International Journal of Management Science
Vol 11, No 2, November 2005

Customer Order Scheduling Problems with a Fixed Machine-Job Assignment*

Jaehwan Yang**
School of Business Administration, University of Seoul, Seoul, Korea
90 Jeonnong-dong, Dongdaemun-gu, Seoul, 130-743, Korea

Yoomi Rho***
Department of Mathematics, University of Incheon, Incheon, Korea
177 Dohwa-dong, Namgu, Incheon, 402-749, Korea

(Received Mar. 2005 ; Revised Jun. 2005 ; Accepted Jun. 2005)

ABSTRACT

This paper considers a variation of the customer order scheduling problem, and the variation is the case where the machine-job assignment is fixed. We examine the parallel machine environment, and the objective is to minimize the sum of the completion times of the batches. While a machine can process only one job at a time, different machines can simultaneously process different jobs in a batch. The recognition version of this problem is known to be NP-complete in the strong sense even if there exist only two parallel machines.
When there are an arbitrary number of parallel machines, we establish three lower bounds and develop a dynamic programming (DP) algorithm which runs in exponential time on the number of batches. We present two simple but intuitive heuristics, SB and GR, and find some special cases where SB and GR generate an optimal schedule. We also find worst case upper bounds on the relative error. For the case of the two parallel machines, we show that GR generates an optimal schedule when processing times of all batches are equal. Finally, the heuristics and the lower bounds are empirically evaluated.

Keywords: Scheduling, Batch Scheduling, Customer Order Scheduling, Dynamic Programming, Heuristics

* This work was supported by New Professor Research Fund of 2004 from University of Incheon. The grant was awarded to Dr. Jaehwan Yang during his service at University of Incheon.

** Corresponding author, Email: jyang@uos.ac.kr

*** Email: rho@incheon.ac.kr

1. INTRODUCTION

In the typical customer order scheduling problem, jobs are dispatched in the batches and the composition of the jobs in the batch is prespecified. There exist no setup times between different jobs or different batches, and the objective is associated with the completion time of batches instead of the completion time of each job where the completion time of the batch is the latest completion time of any job in the batch. While a machine can process only one job at a time, multiple machines can simultaneously process jobs in the batch. This paper considers a variation of this customer order scheduling problem, and the variation is the case where *the machine-job assignment is fixed.* Hence, jobs are processed by preassigned machines. This is common in situations where machines are capable of doing a limited set of tasks. For this paper, we examine the parallel machine environment with two machines and an arbitrary number of machines.

This restricted version of customer order scheduling problem is introduced by Roemer and Ahmadi [14] and Ahmadi *et al.* [1] although their objective is to minimize the weighted sum of the batch completion times. Their research is motivated by a manufacturer who produces three types of semi-finished lenses. Each customer order consists of different quantities of the three types of lenses, and a different type of lenses must be processed by a different machine. The customer order can not be shipped to the customer unless the entire order is completed.

This can be extended to a more general manufacturing example. Consider a manufacturing facility which produces different types of products. A customer can request a variety of products in an order. After the entire order is produced, the products are shipped to the customer. Each order is a batch and a product is a job. The composition of the batch is specified by the order. Also, different products require different processing and are therefore processed by different machines.

The customer order scheduling problem is different from most of other batch scheduling problems because the objective is associated with the completion time of the batches instead of the completion time of each job and there exist no setup times between different jobs or different batches. For a comprehensive survey of general batch scheduling problems, see Jordan [10]. Also, reviews of batching and lot-sizing decision problems are given by Potts and Van Wassenhove [13] and Webster and Baker [16].

Julien and Magazine [11] is probably the first to consider the problem where the objective is associated with the completion time of batches. They study a single machine problem where the objective is to minimize the total completion time of the batches. A job-dependent setup time is incurred between two different types of jobs. They develop a polynomial time DP algorithm for the problem when there are two types of jobs and when the batch processing order is fixed. Coffman *et al.* [5] examine a similar problem where the batch processing order is not fixed. They develop an $O(n^{1/2})$ procedure. Baker [2] considers a problem similar to Coffman *et al.* [5]. However, for one type of job, those jobs processed during the same production run (setup) are not available until the completion of the production run. This restriction is called *batch availability* (see Santos and Magazine [15]). Gupta *et al.* [9] consider the single machine problem where each order must have one job from each of several job classes. Also, there is a setup time whenever the job class changes. Gerodimos *et al.* [7] study single machine problems where each batch has one common job and one distinct job. Ding [6], Liao [12], and Yoon [20] also examine similar problems.

