Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 1921–1939
computers & operations research
www.elsevier.com/locate/dsw

The complexity of customer order scheduling problems on parallel machines

Jaehwan Yang

Department of Business Administration, University of Incheon, 177 Dohwa-dong, Nam-gu, Incheon 402-749, Republic of Korea

Abstract

This paper considers scheduling problems where a set of jobs (customer order) is shipped at the same time. The objective function is associated with the completion time of the orders. While a machine can process only one job at a time, multiple machines can process simultaneously jobs in an order. We first introduce this relatively new class of the customer order scheduling problems on parallel machines. Then, we establish the complexity of several problems with different types of objectives, job restrictions, and machine environments. For some tractable cases, we propose optimal solution procedures.

Keywords: Batch scheduling; Kitting; Customer order scheduling; Computational complexity

1. Introduction

This paper considers customer order scheduling problems where each order (batch) consists of a prespecifed set of products (jobs). While batching of jobs does not occur during processing, each set of jobs (customer order) is shipped as one batch. Orders and machines are available for scheduling at the beginning of the time horizon. Several jobs in an order can be processed by different machines at the same time. Similar to many scheduling problems, there is no setup time between jobs. While batch scheduling problems frequently have setup times between batches, for the applications we describe below, ignoring setup time is reasonable. This is because jobs are only shipped in batches but not processed in batches. If required, sequence-independent setup times can be included in the processing times of the individual jobs. Further, the assembly and packaging time of the entire batch can be included as part of shipping. Also, this problem is different from most other batch scheduling problems because the objective is associated with the completion time of the batches instead of the

E-mail address: jaehyang@dreamwiz.com (J. Yang).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.12.008

completion time of each job. The completion time of a batch is the completion time of the final job in the batch.

Motivation for these relatively new types of batch scheduling problems comes from several real-world applications. For instance, consider scheduling customer orders at a manufacturing company that produces different types of products. A customer can order a combination of the types of products. The composition of the order is prespecifed by a purchase order. The company does not ship the order until all the products in the order are completed. The major concern of the company is to ship the order to each customer as soon as possible. Each order is a batch and a product is a job. This is a common type of situation that exists in industry (Julien and Magazine [1]).

A second application of our problem can be found in automotive repair shops. Each car has at least one broken part. A mechanic represents a machine. A car and each broken part can be considered as a batch and a job, respectively. If a car has more than one broken part, then more than one mechanic may work simultaneously on the car. A car stays in the repair shop until all problems are fixed.

A third application of our problem is found in the production of components for subsequent assembly. For instance, consider an electronics manufacturing facility where computer monitors are produced. Generally, this is a multi-stage production process including several fabrication operations, and a final assembly operation. The fabrication operations create various components including printed circuit boards, front and back cover cabinets, and color display tubes. The components are then "kitted" into a batch, where the batch is the set of parts required to produce one type of a monitor. The final assembly operation is not started until all required components are finished by the fabrication operations. The performance of the fabrication operations is associated with completion time of the set of components needed to produce a monitor and not with the completion time of each component.

A final example is crane scheduling at a port. Each ship has several holds and each hold requires a processing time for loading or unloading. In this case, we can consider a crane as a machine, a ship is a batch, and a hold is a job. Only one crane can load or unload one hold. More than one crane can work simultaneously on the holds of a ship. The capacity of a port is fixed, and is expressed as the number of holds, not the number of ships. A ship remains at port until all work on ship is completed.

We only examine literature which considers batch scheduling problems, where the composition of batches is prespecifed. Jordan [2] gives a survey about other types of batch scheduling problems. Potts and Wassenhove [3] and Webster and Baker [4] provide reviews of batching and lot-sizing decision problems.

The paper by Julien and Magazine [1] is probably the first one to consider the problem where the objective is associated with batches instead of jobs. Each order (batch) contains several jobs, and the jobs are processed on a single machine. They assume a job-dependent setup time between two different types of jobs. The objective is to minimize the total completion time of orders. Julien and Magazine provide a Dynamic Programming (DP) algorithm for the problem when there exist only two types of jobs and the batch processing order is fixed. The computational complexity of the DP algorithm is $O(b^2)$ where b is the number of batches. A similar problem, where the batch processing order is not specified, is studied by Coffman et al. [5]. Baker [6] examines a problem where one job type has the property that those jobs processed during the same setup are not available until the completion of the setup. Santos and Magazine [7] describe this restriction as *batch availability*.

Gupta et al. [8] consider a single machine problem where an order must contain one job from each of several classes. In addition, there is a setup time between two jobs of different classes. Problems where each batch has one common job and one distinct job are considered by Gerodimos et al. [9].

Scheduling cranes at a port is studied by Daganzo [10] and Peterkofsky and Daganzo [11]. Both papers consider the scheduling problem as an open shop with identical machines, where jobs consist of independent, single stage, preemptable tasks. The objective function is the minimization of the weighted tardiness. Daganzo [10] applies the *weighted shortest processing time first* rule to obtain optimal solutions for some special cases. Three basic scheduling principles are developed and used in a heuristic procedure. Six problems are used to evaluate performance of the heuristic. Peterkofsky and Daganzo [11] develop a branch and bound solution procedure. However, they do not provide a theoretical basis for their method.

Blocher and Chhajed [12] examine one of the problems studied in this work. They show that minimizing the average batch completion time is NP-hard in a parallel machine environment, and develop several heuristics and bounds. Blocher et al. [13] extend the problem to a job shop. Yang [14] considers the same problem with a fixed batch sequence and develops a DP which runs in pseudo-polynomial time.

The remainder of this work is organized as follows. In Section 2, we introduce some notation. We provide preliminary results in Section 3. In Section 4, we examine two identical parallel machine problems. We show that the recognition version of the problem with the objective of minimizing average completion time is binary NP-complete even if the number of jobs in each batch is fixed. Two variations of the two machine problem are studied. First, we consider the problem with a fixed batch sequence. Fixed batch sequences occur when a manufacturer employs a policy to determine processing order of batches. Two such policies are when batches are processed in *first come first served* (FCFS) order and when high-priority batches are processed first.

Another variation is when there exists a fixed job–machine assignment. Since each job is preassigned to a machine, we only need to determine the sequence of jobs on each machine. For example, consider a manufacturing facility that produces air conditioners. A unique combination of an indoor fan and a compressor is a batch. In the facility, the fan and compressor are produced separately on different machines, but they are shipped together as one final product. This problem models those situations where different types of jobs require different machines. Because different job types may require different tasks, not all job types can be processed on all machines. Since a given machine performs similar tasks, there is no setup time between two different jobs or batches. Roemer and Ahmadi [15] consider this problem and show that the recognition version of the problem is unary (strongly) NP-complete. We suggest an easier and shorter complexity proof for the problem.

