehouse. They also study two types of transportation with the constraints on transportation capacity and transportation times. They study the class of scheduling problems by analyzing computationally complexity and proposing polynomial or pseudo-polynomial algorithms. For type-1 transportation, Lee and Strusevich [12] study the two-machine flowshop problem with a single interstage transporter of unlimited capacity. Chang and Lee [13] have extended type-2 transportation of Lee and Chen's work to the situation when each job occupies a different amount of space in the vehicle. Li and Ou [14] consider a single machine scheduling problem that takes into account the pickup arrangement of the materials and the delivery arrangement of finished jobs. In above production–transportation models, the machine configuration involves a single machine, parallel machines, flow shop or open shop.

There are also papers that address production and distribution scheduling from a combined distribution cost and batch delivery point of view. This environment is closely related to batching because all the completed jobs are delivered in a batch to the customer. Pundoor and Chen [15] study a production–distribution scheduling problem with one supplier and one or more customers in which the objective is to minimize the maximum delivery tardiness and the total distribution cost. Chen and Vairaktarakis [16] consider single machine and parallel machine scheduling problems with distribution scheduling and routing for delivery of completed jobs to the customers. Hall and Potts [17] analyze the complexity of some single machine scheduling problems with batch deliveries, but without a transporter availability constraint. Hall and Potts [18] consider various single and parallel machines scheduling problems where the various costs are based on the delivery times and delivery cost. Wang and Cheng [19] also consider a parallel machine scheduling problem with batch delivery costs.

Although production–transportation problems have been extensively studied in the literature, litter work has been done on the integration of transportation before processing and production on the single batching machine. Our work differs from the above models that we study not only the transportation schedule of semi-finished jobs, but also the production schedule of a batching machine. Motivated by problems in general manufacturing management, we do not assume zero returning time. Thus, we note that a vehicle in transportation stage may not be viewed as a real "machine". This is because it is occupied but not carrying any job as the vehicle is returning from the batching machine to the holding area. Hence, these vehicles are different from what real parallel machines are supposed to be in traditional scheduling problems. When the returning time is ignored in the transportation stage, the vehicle environment is equivalent with parallel machine environment. To the best of our knowledge, there exist the recent references neither on coordination of transportation and batching scheduling or on scheduling parallel machines followed by a batching machine.

In this paper, for a special case with given job assignment on the vehicle, we give a polynomial time algorithm. For the general problem, we pinpoint the difficulty by first proving the problem to be NP-hard. We also provide a pseudo-polynomial time algorithm to solve this problem, and further show that this problem is ordinarily NP-hard. Finally, via the scaling technique and the proposed dynamic programming, we provide a fully polynomial approximation scheme for the general problem.

The organization of this paper is as follows. In the next section, we describe the problem and provide optimality properties. In Section 3, we provide a polynomial time algorithm to solve a special case. In Section 4, we prove the NP-hardness of the general problem. Then we present a pseudo-polynomial time algorithm for the general problem in Section 5. In Section 6, we present a fully polynomial time approximation scheme (FPTAS). The last section contains a conclusion and some suggestions for future research.

2. Statement of problem

In the following we describe the problem under consideration. Our problem consists of a transportation stage from the holding area to the batching machine and a production stage on the batching machine.

(1) A set of jobs located at a holding area are transported to a batching machine for further processing.
(2) In the transportation stage, there are a number of vehicles available where each vehicle can deliver only one job at a time. All vehicles are initially located at the holding area.
(3) The transportation time of a job is job-dependent.
(4) In the production stage, the batching machine can process several jobs simultaneously as a batch. The maximum number of jobs that can be processed simultaneously in the batching machine is called capacity of that batching machine.
(5) The time needed to process a batch of jobs on the batching machine is denoted by a constant regardless of the jobs grouped together in the batch.
(6) Once processing of a batch is initiated, it cannot be interrupted and other jobs cannot be introduced into the machine until processing is completed.
(7) Each batch to be processed on the batching machine occurs a processing cost.
(8) All jobs are available in the holding area at time 0.