In the batch scheduling problems that we discuss, there exist no setup times between different jobs or different batches. Blocher and Chhajed [3] examine the customer order scheduling problem, minimizing the sum of batch completion times in the parallel machine environment with no variation. They show that the recognition version of the problem is NP-complete in the strong sense when there exist at least three parallel machines and the same version is at least NP-complete in the ordinary sense when there exist two parallel machines. Also, they develop several heuristic methods and two lower bounds. Yang and Posner [17] consider the same problem with two parallel machines, and develop three simple heuristics and find the tight worst case bounds on relative errors of 2, 9/7, and 6/5, respectively. Yang [18] considers a variation of customer order scheduling problem where the batch sequence is fixed. He considers the two parallel machine case establishes that the recognition version of the problem is NP-complete in the ordinary sense, and develops a DP algorithm which runs in pseudo-polynomial time on the number of batches. Yang [19] establishes the complexity of different customer order scheduling problems and summarizes the complexity results. He summarizes the complexity of 10 different cases (or variations) of customer order scheduling problems and establishes the complexity of six different cases.

When the machine-job assignment is fixed, Roemer and Ahmadi [14] show that the recognition version of the problem is NP-complete in the strong sense for the case where there exist only two parallel machines with the objective of minimizing the sum of batch completion times. An easier complexity proof is presented in Yang [19]. Ahmadi *et al.* [1] develop three lower bounds and several heuristics for the problem with the objective of minimizing the sum of weighted batch completion times.

We first introduce notation. Next, we review some preliminary results including complexity of the problem. For the problem with an arbitrary number of parallel machines, three lower bounds are established and a DP algorithm is developed. Since the recognition version of the problem is NP-complete in the strong sense even if there exist only two parallel machines, two simple but intuitive heuristics, SB and GR are developed. We find worst case upper bounds on the relative error and establish several special cases. For the case of two parallel machines, we show that heuristic GR finds an optimal schedule when all batches have the equal processing time. Finally, we present the result of a computational study.

2. NOTATION

The decision variables in our models are

σ_k	= schedule of all jobs on machine k for $k \in M$
σ	= schedule of all jobs $= (\sigma_1, \sigma_2, \cdots, \sigma_m)$.

Other notation that is used in this work include

n	= number of jobs
N	= set of jobs $= \{1, 2, \cdots, n\}$
b	= number of batches
B	= set of batches $= \{1, 2, \cdots, b\}$
n_i	= number of jobs in batch i for $i \in B$
M	= set of machines = $\{1, 2, \cdots, m\}$
B_i	= set of jobs in batch i for $i \in B = \{\Sigma_{j=1}^{i-1} n_j + 1, \Sigma_{j=1}^{i-1} n_j + 2, \cdots, \Sigma_{j=1}^{i} n_j\}$
B_i^k	= set of jobs assigned to machine k for $k \in M$ in batch $i \in B$
m	= number of machines
p_j	= processing time of job j for $j \in N$
P_i	$= \Sigma_{j \in B_i} p_j$ = total processing time of batch $i \in B$
P_i^k	$= \Sigma_{j \in B_i^k} p_j$ = total processing time of jobs assigned to machine k for $k \in M$ in batch $i \in B$
$C_i(\sigma_k)$	= 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 $i \in B = \max_{k \in M} C_i(\sigma_k)$
z^*	= value of optimal schedule.

We represent $C_i(\sigma)$ as C_i when there is no ambiguity. The standard classification scheme for scheduling problems (Graham *et al.* [8]) is $\alpha_1 \mid \alpha_2 \mid \alpha_3$, where α_1 describes the machine structure, α_2 gives the job characteristics or restrictive requirements, and α_3 defines the objective function to be minimized. We extend this scheme to provide for batch completion times by using C_{B_i} in the α_3 field. This notation is used to eliminate the confusion between our problem and the classical scheduling problem. For instance, the problem of minimizing the sum of batch completion times on arbitrary number of parallel machines is written as $P \mid \mid \Sigma C_{B_i}$. A number after P indicates a fixed number of machines rather than an arbitrary number, m.

3. LOWER BOUNDS

In this section, we provide two lower bounds for the optimal solution value. Assume that the batches are indexed so that $P_1 \le P_2 \le \cdots \le P_b$.

Remark 1. *For problem* $P \mid \mid \Sigma C_{B_i}$,

$$\frac{bP_1}{m} + \frac{(b-1)P_2}{m} + \cdots + \frac{P_b}{m} \le z^*. \tag{1}$$

Observe that the left side of the inequality (1) is the optimal solution value to the Linear Programming (LP) relaxation of problem $P \mid \mid \Sigma C_{B_i}$ where a job can be split into pieces of any size and processed, simultaneously if desired, on multiple machines without considering machine-job assignment.