In Section 5, we study problems where the number of batches is arbitrary. As a special case, we consider the problem where processing times of all jobs are equal. For this case, we show that there exists an optimal schedule where jobs are assigned to the first available machine for the objectives of minimizing weighted average completion time of batches and minimizing the maximum lateness of batches. Then, we develop optimal solution procedures for these problems where processing time is arbitrary. We finally show that the recognition version of the problem with the objective of minimizing weighted average batch completion time is binary NP-complete.

In Section 6, we examine problems where there are restrictions on the number of jobs simultaneously in the machine shop. A typical example of this problem comes from crane scheduling problems at a port. The size of a ship is determined by the number of holds in each ship, and the

size of a berth limits the total number of holds in the ships staying at the berth simultaneously. Therefore, even though several small ships can stay at the berth simultaneously, only one or two big ships can stay at the berth simultaneously. The ship is a batch and the hold is a job. We show that two machine problems with the objectives of minimizing the makespan and minimizing the average batch completion time are both unary NP-complete. Finally, we summarize our work and discuss possible future research.

2. Notation

The decision variables for our batch completion time models are

$\sigma_k =$ schedule of all jobs on machine k for $k \in M$,

$\sigma =$ schedule of all jobs $= (\sigma_1, \sigma_2, \ldots, \sigma_m)$.

Other notation used in this work include:

$n =$ number of jobs
$N =$ set of jobs $= \{1, 2, \ldots, n\}$
$b =$ number of batches
$B =$ set of batches $= \{1, 2, \ldots, b\}$
$n_i =$ number of jobs in batch i for $i \in B$
$B_i =$ set of jobs in batch i for $i \in B = \{\sum_{j=0}^{i-1} n_j + 1,\ \sum_{j=0}^{i-1} n_j + 2, \ldots,\ \sum_{j=0}^{i} n_j\}$, where $n_0 = 0$
$m =$ number of machines
$M =$ set of machines $= \{1, 2, \ldots, m\}$
$w_i =$ weight of batch i for $i \in B$
$p_j =$ processing time of job j for $j \in N$
$P_i = \sum_{j \in B_i} p_j =$ total processing time of batch $i \in B$ for $i \in B$ and $k \in M$
$K =$ maximum possible number of jobs simultaneously in the shop where $K \geqslant m$, i.e. if at some instant job set E is being processed and $T = \{i \in B | B_i \cap E \neq \emptyset\}$, then $\sum_{i \in T} n_i \leqslant K$
$C_i(\sigma_k) =$ completion time of batch i on machine k for $i \in B$ and $k \in M$
$C_i(\sigma) =$ completion time of batch i in schedule σ for $i \in B = \max_{k \in M} C_i(\sigma_k)$.

We classify our problem according to three factors: machine environment, job and batch characteristics, and objective function (Graham et al. [16]). To simplify notation, we represent $C_i(\sigma)$ as C_i when there is no ambiguity. Also, we use C_{B_i} to describe batch completion time problems in the three field notation to eliminate the confusion between our problems and classical scheduling problems. A brief discussion of each of these factors and three-field notation is presented.

(1) *Machine environment*: There are both one machine and multiple machine models. In the multiple machine model, the machines process jobs in parallel. Machine speeds are identical, proportional, or unrelated. In the first field, P, Q, and R denote identical, proportional, and unrelated parallel machine speeds, respectively. A number after P, Q, or R indicates a fixed number of machines rather than an arbitrary number.

(2) *Batch and job characteristics*: Each batch can contain multiple jobs, and each machine can process at most one job at a time. The jobs in a batch can be processed simultaneously. Also, the machine shop may have a fixed capacity. If this is the case, then the number of jobs that can be in the shop simultaneously is limited. To designate problems with restrictions on the number of jobs in a machine shop at the same time, we place "K" in the second field.

(3) *Objective function*: The third field denotes the objective to be minimized. The objective criteria that we study are minimizing the last batch to complete ($C_{B_{\max}}$), the maximum batch lateness ($L_{B_{\max}}$), the sum of batch completion times ($\sum C_{B_i}$), and the sum of weighted batch completion times ($\sum w_i C_{B_i}$). Note that for these problems, *the batch completion time* and not the job completion time is being considered. Without K, $C_{B_{\max}}$ is equivalent to $C_{\max}$. Hence, we only consider the minimization of $C_{B_{\max}}$ with K. For instance, the problem of minimizing the sum of batch completion times on two parallel identical machines and with a restriction on the number of jobs in the machine shop at the same time is written as $P2|K|\sum C_{B_i}$.

The following rule is used in this work to select the job processing order.

SB (*Shortest batch rule*): When a machine becomes available, an unscheduled job in the batch with a shortest total processing time is selected for processing.

An SB schedule is a schedule generated using the SB rule.

3. Preliminary results

In this section, we first review some properties of an optimal schedule. A *regular measure* is any nondecreasing function of job completion times (Rinnooy Kan [17]). Since there are no restrictions that delay jobs, we have the following result.

Lemma 3.1. *For scheduling problems with regular measures, and either no job restrictions or just a restriction on the number of jobs simultaneously in the shop, there exists an optimal schedule without inserted idle time.*

Proof. Similar to the proof found in Yang and Posner [18] for the problem $P\|\sum C_{B_i}$. □

We say that batch $i \in B$ is *separated* if on some machine $k \in M$, jobs in batch i are not processed consecutively.

Lemma 3.2. *For scheduling problems with regular measures, and no job restrictions or just a restriction on the number of jobs simultaneously in the shop, there exists an optimal schedule where no batch is separated.*

Proof. Similar to the proof found in Blocher and Chhajed [12] for the problem $P\|\sum C_{B_i}$. □

As a result of Lemma 3.2, we assume batches are not separated in an optimal schedule. We now present another property of batch scheduling problems.

Lemma 3.3 (Blocher and Chhajed [12]). *For batch scheduling problems with a regular measure, in an optimal schedule, each machine processes the batches which are processed on multiple machines in the same order.*

The following result describes the relationship between the complexity of a batch scheduling problem and the corresponding classical scheduling problem.

Remark 3.1. A batch scheduling problem is at least as hard as the corresponding classical scheduling problem.

Proof. Let $n_i = 1$ for $i = 1, 2, \ldots, b$. Then, the batch scheduling problem is equivalent to the corresponding classical scheduling problem. □

4. Two machines

In this section, we present results for two machine batch scheduling problems where there is no restriction on the number of jobs in the shop at the same time.

4.1. Problem $P2\|\sum C_{B_i}$

We show that problem $P2\|\sum C_{B_i}$ remains binary NP-complete for some processing restictions. For the case where b is fixed, we have the following result.