Next, we introduce the notation to be used in this paper.

n	number of jobs
m	number of vehicles
t_j	transportation time of job j from the holding area to the batching machine, $j = 1, 2, \ldots, n$
t	empty moving time of each vehicle from the batching machine back to the holding area
p	processing time of a batch on the batching machine
c	capacity of the batching machine
b	number of batches to be processed on the batching machine
B_l	set of jobs in batch l, $l = 1, \ldots, b$
b_l	number of jobs processed in batch l, $l = 1, \ldots, b$
$\alpha(b)$	processing cost function
C_j	completion time of job j on the batching machine, $j = 1, 2, \ldots, n$
$F = \sum C_j + \alpha(b)$	objective function

In situation where the dimension on the two measurements $\sum C_j$ and $\alpha(b)$ is difficult to unify, we may adjust cost function $\alpha(b)$ to uniform dimension with total completion time. For ease of presentation, denote our problem as *TBS* (transportation and batching scheduling problem).

Next, we present some optimality properties for problem *TBS*.

Lemma 1. *There exists an optimal schedule for TBS such that there is no idle time between the jobs transported on each vehicle in the transportation part.*

Proof. If there exists idle time, we can always move the subsequent jobs earlier without increasing the objective value. □

The following result describes a candidate set of possible starting time points on the batching machine.

Lemma 2. *There exists an optimal schedule for TBS in which the starting time of each batch on the batching machine is made either at the arrival time of some job on the machine or immediately at a time when the machine becomes available.*

Proof. Assume that there is the starting time of some batch which is scheduled neither at the arrival time of some job on the machine nor at a time when the machine becomes available. This starting time can be changed to the latest earlier time which fits either of those conditions. Since the same jobs can be processed at that earlier time, and there are no additional operations, the objective value is not increased. □

Lemma 3. *There exists an optimal schedule $\pi^* = \{B_1, B_2, \ldots, B_b\}$ for TBS in which: (1) all jobs assigned to the same vehicle are scheduled in the non-decreasing order of transportation times. (2) B_l contains all jobs which arrive at the machine in the time interval $(S_{l-1}, S_l]$ if the number of the jobs is no more than c, for $l = 2, \ldots, b$.*

Proof. (1) Assume that jobs J_i and J_j are assigned to the same vehicle and J_j follows J_i immediately such that $t_i \geqslant t_j$ in π^*. Let π' be a schedule obtained by swapping J_i and J_j. We consider two different cases that arise through the possible batch choices of J_i and J_j. *Case (i):* J_i and J_j are processed in the same batch. It is easy to see that $F(\pi^*) = F(\pi')$. *Case (ii):* J_i and J_j are not in the same batch. Without loss of generality, assume that $J_i \in B_l$ and $J_j \in B_{l+1}$ in π^*. The starting time of batch B_l in π' is less than or equal to the starting time in π^* due to Lemma 2 and $t_i \geqslant t_j$. This implies that $\sum C(\pi') \leqslant \sum C(\pi^*)$. Thus, we obtain $F(\pi') \leqslant F(\pi^*)$, regardless of whether J_i and J_j are processed in the same batch or not.

(2) Index all jobs in the increasing order of their arrival times on the batching machine in π^*. Assume that the batches are numbered in accordance with their start times such that $S_1 < S_2 < \cdots < S_b$, and the number of the jobs in each batch is not greater than the machine capacity. Let J_j be the first job in B_{l+1}, and J_j arrives at the machine at time r_j, such that $r_j \leqslant S_l$. Let π' be a schedule obtained by simply assigning J_j to B_l, then $C'_j = S_l + p < S_{l+1} + p = C_j$. The remaining jobs in π' are not changed.

It is obvious that $F(\pi') \leqslant F(\pi^*)$, which is a contradiction. We can show that there exists an optimal schedule in which B_l consists of a number of jobs which finish transportation contiguously in the time interval $(S_{l-1}, S_l]$. □

3. A polynomial time algorithm for a special case

In this section, we consider a special case where the job assignment to the vehicles is predetermined. It is evident that the problem reduces to an optimal batching problem in this case. This special case characterizes the practical situation where each vehicle is dedicated to a special group of jobs. Now we can provide a dynamic programming algorithm to solve the optimal batching problem as follows.