We call this lower bound L1 and the solution value z^{L1}, respectively. Blocher and Chhajed [3] show that L1 is a lower bound for $P \mid \mid \Sigma C_{B_i}$. Hence, L1 is a lower bound for $P \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.

Next, we consider another lower bound L2. For each machine, lower bound L2 reindexes batches in increasing order of processing times. Then, it creates a new set of batches by putting jobs with the same index into one batch. We formally describe lower bound L2.

Lower Bound L2.

0. Reindex the batches so that $P_i \le P_{i+1}$ for $i = 1, 2, \cdots, b-1$.

   If $B_i^k \ne \phi$ for $i \in B$ and $k \in M$, then set $\hat{B}_i^k = B_i^k$. Otherwise, set $\hat{B}_i^k = \phi$.

   Set $\hat{P}_i^k$ = total processing time of jobs in $\hat{B}_i^k$ for $i \in B$ and $k \in M$.
1. Reindex $\hat{B}_i^k$ so that for each $k \in M$, $\hat{P}_i^k \le \hat{P}_{i+1}^k$ for $i = 1, 2, \cdots, b-1$.
2. Create a new set of batches $\hat{B}_i$ such that $\hat{B}_i = \hat{B}_i^1 \cup \hat{B}_i^2 \cup \cdots \cup \hat{B}_i^m$ for $i \in B$.
3. Schedule batches $\hat{B}_i$ for $i \in B$ in index order.
4. Output total completion time.

Steps 0 and 1 require $O(b\log b + mb)$ time. In Step 2, reindexing $\hat{B}_i^k$'s requires $O(mb\log b)$. Since all other operations require $O(mb + n)$, the time requirement of L2 is $O(mb\log b + n)$.

The next remark establishes that z^{L2} is a lower bound for the problem.

Theorem 1. *For $P \,|\,|\, \Sigma C_{B_i}$ with a fixed machine-job assignment, z^{L2} is a lower bound of an optimal solution value.*

Proof. Let σ^* and σ^{L2} be an optimal schedule and the schedule created by L2, respectively. Reindex batches so that schedule σ^{L2} is completed in index order. Note that for each $i \in B$, $C_i(\sigma^*) = \max_{k \in M}\{C_i(\sigma_k^*)\}$. Let $C_{[i]}(\sigma_k^*)$ be completion time of ith completed batch in σ^* for $i \in B$ and $k \in M$. Note that in Step 2 of lower bound L2, $\hat{B}_i^k$ is reindexed so that for each $k \in M$, $\hat{P}_i^k \le \hat{P}_{i+1}^k$ for $i = 1, 2, \cdots, b-1$. Hence, $C_{[i]}(\sigma_k^*) \ge C_i(\sigma_k^{L2}) = \hat{P}_1^k + \hat{P}_2^k + \cdots + \hat{P}_i^k$ where $\hat{P}_1^k \le \hat{P}_2^k \le \cdots \le \hat{P}_i^k$ for $i \in B$ and $k \in M$. Therefore,

$$z^* = \sum_{i=1}^{b} C_{[i]}(\sigma^*) = \sum_{i=1}^{b} \max_{k \in M}\{C_{[i]}(\sigma_k^*)\} \ge \sum_{i=1}^{b} \max_{k \in M}\{C_i(\sigma_k^{L2})\} = \sum_{i=1}^{b} C_i(\sigma^{L2}) = z^{L2}.$$

Finally, we consider the third lower bound L3. Let σ^{L2} and σ^{L3} be schedules generated by L2 and L3, respectively. Also, let $C_{[i]}(\sigma_k^{L2})$ and $C_{[i]}(\sigma_k^{L3})$ be the completion times of the ith completed batch for schedules σ^{L2} and σ^{L3}, respectively. Suppose that the batches are indexed so that $P_1 \le P_2 \le \cdots \le P_b$. Then, the completion time of the ith completed batch for L3 is $\max\{(P_1 + P_2 + \cdots +$

$P_i)/m, C_{[i]}(\sigma^{L2})\}$. In other words, L3 takes the maximum of the completion times of i th completed batch from L1 and L2.

Note that L3 dominates L1 and L2. We now formally describe lower bound L3.

Lower Bound L3.

0. Reindex the batches so that $P_i \le P_{i+1}$ for $i = 1, 2, \cdots, b-1$.
   Set $C_{[i]}(\sigma^{L2}) =$ completion time of the i th completed batch from L2 for $i = 1, 2, \cdots, b$.
1. Set $C_{[i]}(\sigma^{L3}) = \max\{(P_1 + P_2 + \cdots + P_i)/m, C_{[i]}(\sigma^{L2})\}$ for $i = 1, 2, \cdots, b$.
2. Output total completion time.