Remark 4.1 (Blocher and Chhajed [12]). The recognition version of $P2\|\sum C_{B_i}$ is binary NP-complete even if b is a constant.

Blocher and Chhajed [12] let $b = 1$, but let n_1 increase to establish the complexity of the problem. Now, we show that if we restrict the n_i, but let b increase, then a similar result holds. We consider the case when $n_i \in \{1, 2\}$ for all $i \in B$. First, a preliminary result is needed.

Remark 4.2. If each batch contains only one job, then the SB schedule is optimal for problem $P\|\sum C_{B_i}$.

Proof. If each batch has only one job, then problem $P\|\sum C_j$ is identical to problem $P\|\sum C_{B_i}$. Also, for this case the SB rule is the same as the shortest processing time (SPT) rule. Since SPT produces an optimal schedule for problem $P\|\sum C_j$ (Smith [19]), SB produces an optimal schedule for problem $P\|\sum C_{B_i}$. □

Now, we establish the complexity of problem $P2\|\sum C_{B_i}$ with $n_i \in \{1, 2\}$ for all $i \in B$ by reduction from the following binary NP-complete problem.

Even–Odd Partition (Garey and Johnson [20]). Given a set of 2ℓ positive integers $A = \{a_1, a_2, \ldots, a_{2\ell}\}$ such that $a_1 < a_2 < \cdots < a_{2\ell}$, does there exist a partition of A into two subsets A_1 and

A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$, and such that for each $1 \leqslant i \leqslant \ell$, A_1 contains exactly one of $\{a_{2i-1}, a_{2i}\}$?

Theorem 4.1. *The recognition version of problem* $P2\|\sum C_{B_i}$ *is binary* NP*-complete even if* $n_i \in \{1,2\}$ *for all* $i \in B$.

Proof. Given an instance of Even–Odd Partition, we construct the following instance of problem $P2\|\sum C_{B_i}$:

$$
\begin{aligned}
n &= 2\ell + 2,\\
b &= 2\ell + 1,\\
n_i &= 1, \quad i = 1,2,\ldots,2\ell,\\
n_{2\ell+1} &= 2,\\
p_j &= a_j, \quad j = 1,2,\ldots,2\ell,\\
p_j &= \sum_{i=1}^{2\ell} a_i, \quad j = 2\ell+1, 2\ell+2,\\
z &= \sum_{j=1}^{\ell} (\ell - j + 1)(a_{2j-1} + a_{2j}) + 3\sum_{i=1}^{2\ell} a_i/2.
\end{aligned}
$$

The recognition version of problem $P2\|\sum C_{B_i}$ is in NP since given any schedule, we can calculate $\sum C_i$ in polynomial time. We prove that there exists a solution to Even–Odd Partition if and only if there exists a schedule for problem $P2\|\sum C_{B_i}$ with $\sum C_i \leqslant z$.

($\Rightarrow$) Suppose that there exists a solution, (A_1, A_2), to Even–Odd Partition. Assign batches (jobs) in A_1 and A_2 to machines 1 and 2 in increasing index order, respectively. Then, assign one job in batch $2\ell + 1$ to each machine. Since even–odd partition has a solution, both jobs in batch $2\ell + 1$ start at time $(\sum_{i=1}^{2\ell} p_i)/2 = \sum_{i=1}^{2\ell} a_i/2$. Hence, the completion time of batch $2\ell + 1$ is $3\sum_{i=1}^{2\ell} a_i/2$. The total completion time is

$$
\begin{aligned}
\sum_{i=1}^{2\ell+1} C_i &= \ell a_1 + \ell a_2 + (\ell - 1)a_3 + (\ell - 1)a_4 + \cdots + a_{2\ell-1} + a_{2\ell} + 3\sum_{i=1}^{2\ell} a_i/2\\
&= \sum_{j=1}^{\ell} (\ell - j + 1)(a_{2j-1} + a_{2j}) + 3\sum_{i=1}^{2\ell} a_i/2\\
&= z.
\end{aligned}
$$

See Fig. 1 for an example of an optimal schedule.

($\Leftarrow$) Suppose that there exists a schedule for problem $P2\|\sum C_{B_i}$ with $\sum C_i \leqslant z$. Since the processing time of each job in batch $2\ell + 1$ is as large as the summation of the processing times of all jobs not in this batch, each machine processes exactly one job of batch $2\ell + 1$ in an optimal schedule. If either job $2\ell + 1$ or $2\ell + 2$ is not processed last on their respective machine, then we

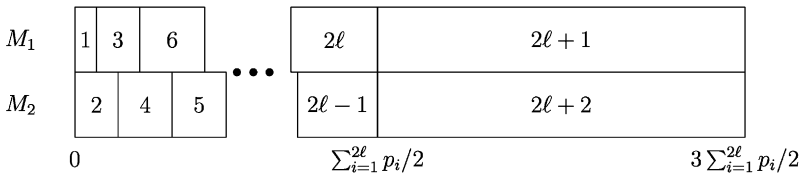

Fig. 1. An example of an optimal schedule for Theorem 4.1.

can reduce the total batch completion time by at least $\sum_{i=1}^{2\ell} a_i - a_{2\ell}$ by moving the appropriate job to the end of the schedule.

From Remark 4.2, the SB rule provides an optimal schedule for batches $1,2,\ldots,2\ell$. The total completion time generated by the SB rule for batches $1,2,\ldots,2\ell$ is $\sum_{i=1}^{2\ell} C_i = \sum_{j=1}^{\ell} (\ell - j + 1)(a_{2j-1} + a_{2j})$. Observe that in the schedule generated by SB, switching the processing machine of batch $2j-1$ with batch $2j$ for $1 \leqslant j \leqslant \ell$ does not change $\sum_{i=1}^{2\ell} C_i$. Furthermore, to satisfy the condition that the total completion time $\leqslant z$, batch $2\ell+1$ must start no later than time $(\sum_{i=1}^{2\ell} p_i)/2 = (\sum_{i=1}^{2\ell} a_i)/2$. This is only possible if the jobs of $1,2,\ldots,2\ell$ form an even–odd partition. □

4.2. Problem $P2\|\sum C_{B_i}$ with a fixed batch sequence

Consider the problem where batch sequence π is fixed. In this problem, if batch i precedes batch j in π for $i,j \in B$, then each job in batch i starts no later than start time of batch j. From Lemma 3.1, there always exists an optimal schedule without inserted idle time. Hence, we only need to determine an optimal machine–job assignment. Yang [14] shows that the recognition version of $P2\|\sum C_{B_i}$ with a fixed batch sequence is binary NP-complete even if b is a constant or n_i is given for all $i \in B$. He also develops a DP algorithm which runs in $O(\bar{p}^2 n \sum_{i \in B} P_i)$ where $\bar{p} = \max_{j \in N} \{p_j\}$.