Schedule the jobs on each vehicle in non-decreasing transportation time order based on the Lemma 3, and then re-index all jobs in accordance with the job arrival time on the batching machine, i.e., $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_n$. It suffices to consider one job sequence and apply it to the processing of jobs on the machine. So the starting time of each batch on the machine need to be decided, and this can be done by dynamic programming. When the machine finishes one batch, it will either process a new batch immediately or wait until the last job of new batch arrives. In the first case, 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 case, the starting time is r_j, for some job j. We can see that in the first case, x can be traced back and the starting time can be expressed as $r_j + qp$ for some $q \leqslant n - j$ and some job j. Hence, the possible starting times of the machine can be $r_j, r_j + p, \ldots, r_j + qp$, for some $q \leqslant n - j$, and $j = 1, 2, \ldots, n$. Let s_l denote the actual time of batch B_l corresponding to starting time point h, for $h = r_j, r_j + p, \ldots, r_j + qp$, $l = 1, 2, \ldots, b$, where $\lceil n/c \rceil \leqslant b \leqslant n$ and $s_0 = 0$.

Define $f(k, j, s_l)$ as the minimal total completion time to schedule the first k jobs $1, 2, \ldots, k$, provided that the current last batch contains jobs $j, j+1, \ldots, k$ and starts to be processed at time s_l where $k - j + 1 \leqslant c$ and $s_l \geqslant r_k$. If we know the available time of the batching machine before we process jobs $j, j+1, \ldots, k$, then the increase of total completion time due to jobs $j, j+1, \ldots, k$ is actually fixed. Namely, $f(k, j, s_l)$ that satisfies the following three properties:

(i) $0 < k - j + 1 \leqslant c$;
(ii) $s_l = r_j, r_j + p, \ldots, r_j + qp$, and $s_l - s_{l-1} \geqslant p$, for $l = 2, 3, \ldots, b$;
(iii) $\lceil k/c \rceil \leqslant l \leqslant k$;

otherwise, $f(k, j, s_l) = \infty$.

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

$$f(k, j, s_l) = \min\{f(j-1, i, s_{l-1}) + (k - j + 1)(s_l + p) | \text{all possible states } (i, s_{l-1})\},$$

where $j - 1, i, s_{l-1}$ satisfy the conditions (i), (ii) and (iii) described above.

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

$$F(n) = \min\{f(n, j, s_b) + \alpha(b) | \text{all possible states } (j, s_b)\}.$$

By recording all the necessary information in the above process, an optimal schedule can be calculated. From the above description and analysis, it is also not difficult to see that the time complexity of the algorithm is $O(cn^3)$ time.

We now demonstrate the above solution method with a numerical example.

Example. Consider the instance with $J = \{J_1, J_2, J_3, J_4\}$, $m = 2$, $t = 1$, $c = 3$, $p = 5$, $t_1 = 1$, $t_2 = 4$, $t_3 = 2$, $t_4 = 4$, and $\alpha(b) = 6b$. We assume that J_1 and J_2 are transported by one vehicle, J_3 and J_4 are transported by another vehicle. From the above method, we know $r_1 = 1$, $r_2 = 2$, $r_3 = 6$, and $r_4 = 7$. We have the following results:

$$f(1, 1, s_1) = f(0, 0, 0) + r_1 + p = 6;$$
$$f(2, 1, s_1) = f(0, 0, 0) + 2(r_2 + p) = 14;$$
$$f(2, 2, s_2) = f(1, 1, s_1) + (s_2 + p) = 17;$$
$$f(3, 1, s_1) = f(0, 0, 0) + 3(r_3 + p) = 33;$$
$$f(3, 2, s_2) = f(1, 1, s_1) + 2(s_2 + p) = 28;$$
$$f(3, 3, s_2) = f(2, 1, s_1) + (s_2 + p) = 26;$$
$$f(3, 3, s_3) = f(2, 2, s_2) + (s_3 + p) = 33;$$
$$f(4, 2, s_2) = f(1, 1, s_1) + 3(r_4 + p) = 42;$$
$$f(4, 3, s_2) = f(2, 1, s_1) + 2(s_2 + p) = 38;$$
$$f(4, 4, s_2) = f(3, 1, s_1) + (s_2 + p) = 49;$$
$$f(4, 3, s_3) = f(2, 2, s_2) + (s_3 + p) = 49;$$
$$f(4, 4, s_3) = \min \begin{cases} f(3, 2, s_2) + (s_3 + p) \\ f(3, 3, s_2) + (s_3 + p) \end{cases} = 43;$$
$$f(4, 4, s_4) = f(3, 3, s_3) + (s_4 + p) = 54;$$
$$F(4) = \min\{f(4, j, s_b) + \alpha(b) | \text{ all possible states } (j, s_b)\} = 50.$$

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

4. Analysis of the NP-hardness

In this section, we show that for *TBS* is NP-hard. This is done by reducing the NP-complete Partition problem (see [20]) to the decision version of *TBS*.