Recall that time requirement of L2 is $O(mb\log b + n)$. Step 0 requires $O(b\log b)$ time. Step 1 requires $O(mb\log b + n)$, and Step 2 requires $O(b)$ time. Therefore, time requirement of L3 is $O(mb\log b + n)$.

The next theorem establishes that z^{L3} is a lower bound for the problem.

Theorem 2. *For $P \,||\, \Sigma C_{B_i}$ with a fixed machine-job assignment, z^{L3} is a lower bound of an optimal solution value.*

Proof. Let σ^*, σ^{L2}, and σ^{L3} be an optimal schedule, a schedule created by L2, and a schedule created by L3, respectively. For any schedule σ, let $C_{[i]}(\sigma)$ be the completion time of i th completed batch in σ for $i \in B$. Also, suppose that the batches are indexed so that $P_1 \le P_2 \le \cdots \le P_b$.

From the proof of Theorem 1, $C_{[i]}(\sigma^*) \ge C_{[i]}(\sigma_k^{L2})$ for all $i \in B$. Also, $C_{[i]}(\sigma^*) \ge (P_1 + P_2 + \cdots + P_i)/m$. Since $C_{[i]}(\sigma^{L3}) = \max\{(P_1 + P_2 + \cdots + P_i)/m, C_{[i]}(\sigma^{L2})\}$ for all $i \in B$, $C_{[i]}(\sigma^*) \ge C_{[i]}(\sigma_k^{L3})$ for all $i \in B$. Therefore, $z^* = \sum_{i \in B} C_{[i]}(\sigma^*) \ge \sum_{i \in B} C_{[i]}(\sigma^{L3}) = z^{L3}$ □

4. PRELIMINARY RESULTS

In this section, we review some preliminary results for our problem.

4.1 General Results

We begin with the following lemma.

Lemma 1. (Yang [19]) *For scheduling problems with regular measures, there exists an optimal schedule without inserted idle time.*

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 2. (Yang [19]) *For scheduling problems with regular measures, there exists an optimal schedule where no batch is separated.*

As a result of Lemma 2, we assume batches are not separated in an optimal schedule. We now present another property of batch scheduling problems.

Lemma 3. (Blocher and Chhajed [3]) *For batch scheduling problems with a regular measure, each machine processes the batches which are processed on multiple machines in the same order.*

4.2 Complexity

Since each job is pre-assigned to a machine, we only need to determine the sequence of jobs on each machine. Recall that Lemma 3 implies that we only consider schedules where batches that are 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 [14]. An easier proof is presented in Yang [19].

Theorem 3. *The recognition version of $P \mid \mid \sum C_{B_i}$ with a fixed machine-job assignment is NP-complete in the strong sense.*

5. DP ALGORITHM

Now, we present a DP algorithm for $P \mid \mid \sum C_{B_i}$ with fixed machine-job assignment. Since there are b batches to sequence, enumeration requires $O(b!)$ time to obtain an optimal schedule. From *Stirling's approximation* formula, $b! = \sqrt{2\pi b}$ $(b/2)^b(1+\Theta(1/b))$. We present a DP algorithm which finds an optimal schedule for $P \mid \mid \Sigma C_{B_i}$ with fixed machine-job assignment in $O(mb2^b)$ time. Recall that P_i^k is total processing time of batch i on machine k for $i \in B$ and $k \in M$.

Let $f(V)$ be minimum total completion time of the batches in set V for $V \subseteq B$. Set V is the first $|V|$ batches in a schedule generated by set B. Observe that given $V \subseteq B$, the completion time of the last batch is the same regardless of batch processing order because total processing time on each machine is fixed. Hence, to obtain an optimal schedule, we only examine $|V|$ different cases where the different batches in V complete last overall. As an initial condition, we let $f(\phi) = 0$. Define $f(V)$ recursively as follows:

$$f(V) = \min_{i \in V} \left\{ f(V \setminus \{i\}) + \max\{\sum_{\ell \in V} P_\ell^1, \sum_{\ell \in V} P_\ell^2, \ldots, \sum_{\ell \in V} P_\ell^m\} \right\} \text{ for } |V| \geq 1. \tag{2}$$