Remark 4.3 (Yang [14]). The recognition version of $P2\|\sum C_{B_i}$ with a fixed batch sequence is pseudopolynomial.

4.3. Problem $P2\|\sum C_{B_i}$ with a fixed machine–job assignment

Consider problem $P2\|\sum C_{B_i}$ where the assignment of jobs to machines is specified. Since each job is preassigned to a machine, we only need to determine the sequence of jobs on each machine. Recall that Lemma 3.3 implies that we only consider schedules where batches that re processed by both machines are processed in the same order. Consequently, to obtain an optimal schedule, we only need to determine an optimal batch sequence.

The following result is due to Roemer and Ahmadi [15].

Theorem 4.2. *The recognition version of* $P2\|\sum C_{B_i}$ *with a fixed machine–job assignment is unary* NP-*complete*.

However, the proof is difficult and long. An easier and more intuitive proof is given in the appendix.

Corollary 4.1. *The recognition version of* $R2||\sum C_{B_i}$ *is unary* NP-*complete.*

Proof. If each job has a large cost on one of the two machines, then $P2||\sum C_{B_i}$ with fixed machine–job assignment reduces to $R2||\sum C_{B_i}$. □

5. Arbitrary number of machines

In this section, we examine problems with arbitrary number of machines.

5.1. Problem $P||\sum C_{B_i}$

Consider the problem where there exists an arbitrary number of machines. Blocher and Chhajed [12] establish the following complexity result for problem $P||\sum C_{B_i}$.

Remark 5.1 (Blocher and Chhajed [12]). The recognition version of $P||\sum C_{B_i}$ is unary NP-complete.

5.2. Problems with equal processing time

We consider the restriction that the processing times of all jobs are equal. For the problems considered in this work, we assume without loss of generality that $p_j = 1$ for all j.

From Lemma 3.2, there exists an optimal schedule without batch separation for $P|p_j = 1|L_{B_{\max}}$ and $P|p_j = 1|\sum w_i C_{B_i}$. For problems in this section, jobs from a given batch that are processed on a specified machine can be processed in any order. Since the order of processing within a batch does not change the cost, a schedule is completely defined by a batch sequence and a machine–job assignment. We begin by presenting a property of an optimal schedule for problems with unit processing times.

Theorem 5.1. *For* $P|p_j = 1|L_{B_{\max}}$ *and* $P|p_j = 1|\sum w_i C_{B_i}$, *suppose we are given an optimal batch sequence. Then, there exists an associated optimal schedule where if batch* i *precedes batch* ℓ, *then no job in* ℓ *starts before a job in* i.

Proof. We first establish the result for $P|p_j = 1|\sum w_i C_{B_i}$. Let σ^* be an optimal schedule, and let λ^* be an optimal *completion time order* of the batches. We reindex the batches so that $\lambda^* = (1, 2, \ldots, b)$. Let σ be a schedule where λ^* is the completion time order of the batches and where the jobs satisfy the conditions of the theorem. Consequently, for $i, i+1 \in B$, batches i and $i+1$ finish processing consecutively in σ and $C_i(\sigma) \leqslant C_i(\sigma^*)$. Also, the total processing time of the jobs processed before $C_\ell(\sigma^*)$ in σ^* is always greater than or equal to the total processing time of the jobs processed before $C_\ell(\sigma)$ in σ. Hence, $C_\ell(\sigma) \leqslant C_\ell(\sigma^*)$. Therefore, $w_i C_i(\sigma^0) + w_\ell C_\ell(\sigma) \leqslant w_i C_i(\sigma^*) + w_\ell C_\ell(\sigma^*)$. We can apply the same argument to the remaining batches. This establishes the result for $P|p_j = 1|\sum w_i C_{B_i}$.

A similar proof applies for $P|p_j = 1|L_{B_{\max}}$. □

As a result of Theorem 5.1, we assume that jobs are always assigned to the first available machine for $P|p_j = 1|L_{B_{\max}}$ and for $P|p_j = 1|\sum w_i C_{B_i}$. Consequently, a batch sequence is sufficient to describe a schedule for these problems.

The next two results establish optimal solution procedures for $P|p_j=1|\sum C_{B_i}$ and $P|p_j=1|L_{B_{\max}}$, respectively.

Theorem 5.2. *For $P|p_j=1|\sum C_{B_i}$, an SB schedule is optimal.*

Proof. We use an adjacent batch interchange argument. Suppose that σ^* is an optimal schedule but is not an SB schedule. Then, there exist two batches i and ℓ in σ^* such that batches i and ℓ are processed consecutively with i processed before ℓ but $n_i > n_\ell$ for $i,\ell \in B$. Let $t=$ time when batch i starts in σ^*. Let σ be the same schedule as σ^* except at time t, batch ℓ is processed first. Suppose that at time t, $m_1 \leqslant m$ machines are available to start processing new batches.

In σ^*, $C_i(\sigma^*) = t + 1 + \lceil (n_i - m_1)/m \rceil$ and $C_\ell(\sigma^*) = t + 1 + \lceil (n_i + n_\ell - m_1)/m \rceil$. In σ, $C_\ell(\sigma) = t + 1 + \lceil (n_\ell - m_1)/m \rceil$ and $C_i(\sigma) = t + 1 + \lceil (n_i + n_\ell - m_1)/m \rceil$. Notice also that $C_i(\sigma) = C_\ell(\sigma^*)$ and $C_\ell(\sigma) \leqslant C_i(\sigma^*)$ since $n_i > n_\ell$. Therefore, the total completion time of σ is no larger than that of σ^*. We can repeat this argument for every other pair of batches that are not in SB order. Since there are a finite number of batches, the process terminates after a finite number of interchanges with an optimal SB schedule. □

Theorem 5.3. *For $P|p_j=1|L_{B_{\max}}$, an EDD (a batch with earliest due date first) schedule is optimal.*

Proof. Follows from an adjacent batch interchange argument. □

Now, we establish the complexity of $P|p_j=1|\sum w_i C_{B_i}$ by reduction from the following binary NP-complete problem.

Partition (Garey and Johnson [20]). Given a set of 2ℓ positive integers $A = \{a_1, a_2, \ldots, a_{2\ell}\}$ such that $a_1 < a_2 < \cdots < a_{2\ell}$, does there exist a partition of A into two subsets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$?