Partition problem: Given h items, $H = \{1, 2, \ldots, h\}$, each item $j \in H$ has a positive integer size a_j, such that $\sum_{j=1}^{h} a_j = 2a$, for some integer a. The question asks if there are two disjoint subsets G and $H \setminus G$, such that $\sum_{j \in G} a_j = \sum_{j \in H \setminus G} a_j = a$.

The following theorem states the computational complexity of *TBS*.

Theorem 1. *The problem TBS is NP-hard even if m = 2.*

Proof. To any instance of the Partition problem, we construct an instance of *TBS* as follows. There are $n = 2h$ jobs split into two groups: the *P*-jobs (Partition jobs) denoted by $P_j,\ j = 1, 2, \ldots, h$, the *X*-jobs (auxiliary jobs) denoted by $X_j,\ j = 1, \ldots, h$. Their transportation times and other parameters are given by the formulas:

Transportation times: $t_{p_j} = a_j,\ t_{X_j} = 0,\ j = 1, \ldots, h$;
Processing time: $p = a$;
Returning time: $t = 0$;
Processing cost: $\alpha(b) = 3bah$;
Machine capacity: $c = h$;
Threshold value: $y = 9ah$.

We are going to show that for the constructed scheduling problem instance, a schedule π with $\sum C_j + \alpha(b) \leqslant y$ exists if and only if the Partition problem has a solution.

→If there is a solution to the Partition problem instance, we show that there is a schedule π with $\sum C_j + \alpha(b) \leqslant y$ for the above-constructed instance of *TBS*. Suppose that the Partition problem instance has a solution G_1 and G_2. Now we can construct the following schedule π:

Vehicle 1 transports the jobs of G_1 one by one, and vehicle 2 transports the jobs of G_2 one by one. Let T_u denote the total running time of vehicle u, for $u = 1, 2$. We can see that $T_1 = T_2 = a$. Since the transportation times of *X*-jobs are equal to 0, the batching machine can first process these jobs. Due to $c = h$, the *X*-jobs are processed as the first batch and the *P*-jobs are processed as the second batch. Then the total completion time of all the jobs is $3ha$. It is easy to check that $\sum C_j + \alpha(b) \leqslant y$ (see Fig. 1).

←Now we show that if there exists a schedule π with $\sum C_j + \alpha(b) \leqslant y$ to the instance of *TBS*, then the Partition problem has a solution. First, it is easy to see that, in an optimal schedule for the constructed instance of *TBS*, the following properties hold: (i) schedule π exactly contains two batches. (ii) The first batch only contains all *X*-jobs and the starting time of the first batch is 0. (iii) All *P*-jobs are processed as the second batch on the batching machine. Next, we prove three properties as follows:

(i) Suppose that there are q batches in schedule π. Due to $c = h$, we have $q \geqslant 2$. Without loss of generality, we assume that $q = 3$. Obviously, the total processing cost is $9ah$. Then the objective function of schedule π is more than y, which is a contradiction. Thus, due to the above inequality, schedule π cannot have three or more batches. Hence, schedule π exactly contains two batches and each batch contains h jobs. We also obtain that total processing cost of schedule π is $6ah$. This will then imply that $3ah$ is an upper bound of total completion time.

(ii) Since the transportation times of *X*-jobs are equal to 0, we can see that the earliest possible starting time on the batching machine is 0. Denote S_1 and S_2 as the starting time of the first batch and the second batch on the batching machine, respectively. Note that $S_1 + a \leqslant S_2$. Since the processing time of each batch on the machine is a, we have

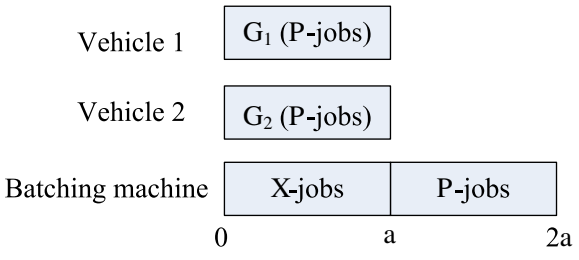

Fig. 1. A schedule for the instance of problem *TBS*.

$\sum C_i = h(S_1 + a) + h(S_2 + a) \leqslant 3ah$. Hence, we obtain that $S_1 = 0$ and $S_2 = a$. At time zero, there are only X-jobs available. Thus, the batching machine processes the X-jobs as the first batch at time 0. The total completion time of the first batch is ah.

(iii) From (i) and (ii), we know that the second batch in schedule π contains P-jobs. We also can know that the P-jobs must arrive at the batching machine before time a.

Let G_1 and G_2 be a Partition of P-jobs. We assume that vehicle 1 transports the jobs of G_1 one by one, and vehicle 2 transports the jobs of G_2 one by one. Let T_u denote the total running time of the vehicle u, for $u = 1, 2$. Based on above discussion, it is easy to see that the starting time of the second batch is $\max\{T_1, T_2, a\}$ where $T_1 = \sum_{j\in G_1} a_j \leqslant a$ and $T_2 = \sum_{j\in G_2} a_j \leqslant a$.

If $T_1 = \sum_{j\in G_1} a_j < a$, then it implies $T_2 = \sum_{j\in G_2} a_j > a$, which is a contradiction. Hence, we have $T_1 = T_2 = a$, Thus, it must be true that $\sum_{j\in G_1} a_j = \sum_{j\in G_2} a_j = a$. Then it is easy to see that G_1 and G_2 form a solution to the Partition problem instance.

Combining the "if" part and the "only if" part, we have proved the theorem. □

5. A pseudo-polynomial dynamic programming algorithm

In this section, we provide a pseudo-polynomial-time dynamic programming algorithm to solve problem *TBS*. An algorithm for a problem P is pseudo-polynomial if it solves any instance I of P in time bounded by a polynomial in $|I|$ and number (I), where $|I|$ is the size of the instance I and number (I) is the largest integer appearing in I. For a NP-hard problem, the existence and derivation of such a pseudo-polynomial algorithm means that this problem is NP-hard in the ordinary sense. Unless $P = NP$, there can be no pseudo-polynomial algorithm for any strongly NP-hard problem. In this paper, it is of interest to derive a pseudo-polynomial time algorithm and show that our problem is NP-hard in the ordinary sense, not strongly NP-hard.

An extension of dynamic programming algorithm described in Section 3 can solve this problem. Based on Lemmas 1 and 3, we know that the jobs assigned to each vehicle are scheduled consecutively in nondecreasing transportation time order. Now, we need to determine the job assignment on the vehicles and job batching on the batching machine. The starting time of a job on the batching machine is influenced by the job assignment on the vehicle. When the job batching is determined, the total processing cost can be determined.

Index the jobs such that $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n$. Assume that T is the total running time of all vehicles, that is $T = \sum_{i=1}^{n} t_i + (n - m)t$. Then we can see that the total running time of any vehicle is no more than T. For each vehicle, there are only a finite number of possible departure time points for transporting jobs. These possible departure times of each vehicle from the holding area can be $0, 1, \ldots, T$. Based on Lemma 2, when the batching machine is idle, it will either process a new batch immediately or wait until the arrival time of a job before processing the batch that contains that job. Thus, the starting time of a batch on the batching machine is determined by the completion time of the previous batch on the machine and the completion time of this batch on vehicles. To describe the dynamic programming procedure, some auxiliary functions are introduced as follows.

Define $f_l(k, j, D_u, S_l)$ as the minimum total completion time if we have scheduled jobs $1, \ldots, k$ subject to the following constraints: the current total running time of vehicle u is D_u, for $u = 1, 2, \ldots, m$; the current last batch B_l contains jobs j, $j + 1, \ldots, k$; the starting time of batch B_l is S_l on the batching machine, for $l = 1, \ldots, b$ and $b \in \{\lceil n/c \rceil, \ldots, n\}$. Based on the above discussion, batch B_l will be satisfying with the following three properties:

(1) $S_l - S_{l-1} \geqslant p$ for $l = 2, \ldots, b$.
(2) $S_l \geqslant D_u$ for $l = 1, 2, \ldots, b$.
(3) $0 < k - j + 1 \leqslant c$.

Then the increase of the total completion time due to jobs $j, j + 1, \ldots, k$ is $(k - j + 1)(S_l + p)$. The algorithm is formally described as follows.