Theorem 5.4. *The recognition version of $P|p_j=1|\sum w_i C_{B_i}$ is binary* NP*-complete.*

Proof. Given an instance of Partition, we construct the following instance of $P|p_j=1|\sum w_i C_{B_i}$:

$$n = \sum_{i=1}^{2\ell} a_i,$$

$$m = \sum_{i=1}^{2\ell} a_i/2,$$

$$b = 2\ell,$$

$$n_i = a_i, \quad i = 1,2,\ldots,2\ell,$$

$$B_i = \{a_{i-1}+1, a_{i-1}+2, \ldots, a_i\}, \quad i = 1,2,\ldots,2\ell,$$

$$w_i = a_i, \quad i = 1,2,\ldots,2\ell,$$

$$z = 3\sum_{i=1}^{2\ell} a_i/2,$$

where $a_0 = 0$.

The recognition version of $P|p_j = 1|\sum w_i C_{B_i}$ is in NP since we can calculate $\sum w_i C_i$ in polynomial time.

We prove that there exists a solution to Partition if and only if there exists a solution to $P|p_j = 1|\sum w_i C_{B_i}$ with $\sum w_i C_i \leqslant z$.

($\Rightarrow$) Suppose that there exists a solution, (A_1, A_2), to Partition. Assign all batches (jobs) in A_1 to first available machines. Then, assign all batches (jobs) in A_2 to first available machines. Since Partition has a solution, all jobs in A_1 finish at time 1 and all jobs in A_2 finish at time 2. The total weighted completion time is

$$\begin{aligned}\sum_{i=1}^{2\ell} w_i C_i &= \sum_{i\in A_1} w_i + \sum_{i\in A_2} 2w_i \\ &= 3\sum_{i=1}^{2\ell} a_i/2 \\ &= z.\end{aligned}$$

($\Leftarrow$) Suppose that there exists a schedule for problem $P|p_j = 1|\sum w_i C_{B_i}$ with $\sum w_i C_i \leqslant z$. From Theorem 5.1, there exists at most one batch which starts at time 0 and completes at time 2. The other batches either start at time 0 and finish at time 1 or start at time 1 and finish at time 2. Suppose there exists a batch $k \in B$ that starts at time 0 and finishes at time 2. We reindex batches such that batch k starts at time 0 and finishes at time 2, batches $1,2,\ldots,k-1$ start at time 0 and finish at time 1, and batches $k+1, k+2, \ldots, 2\ell$ start at time 1 and finish at time 2 for $k \in B$.

Then, the total weighted completion time is

$$\begin{aligned}\sum_{i=1}^{2\ell} w_i C_i &= w_1 + w_2 + \cdots + w_{k-1} + 2w_k + 2w_{k+1} + 2w_{k+2} + \cdots + 2w_{2\ell} \\ &= \sum_{q=1}^{k-1} w_q + 2\left(\sum_{i=1}^{2\ell} a_i/2 - \sum_{q=1}^{k-1} w_q\right) + 2\left(\sum_{i=1}^{2\ell} a_i/2 - \sum_{q=k+1}^{2\ell} w_q\right) + \sum_{q=k+1}^{2\ell} 2w_q \\ &> \sum_{q=1}^{k-1} w_q + \left(\sum_{i=1}^{2\ell} a_i/2 - \sum_{q=1}^{k-1} w_q\right) + 2\left(\sum_{i=1}^{2\ell} a_i/2 - \sum_{q=k+1}^{2\ell} w_q\right) + \sum_{q=k+1}^{2\ell} 2w_q \\ &= z.\end{aligned}$$

Contradiction. Hence, all batches are either start at time 0 and finish at time 1 or start at time 1 and finish at time 2. This is only possible if the jobs of $1,2,\ldots,2\ell$ form a partition. □

6. Two machine problems with restriction K

In this section, we establish the complexity of $P2|K|C_{B_{\max}}$ and $P2|K|\sum C_{B_i}$. Recall that if K is specified, then the maximum number of jobs in those batches where processing is simultaneously occurring is K.

6.1. Problem $P2|K|C_{B_{\max}}$

First, we establish the complexity of $P2|K|C_{B_{\max}}$.

Theorem 6.1. *The recognition version of* $P2|K|C_{B_{\max}}$ *is unary* NP*-complete.*

Proof. Given an instance of 3-Partition, we construct the following instance of $P2|K|C_{B_{\max}}$:

$$
\begin{aligned}
n &= 8\ell,\\
b &= 5\ell,\\
n_i &= 2, \quad i = 1,2,\ldots,3\ell,\\
n_i &= 1, \quad i = 3\ell+1, 3\ell+2, \ldots 5\ell,\\
B_i &= \{2i-1, 2i\}, \quad i = 1,2,\ldots,3\ell,\\
B_i &= \{i+3\ell\}, \quad i = 3\ell+1, 3\ell+2, \ldots, 5\ell,\\
p_j &= 5L + y/3 - a_{(j+1)/2}, \quad j = 1,3,\ldots,6\ell-1,\\
p_j &= L - y/3 + a_{j/2}, \quad j = 2,4,\ldots,6\ell,\\
p_j &= 6L, \quad j = 6\ell+1, 6\ell+2, \ldots, 8\ell,\\
K &= 3,\\
z &= 15\ell L,
\end{aligned}
$$

where $L = \sum_{i=1}^{3\ell} a_i$.

The recognition version of $P2|K|C_{B_{\max}}$ is in NP since we can calculate $C_{B_{\max}}$ in polynomial time.

We prove that there exists a solution to 3-Partition if and only if there exists a solution to $P2|K|C_{B_{\max}}$ with $C_{B_{\max}} \leqslant z$.

($\Rightarrow$) Consider the schedule, $\sigma = (\sigma_1, \sigma_2)$ shown in Fig. 2, where

$$\sigma_1 = (1, 3, \ldots, 6\ell - 1),$$

$$\sigma_2 = (2, 6\ell+1, 4, 6\ell+2, 6; 8, 6\ell+3, 10, 6\ell+4, 12; \ldots; 6\ell-4, 8\ell-1, 6\ell-2, 8\ell, 6\ell).$$

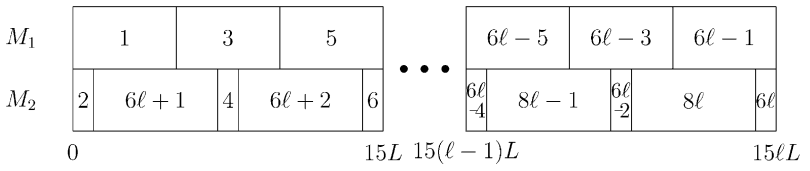

Fig. 2. An example of an optimal schedule.