Algorithm DP

Renumber the jobs in the non-decreasing transportation time order, i.e., $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_n$.
Recursive relations:

$$f_l(k, j, D_u, S_l) = \min_{\substack{0<k-j+1\leqslant c \\ \lceil k/c\rceil \leqslant l \leqslant b}} \{f_{l-1}(j-1, i, D_u - t_v, S_{l-1}) + (k - j + 1)(S_l + p), |u = 1, \ldots, m;\ v = j, \ldots, k\},$$

where $S_l = \begin{cases} D_u, & l = 1, \quad u = 1, \ldots, m, \\ \max\{S_{l-1} + p, D_u\}, & l = 2, \ldots, b, \quad u = 1, \ldots, m. \end{cases}$

Initial conditions:

$$f_l(k,j,D_u,S_l)=\begin{cases}0, & k=j=0,\quad D_1=D_2=\cdots=D_m=0,\quad S_l=0,\\ \infty, & \text{otherwise},\end{cases}$$

for $b\in\{\lceil n/c\rceil,\ldots,n\}$ and $D_u=0,1,\ldots,T$.

We emphasize that $f_l(k,j,D_u,S_l)$ takes value $+\infty$ when no feasible schedule with $b_l>c$ exists.

Optimal solution:

$$F^*=\min\{f_b(n,j,D_u,S_b)+\alpha(b)|b\in\lceil n/c\rceil,\ldots,n\}.$$

Theorem 2. *Algorithm DP finds an optimal schedule for problem TBS in $O(cmn^2T^{2(m-1)})$ time.*

Proof. Based on Lemma 3, there exists an optimal schedule with jobs assigned to each vehicle in the non-decreasing transportation time order. The job index in the first step ensures this optimal property. In these recursive formulas, to guarantee the optimality of $f_l(k,j,D_u,S_l)$, we enumerate all possibilities of the positions on the vehicles for jobs $j,\ldots,k$. We also have justified the validity of the recursive relations based on the above discussions of D_u and S_l. Hence, Algorithm DP can solve problem *TBS*.

The time complexity of the algorithm can be established as follows. We observe that $m-1$ of the value $D_1,D_2,\ldots,D_m$ are independent, the number of different states of the recursive relations is at most T^{m-1}. Since the calculation of each S_l have to perform $D_1,\ldots,D_m$, each state requires $O(mT^{m-1})$. By definition, $k-j+1\leqslant c$, $k,j\leqslant n$, and $1\leqslant l\leqslant b\leqslant n$. Each state requires $O(cn^2)$. Therefore, the overall time complexity of Algorithm DP is $O(cmn^2T^{2(m-1)})$. □

Hence, from the above description and analysis, we obtain the following theorem.

Theorem 3. *Problem TBS is NP-hard in the ordinary sense.*

6. An FPTAS

In this section, we provide a fully polynomial approximation scheme for our problem *TBS*. Recall, that a polynomial time approximation scheme (PTAS) for a problem is a $(1+\varepsilon)$-approximation algorithm if we have $F\leqslant(1+\varepsilon)F^*$ for all instances, where F^* denotes the optimal solution value and F denotes the value of the solution given by the algorithm. Furthermore, if the time complexity of a PTAS is polynomial in $1/\varepsilon$, then it is called a fully polynomial time approximation scheme (FPTAS). In pure technical sense, an FPTAS is a best one may tackle to solve an NP-hard optimization problem, unless $P=NP$.

The general outline of an FPTAS for solving a problem can be stated as follows. We formulate a certain scaled problem for the original problem. The pseudo-polynomial algorithm for the original problem is applied to this scaled problem with its parameter. This algorithm provides a fully polynomial approximation scheme for the original problem. This scaling technique has been already successfully applied to design a fully polynomial approximation schedule for 0–1 Knapsack problem (see [21]). There are recent applications of this scaling technique for some scheduling problems (for example, [22,23]). In this paper, we formulate a scaled problem for *TBS* and prove that any efficient algorithm for the scaled problem is a 1+ε-approximation algorithm for the original problem *TBS*. Next, we will turn the pseudo-polynomial time algorithm using scaling technique into an FPTAS for *TBS*.

An FPTAS for problem *TBS*

Step 1: given an arbitrary $\varepsilon>0$, let $\delta=\varepsilon T/n(3n-1)$.

Step 2: formulate a scaled problem. Define new transportation time and processing time of each job j:

$$t'_j=\left\lfloor\frac{t_j}{\delta}\right\rfloor,\quad t'=\left\lfloor\frac{t}{\delta}\right\rfloor,\quad p'=\left\lfloor\frac{p}{\delta}\right\rfloor,\qquad j=1,2,\ldots,n.$$

Step 3: for the instance of the scaled problem, use the pseudo-polynomial time algorithm DP to obtain a minimal objective value schedule π' of jobs.