Let $\beta_i =$ first job in B_i for $i \in B$. Since $a_{3j-2} + a_{3j-1} + a_{3j} = y$ for $j = 1, 2, \ldots, \ell$, the maximum completion time is

$$\begin{aligned} C_{B_{\max}} &= \sum_{i=1}^{3\ell} p_{\beta_i} \\ &= \sum_{i=1}^{3\ell} \left(5L + \frac{y}{3} - a_i\right) \\ &= 15\ell L \\ &= z. \end{aligned}$$

($\Leftarrow$) Since batches $3\ell + 1, 3\ell + 2, \ldots, 5\ell$ are identical, we assume that they are processed in index order. Let I_1 and I_2 be sets of batches such that $I_1 = \{1, 2, \ldots, 3\ell\}$ and $I_2 = \{3\ell + 1, 3\ell + 2, \ldots, 5\ell\}$.

Since $\sum_{j \in N} p_j = 30\ell L$, to satisfy $C_{B_{\max}} \leqslant z$, the schedule must be optimal and have no idle time.

Because $K = 3$ and $n_i = 2$ for $i \in I_1$, two batches in I_1 cannot be processed simultaneously. Since $\sum_{j=1}^{3\ell} p_{2j-1} = 15\ell L$, exactly one job of $\{1, 3, \ldots, 6\ell - 1\}$ must be in process at each point in time. Without loss of generality, we assume that these jobs are processed on machine 1. For each batch $i \in I_1$, job $2i$ must start no earlier than start time of job $2i - 1$ and finish processing no later than the completion time of job $2i - 1$ to prevent idle time.

In an optimal schedule, let the batches in I_1 be processed in the order $v_1, v_2, \ldots, v_{3\ell}$. Further, for notational convenience, we let $v_1 = 1, v_2 = 2, \ldots, v_{3\ell} = 3\ell$. We first assign job 1 to machine 1 and job 2 to machine 2. Now, due to the capacity constraint, neither job 3 nor job 4 can start on machine 2 until job 1 completes on machine 1. Hence, it is optimal to schedule job $6\ell + 1$ on machine 2. Using a similar argument, job $6\ell + 2$ must be processed between jobs 4 and 6.

If jobs 5 and 6 do not complete at the same time, then to avoid idle time, the only available job that satisfies our assumptions and the capacity constraint is job $6\ell + 3$. However, processing this job would delay the processing of job 8 until after job 7 completes. This creates idle time because batches 4 and 5 can not be processed simultaneously. Consequently, jobs 5 and 6 must complete at the same time.

We can repeat this argument to establish that both jobs of batches $3, 6, \ldots, 3\ell$ must complete at the same time. This is only possible if there exists a solution to 3-Partition. □

6.2. Problem $P2|K|\sum C_{B_i}$

Following theorem establishes the complexity of $P2|K|\sum C_{B_i}$.

Theorem 6.2. *The recognition version of* $P2|K|\sum C_{B_i}$ *is unary* NP-*complete.*

Proof. Given an instance of 3-Partition, we construct the following instance of $P2|K|\sum C_{B_i}$:

$$n = 8\ell,$$
$$b = 5\ell,$$

$$n_i = 2, \quad i = 1,2,\ldots,3\ell,$$

$$n_i = 1, \quad i = 3\ell+1, 3\ell+2, \ldots 5\ell,$$

$$B_i = \{2i-1, 2i\}, \quad i = 1,2,\ldots,3\ell,$$

$$B_i = \{i+3\ell\}, \quad i = 3\ell+1, 3\ell+2, \ldots, 5\ell,$$

$$p_j = 5L + y/3 - a_{(j+1)/2}, \quad j = 1,3,\ldots,6\ell-1,$$

$$p_j = L - y/3 + a_{j/2}, \quad j = 2,4,\ldots,6\ell,$$

$$p_j = 6L, \quad j = 6\ell+1, 6\ell+2, \ldots, 8\ell,$$

$$K = 3,$$

$$z = (75\ell^2 + 27\ell)L/2,$$

where $L = \sum_{i=1}^{3\ell} a_i$.

The recognition version of $P2|K|\sum C_{B_i}$ is in NP since given any schedule, we can calculate $\sum C_i$ in polynomial time.

We prove that there exists a solution to 3-Partition if and only if there exists a solution to $P2|K|\sum C_{B_i}$ with $\sum C_i \leqslant z$.

($\Rightarrow$) We use the same schedule as in ($\Rightarrow$) part of the proof of Theorem 4.2. Thus, $\sum_{i=1}^{5\ell} C_i \leqslant z$.

($\Leftarrow$) Let $I_1 = \{1,2,\ldots,3\ell\}$ and $I_2 = \{3\ell+1, 3\ell+2, \ldots, 5\ell\}$. Two different batches in I_1 cannot be processed simultaneously because $n_i = 2$ for $i \in I_1$ and $K = 3$. Thus, we assume without loss of generality that jobs in $\{1,3,\ldots,6\ell-1\}$, jobs with longer processing time from I_1, are processed on one machine, say machine 1.

By using a similar argument in ($\Leftarrow$) part of the proof of Theorem 6.1, we can show that in an optimal schedule, jobs in $\{2,4,\ldots,6\ell\}$ and I_2 are processed on machine 2. Then, from the proof of Theorem 4.2, the schedule shown in Fig. 2 is optimal. The remainder of the proof is similar to ($\Leftarrow$) part of the proof of Theorem 4.2. □

7. Summary

We summarize our work in Table 1. The entries "π" and "J_F" in the Variation column imply that the batch sequence is fixed and the machine–job assignment is fixed, respectively. An "R" in the Variation denotes unrelated parallel machine speed. A blank in the Number of machines column implies that the number of machines is arbitrary. In the Complexity results column, $\bar{p} = \max_{j \in N}\{p_j\}$. We also list the theorem, remark, or corollary where each complexity is found.

8. Discussion and further research

We have explored a relatively new class of scheduling problems. These problems are simpler than many batch scheduling problems because the composition of the batches is prespecifed. Also,

Table 1
The complexity of customer order scheduling problems

Objective	Number of machines	K	Variation	Complexity results	
$C_{B_{\max}}$	2	Yes		Unary NP-complete	Thm. 6.1
$\sum C_{B_i}$	2			Binary NP-complete	Rmrk. 4.1 or Thm. 4.1
$\sum C_{B_i}$	2		π	$O(\bar{p}^2 n \sum_{i\in B} P_i)$	Rmrk. 4.3
$\sum C_{B_i}$	2		J_F	Unary NP-complete	Thm. 4.2
$\sum C_{B_i}$	2		R	Unary NP-complete	Cor. 4.1
$\sum C_{B_i}$	2	Yes		Unary NP-complete	Thm. 6.2
$\sum C_{B_i}$				Unary NP-complete	Rmrk. 5.1
$\sum C_{B_i}$			$p_j = 1$	$O(b \log b)$	Thm. 5.2
$L_{B_{\max}}$			$p_j = 1$	$O(b \log b)$	Thm. 5.3
$\sum w_i C_{B_i}$			$p_j = 1$	Binary NP-complete	Thm. 5.4

the objective is concerned with batch completion times instead of job completion times. While the structure is simple, the problem has various real-world applications such as scheduling customer orders, scheduling the production of components for subsequent assembly into final products, crane scheduling at a port, and automotive repair shop scheduling. We hope that our results can be used to develop solution procedures for more complex and realistic applications.