Step 4: take π' as an approximate solution to the original problem instance.

For every $b=\lceil n/c\rceil,\ldots,n$, suppose that we have found an schedule π' such that its total completion time, denoted by $f'_b(\pi')=f'_b(n,j,D'_u,S'_b)$, is minimized, for the scaled problem by Algorithm DP in Section 5. For every $b=\lceil n/c\rceil,\ldots,n$, denote by π^* an schedule to the original problem such that its total completion time is minimized, and set $f^*_b(\pi^*)=f^*_b(n,j,D_u,S_b)$. For ease of expression, we use $f_b(\pi)$ to substitute $f_b(n,j,D_u,S_b)$.

Theorem 4. *The objective value of the schedule found by Algorithm DP for the scaled problem is at most a factor of $1+\varepsilon$ above the objective value of the optimal schedule found by the Algorithm DP for the original problem TBS.*

Proof. Since the original problem and the scaled problem have the same set of restrictions, π' exists if and only if π^* exists. The original problem and the scaled problem have the same total processing cost. It remains to demonstrate that $f_b(\pi')\leqslant(1+\varepsilon)f^*_b(\pi^*)$.

Note that the schedule has at most n transportations and $n-1$ returns on each vehicle. Increase each transportation time t'_j by $t_j/\delta - t'_j$, which increases C_j by at most n. Increase each returning time t by $t/\delta - t$, which increases C_j by at most $n-1$. Increase each processing time p' by $p/\delta - p'$, which increases C_j by at most n. Further, the total completion time increases at most $n(3n-1)$. Denote $f_b^0(\pi) = f_b(\pi)/\delta$ as a function obtained from $f_b(\pi)$ by substituting t_j/δ, t/δ and p/δ for t_j, t and p, respectively, $j = 1, 2, \ldots, n$. Thus, we can get a schedule with respect to t_j, t and p, and its objective value is given by

$$f_b(\pi') = f_b^0(\pi')\delta \leqslant f'_b(\pi')\delta + n(3n-1)\delta.$$

We have $f'_b(\pi') \leqslant f'_b(\pi^*)$ due to the definition of π'. It follows that $f'_b(\pi^*) \leqslant f^*_b(\pi^*)/\delta$ from $\lfloor x \rfloor \leqslant x$ for any number x. Combining established relations in one chain, for every $b = \lceil n/c \rceil, \ldots, n$, we obtain

$$f_b(\pi') \leqslant \delta f'_b(\pi^*) + n(3n-1)\delta \leqslant f^*_b(\pi^*) + \varepsilon T \leqslant (1+\varepsilon) f^*_b(\pi^*).$$

This implies that

$$\min_{\lceil n/c \rceil \leqslant b \leqslant \cdots \leqslant n} \{f_b(\pi') + \alpha(b)\} \leqslant \min_{\lceil n/c \rceil \leqslant b \leqslant \cdots \leqslant n} \{(1+\varepsilon) f^*_b(\pi^*) + \alpha(b)\}.$$

Thus, we can see that the approximate solution is at most a factor of $1+\varepsilon$ away from the optimum solution. □

The time complexity of the approximation scheme is dominated by the step to solve the scaled problem. Let $T' = \sum_{j=1}^{n} t'_j + (n-m)t'$. It is easy to see that T' holds that

$$T' \leqslant \frac{\sum_{j=1}^{n} t_j + (n-m)t}{\delta} = \frac{T}{\delta} = \frac{n(3n-1)}{\varepsilon}.$$

Thus, the running time of the approximation scheme is bounded by

$$O\left(cmn^2 (T')^{2(m-1)}\right) \leqslant O\left(cmn^2 \left(\frac{n(3n-1)}{\varepsilon}\right)^{2(m-1)}\right) \leqslant O\left(cmn^2 \left(\frac{3n^2}{\varepsilon}\right)^{2(m-1)}\right),$$

which is polynomial for given m and $1/\varepsilon$.

Combining Theorem 4 and the above time complexity, we summarize our main result as the following statement.

Theorem 5. *There exists an FPTAS with the running time $O(cmn^2(3n^2/\varepsilon)^{2(m-1)})$ for problem TBS, where m is the number of vehicles.*

7. Concluding remarks

In this paper, we study a batch machine scheduling problem that incorporates transportation before processing. Our goal is to optimize a combined objective function that considers the total completion time and the total processing cost. We prove that this problem is NP-hard by a reduction from Partition problem and further prove that it is ordinarily NP-hard by providing a pseudo-polynomial time algorithm. The existence of an FPTAS for this problem is established. We also provide a polynomial time algorithm to solve a special case where the job assignment to the vehicles is predetermined.

There are several possible extensions to this research. First, it is interesting to investigate the problems with other objective functions such as minimizing makespan or minimizing maximum job tardiness. Another interesting issue is to develop effective heuristics to solve the general problem and investigate polynomial time algorithms for some special cases.

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

[1] C.N. Potts, M.Y. Kovalyov, Scheduling with batching: a review, Eur. J. Oper. Res. 120 (2000) 228–249.
[2] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, S.L. van de Velde, Scheduling a batching machine, J. Schedul. 1 (1) (1998) 31–54.
[3] V. Chandru, C.Y. Lee, R. Uzoy, Minimizing total completion time on batch processing machines, Int. J. Prod. Res. 31 (1993) 2097–2121.
[4] D.S. Hochbaum, D. Landy, Scheduling semiconductor burn-in operations to minimize total flowtime, Oper. Res. 45 (1997) 874–885.
[5] M. Cai, X. Deng, H. Feng, G. Li, G. Liu. A PTAS for minimizing total completion time of bounded batch scheduling. Lect. Notes Comput. Sci. 2337 (IPCO'2002), MIT2002, 304–314.
[6] X.T. Deng, H.D. Feng, P.X. Zhang, Y.Z. Zhang, H. Zhu, Minimizing mean completion time in a batch processing system, Algorithmica 38 (2004) 513–528.
[7] Z.L. Liu, T.C.E. Cheng, Approximation schemes for minimizing total (weighted) completion time with release dated on a batch machine, Theor. Comput. Sci. 347 (2005) 288–298.
[8] J.H. Ahmadi, R.H. Ahmadi, S. Dasu, C.S. Tang, Batching and scheduling jobs on batch and discrete processors, Oper. Res. 39 (1992) 750–763.
[9] C.S. Sung, Y.H. Kim, S.H. Yoon, A problem reduction and decomposition approach for scheduling for a flowshop of a batch processing machines, Eur. J. Oper. Res. 121 (2000) 179–192.

[10] C.S. Sung, Y.H. Kim, Minimizing makespan in a two-machine flowshop with dynamic arrivals allowed, Comput. Oper. Res. 29 (2002) 275–294.
[11] C.Y. Lee, Z.L. Chen, Machine scheduling with transportation considerations, J. Schedul. 4 (2001) 3–24.
[12] C.Y. Lee, V.A. Strusevich, Two-machine shop scheduling with an uncapacited interstage transporter, IIE Trans. 37 (2005) 725–736.
[13] Y.C. Chang, C.Y. Lee, Machine scheduling with job delivery coordination, Eur. J. Oper. Res. 158 (2004) 470–487.
[14] C.L. Li, J. Ou, Machine scheduling with pickup and delivery, Naval Res. Logis. 52 (2005) 617–630.
[15] G. Pundoor, Z.L. Chen, Scheduling a production–distribution system to optimize the tradeoff between delivery tardiness and distribution cost, Naval Res. Logis. 52 (2005) 571–589.
[16] Z.L. Chen, G.L. Vairaktarakis, Integrated scheduling of production and distribution operations, Manage. Sci. 51 (2005) 614–628.
[17] N.G. Hall, C.N. Potts, Supply chain scheduling: batching and delivery, Oper. Res. 51 (2003) 566–584.
[18] N.G. Hall, C.N. Potts, The coordination of scheduling and batch deliveries, Ann. Oper. Res. 135 (2005) 41–64.
[19] G. Wang, T.C.E. Cheng, Parallel machine scheduling with batch delivery costs, Int. J. Prod. Econom. 68 (2005) 177–183.
[20] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[21] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications, New York, 1998.
[22] J.J. Yuan, Z.H. Liu, C.T. Ng, T.C.E. Cheng, The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan, Theor. Comput. Sci. 320 (2004) 199–212.
[23] Z.H. Liu, T.C.E. Cheng, Scheduling with job release dates, delivery times and preemption penalties, Inf. Process. Lett. 82 (2002) 107–111.