From Theorem 4.1, $P2\|\sum C_{B_i}$ is at least binary NP-complete, but it is unknown whether it is unary NP-complete. Hence, one future challenge is either to develop a pseudo-polynomial time DP algorithm or to determine that the problem is unary NP-complete.

There are several extensions of our research that might be considered in the future. One is to include capacity restrictions in the model. Another extension is to study our problem with different machine speeds such as proportional and unrelated parallel machines. Also our problem can be studied with different shop environments such as job shop, open shop, and flow shop. These different shop environments have a variety of realistic applications. Further, our problem can be analyzed with different objectives such as $\sum w_i C_{B_i}$, $L_{B_{\max}}$, $\sum w_i U_{B_i}$, and $\sum w_i T_{B_i}$. Different real world applications require different objectives. For example, for the automotive repair shop problem, $\sum w_i T_{B_i}$ is a realistic objective.

Appendix

We establish Theorem 4.2 using a reduction from following unary NP-complete problem.

3-Partition (Garey and Johnson [20]). Given 3ℓ elements with integer sizes $a_1, a_2, \ldots, a_{3\ell}$, where $\sum_{i=1}^{3\ell} a_i = \ell y$ and $y/4 < a_i < y/2$ for $i = 1, 2, \ldots, 3\ell$, does there exist a partition $T_1, T_2, \ldots, T_\ell$ of the index set $\{1, 2, \ldots, 3\ell\}$ such that $|T_j| = 3$ and $\sum_{i\in T_j} a_i = y$ for $j = 1, 2, \ldots, \ell$?

We assume without loss of generality that, if there exists a solution to 3-Partition, then the elements are indexed such that $a_{3j-2} + a_{3j-1} + a_{3j} = y$ for $j = 1, 2, \ldots, \ell$.

Proof of Theorem 4.2. Given an instance of 3-Partition, we construct the following instance of $P2||\sum C_{B_i}$ with a fixed machine–job assignment:

$$n = 8\ell,$$

$$b = 5\ell,$$

$$n_i = 2, \quad i = 1,2,\ldots,3\ell,$$

$$n_i = 1, \quad i = 3\ell+1, 3\ell+2, \ldots 5\ell,$$

$$B_i = \{2i-1, 2i\}, \quad i = 1,2,\ldots,3\ell,$$

$$B_i = \{i+3\ell\}, \quad i = 3\ell+1, 3\ell+2, \ldots, 5\ell,$$

$$p_j = 5L + y/3 - a_{(j+1)/2}, \quad j = 1,3,\ldots,6\ell-1,$$

$$p_j = L - y/3 + a_{j/2}, \quad j = 2,4,\ldots,6\ell,$$

$$p_j = 6L, \quad j = 6\ell+1, 6\ell+2, \ldots, 8\ell,$$

$$J_1 = \{1,3,\ldots,6\ell-1\},$$

$$J_2 = \{2,4,\ldots,6\ell, 6\ell+1, 6\ell+2, \ldots, 8\ell\},$$

$$z = (75\ell^2 + 27\ell)L/2,$$

where $L = 2\ell^2 y$.

The recognition version of $P2||\sum C_{B_i}$ with a fixed machine–job assignment is in NP since given any schedule, we can calculate $\sum C_i$ in polynomial time.

We prove that there exists a solution to 3-Partition if and only if there exists a schedule for $P2||\sum C_{B_i}$ with a fixed machine–job assignment with $\sum C_i \leqslant z$.

($\Rightarrow$) Consider schedule $\sigma = (\sigma_1, \sigma_2)$ shown in Fig. 2 where

$$\sigma_1 = (1,3,\ldots,6\ell-1),$$

$$\sigma_2 = (2, 6\ell+1, 4, 6\ell+2, 6; 8, 6\ell+3, 10, 6\ell+4, 12; \ldots; 6\ell-4, 8\ell-1, 6\ell-2, 8\ell, 6\ell).$$

Since $a_{3j-2} + a_{3j-1} + a_{3j} = y$ for $j = 1,2,\ldots,\ell$ by assumption, the total completion time is

$$\sum_{i=1}^{5\ell} C_i = \sum_{j=1}^{\ell} (C_{3j-2} + C_{3\ell+2j-1} + C_{3j-1} + C_{3\ell+2j} + C_{3j})$$

$$= \sum_{j=1}^{\ell} \left[(j-1)\cdot 5 \cdot 15L + \left(5L + \frac{y}{3} - a_{3j-2}\right) + \left(7L - \frac{y}{3} + a_{3j-2}\right) \right.$$

$$\left. + \left(10L + \frac{2y}{3} - a_{3j-2} - a_{3j-1}\right) + \left(14L - \frac{2y}{3} + a_{3j-2} + a_{3j-1}\right) \right.$$

$$+ (15L + y - a_{3j-2} - a_{3j-1} - a_{3j})\Big]$$

$$= \sum_{j=1}^{\ell} [(j-1)75L + 51L]$$

$$= \frac{(75\ell^2 + 27\ell)L}{2}$$

$$= z.$$

Thus, $\sum_{i=1}^{5\ell} C_i \leqslant z$.

($\Leftarrow$) Let $I_1 = \{1, 2, \ldots, 3\ell\}$ and $I_2 = \{3\ell + 1, 3\ell + 2, \ldots, 5\ell\}$. Since all batches in I_2 are identical, we assume that in an optimal schedule, the batches are processed on machine 2 in index order.

Observe that only jobs $1, 3, \ldots, 6\ell - 1$ are processed on machine 1. In an optimal schedule σ^*, let the batches in I_1 be processed in the order $v_1, v_2, \ldots, v_{3\ell}$. For notational convenience, we let $v_1 = 1, v_2 = 2, \ldots, v_{3\ell} = 3\ell$. Then, 1, 3, and 5 are the first three jobs processed on machine 1 in σ^*. Also from Lemma 3.3, jobs 2, 4, and 6 are processed on machine 2 in the order 2, 4, and 6. First, we describe an optimal schedule. Then, we show that a schedule does not have a total batch completion time $\leqslant z$ unless it is optimal.

It is optimal to assign job 2 to machine 2 first because any schedule which starts with job $6\ell + 1$ can be improved by at least $L - y/3 + a_1$ if we switch the orders of jobs 2 and $6\ell + 1$.

Since job 4 can start as late as time $9L + y - a_1 - 2a_2$ without increasing its batch completion time, it is optimal to schedule job $6\ell + 1$ between jobs 2 and 4. Similarly, job 6 can start as late as time $14L + 4y/3 - a_1 - a_2 - 2a_3$ without increasing its batch completion time. Hence, it is optimal to schedule job $6\ell + 2$ between jobs 4 and 6. By using a similar argument, we can determine the optimal schedule for the remaining jobs on machine 2.

Observe that $C_i(\sigma_1^*) > C_i(\sigma_2^*)$ for batches $i = 3j - 2, 3j - 1$ where $j = 1, 2, \ldots, \ell$. Then, the total completion time is

$$\sum_{i=1}^{5\ell} C_i(\sigma^*) = \sum_{j=1}^{\ell} (C_{3j-2}(\sigma^*) + C_{3\ell+2j-1}(\sigma^*) + C_{3j-1}(\sigma^*) + C_{3\ell+2j}(\sigma^*) + C_{3j}(\sigma^*))$$

$$= \sum_{j=1}^{\ell} \Big[(j-1) \cdot 5 \cdot 15L + \left(5L + \sum_{k=1}^{j-1} (y - a_{3k-2} - a_{3k-1} - a_{3k}) + \frac{y}{3} - a_{3j-2}\right) + \left(7L + \sum_{k=1}^{j-1} (-y + a_{3k-2} + a_{3k-1} + a_{3k}) - \frac{y}{3} + a_{3j-2}\right)$$

$$
\begin{aligned}
&+\left(10L+\sum_{k=1}^{j-1}(y-a_{3k-2}-a_{3k-1}-a_{3k})+\frac{2y}{3}-a_{3j-2}-a_{3j-1}\right)\\
&+\left(14L+\sum_{k=1}^{j-1}(-y+a_{3k-2}+a_{3k-1}+a_{3k})-\frac{2y}{3}+a_{3j-2}+a_{3j-1}\right)\\
&+\ \max\{C_{3j}(\sigma_1^*),C_{3j}(\sigma_2^*)\}\Big]\\
&=12L+24L+15L+\max\{y-a_1-a_2-a_3,-y+a_1+a_2+a_3\}\\
&\quad+\sum_{j=2}^{\ell}\Bigg[(j-1)75L+36L+15L\\
&\quad+\ \max\left\{\sum_{k=1}^{j}(y-a_{3k-2}-a_{3k-1}-a_{3k}),\ \sum_{k=1}^{j}(-y+a_{3k-2}+a_{3k-1}+a_{3k})\right\}\Bigg]\\
&=\sum_{j=1}^{\ell}\Bigg[(j-1)75L+51L\\
&\quad+\ \max\left\{\sum_{k=1}^{j}(y-a_{3k-2}-a_{3k-1}-a_{3k}),\ \sum_{k=1}^{j}(-y+a_{3k-2}+a_{3k-1}+a_{3k})\right\}\Bigg]\\
&=(75\ell^2+27\ell)L/2\\
&\quad+\sum_{j=1}^{\ell}\max\left\{\sum_{k=1}^{j}(y-a_{3k-2}-a_{3k-1}-a_{3k}),\sum_{k=1}^{j}(-y+a_{3k-2}+a_{3k-1}+a_{3k})\right\}.
\end{aligned}
\tag{A.1}
$$

From (A.1), the total completion time only depends on $|y-a_{3j-2}-a_{3j-1}-a_{3j}|$ for $j=1,2,\ldots,\ell$. Furthermore, $\sum_{i=1}^{5\ell}C_i(\sigma)\leqslant z$ only if σ^* is optimal schedule and if $y-a_{3j-2}-a_{3j-1}-a_{3j}=0$ for $j=1,2,\ldots,\ell$. This is only possible if there exists a solution to 3-Partition. □

References

[1] Julien FM, Magazine MJ. Scheduling customer orders: an alternative production scheduling approach. Journal of Manufacturing and Operations Management 1990;3:177–99.

[2] Jordan C. Batching and scheduling: models and methods for several problem classes. New York: Springer; 1996.

[3] Potts CN, Van Wassenhove LN. Integrating scheduling with batching and lot-sizing: a review of algorithm and complexity. Journal of Operational Research Society 1992;43:395–406.

[4] Webster C, Baker KR. Scheduling groups of jobs on a single machine. Operations Research 1995;43:692–703.

[5] Coffman EG, Nozari A, Yannakakis M. Optimal scheduling of products with two subassemblies on a single machine. Operations Research 1989;37:426–36.

[6] Baker KR. Scheduling the production of components at a common facility. IIE Transactions 1988;20:32–5.
[7] Santos C, Magazine M. Batching in single operation manufacturing systems. Operations Research Letters 1985;4: 99–103.
[8] Gupta JND, Ho JC, van der Veen AA. Single machine hierarchical scheduling with customer orders and multiple job classes. Annals of Operations Research 1997;70:127–43.
[9] Gerodimos AE, Glass CA, Potts CN. Scheduling the production of two-component jobs on a single machine. European Journal of Operational Research 2000;120:250–9.
[10] Daganzo CF. The Crane scheduling problem. Transportation Research Part B 1989;23B:159–75.
[11] Peterkofsky RI, Daganzo CF. A branch and bound solution method for the crane scheduling problem. Transportation Research Part B 1990;24B:159–72.
[12] Blocher JD, Chhajed D. The customer order lead time problem on parallel machines. Naval Research Logistics 1996;43:629–54.
[13] Blocher JD, Chhajed D, Leung M. Customer order scheduling in a general job shop environment. Decision Sciences 1998;29:951–81.
[14] Yang J. Scheduling parallel machines for the customer order problem with fixed batch sequence. Journal of the Korean Institute of Industrial Engineers 2003;29:304–311.
[15] Roemer TA, Ahmadi R. Complexity of scheduling customer orders. Working paper, Anderson School at UCLA, USA, 1997.
[16] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic machine scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[17] Rinnooy Kan AHG. Machine scheduling problems: classification, complexity and computations. The Hague: Nijhoff; 1976.
[18] Yang J, Posner ME. Scheduling parallel machines for the customer order problem. Working paper, Ohio State University, Columbus, OH, USA.
[19] Smith WE. Various optimizers for single-stage production. Naval Research Logistics Quarterly 1956;3:59–66.
[20] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: W.H. Freeman & Co.; 1979.

Jaehwan Yang is an assistant professor in the department of business administration, University of Incheon, Incheon, Korea. He earned his M.S. and Ph.D. degree in industrial and systems engineering from Ohio State University, USA, and his major academic interests include scheduling and optimization.