Observe that $\max\{\sum_{\ell \in V} P_\ell^1, \sum_{\ell \in V} P_\ell^2, \ldots, \sum_{\ell \in V} P_\ell^m\}$ is the completion time of the last batch. In order to obtain $f(B)$ the minimal total completion time, we calculate $f(V)$ for all $V \subset B$ and $1 \leq |V| \leq b-1$. To record the optimal choice at each iteration, let $\upsilon(V)$ represent the last batch in a schedule which corresponds to $f(V)$. Then,

$$\upsilon(V) = \arg\min_{i = V} \left\{ f(V \setminus \{i\}) + \max\{\sum_{\ell \in V} P_\ell^1, \sum_{\ell \in V} P_\ell^2, \cdots, \sum_{\ell \in V} P_\ell^m\} \right\} \text{ for } |V| \geq 1. \tag{3}$$

We now formally describe a DP procedure that finds an optimal schedule for $P \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.

Algorithm A1.

```
0. Set $\sigma^* = \phi$, $i = 1$, and $B = \{1, 2, \cdots, b\}$
1. For all $V$ such that $V \subset B$ and $|V| = i$, find $f(V)$ and $\upsilon(V)$ using (2) and
   (3). Break ties arbitrarily in equation (3).
2. If $i = b$, then go to Step 3.
   Otherwise, set $i = i + 1$ and go to Step 1.
3. Use $\upsilon$ to calculate the optimal schedule $\sigma^*$.
   Output $\sigma^*$ and $f(B)$.
```

The following theorem verifies the optimality of Algorithm A1.

Theorem 4. *Algorithm A1 produces an optimal job-machine assignment to* $P \mid \mid \Sigma C_{B_i}$ *with a fixed machine-job assignment in* $O(mb2^b)$ *time.*

Proof. Breaking ties arbitrarily in Step 1 does not change the optimal solution value because given $V \subset B$, the last batch completes at $\max\{\sum_{\ell \in V} P_\ell^1, \sum_{\ell \in V} P_\ell^2$ $\cdots, \sum_{\ell \in V} P_\ell^m\}$ regardless of batch processing orders. Hence, the choice of $v(V)$ does not affect the value of $f(V^o)$ for $V \subset V^o \subset B$ and $|V^0| = |V| + 1$.
In A1, consider the determination of $f(V)$ in (2) for any set $V \subset B$. For all $V' \subset V$ such that $|V'| = |V| - 1$, $f(V')$ is known. Recall that for any $V \subset B$, the completion time of the last batch is the same regardless of batch processing order. Hence, in Step 1, only $|V|$ different cases are considered. For these $|V|$ different cases, $|V|$ different batches in V complete last. From the definition of $f(V)$ and $f(V')$ for $V' \subset V$ and $|V'| = |V| - 1$, $f(V)$ is the minimal total completion time for the batches in V. Thus, $f(B)$ is the optimal value. Also, σ^* is an optimal schedule because $\sum_{i=1}^b C_i(\sigma^*) = f(B)$.
Step 0 requires $O(b)$ time. Step 1 considers $\binom{b}{i}$ different sets of batches for $i = 1, 2, \cdots, b$. Also, for each set, i different candidates are compared. Thus, Step 1 requires $O(mi \cdot \binom{b}{i})$ time. Step 1 repeats for $i = 1, 2, \cdots, b$. Steps 3, 4, 5, and 6 require constant time, and they repeat b times. Step 7 requires constant time. Now,

$$\sum_{i=1}^{b}\left(i \cdot \binom{b}{i}\right) = 1 \cdot \binom{b}{1} + 2 \cdot \binom{b}{2} + \cdots + b \cdot \binom{b}{b}$$

$$= \frac{1+(b-1)}{2}\binom{b}{1} + \frac{2+(b-2)}{2}\binom{b}{2} + \cdots + \frac{(b-1)+1}{2}\binom{b}{b-1} + b\binom{b}{b}$$

$$= b\left[\frac{\left(\sum_{i=1}^{b}\binom{b}{i}\right) - 1}{2}\right] + b = b\left[\frac{(2^b - 1) - 1}{2}\right] + b$$

$$= b2^{b-1}.$$

Therefore, A1 finds a solution in $O(\sum_{i=1}^{b} m[i\cdot\binom{b}{i}]) = O(mb2^b)$ time. Because the number of elementary operations and size of all values are bounded by a exponential function of the input length, A1 runs in exponential time.

We now illustrate the algorithm with an example.

Example. Consider the instance where $b=3$, $n_1=2$, $n_2=2$, $n_3=1$, $p_1=p_2=1$, $p_3=1$, $p_4=2$, and $p_5=3$. Also, $B_1^1=\{1\}$, $B_1^2=\{2\}$, $B_2^1=\{3\}$, $B_2^2=\{4\}$, and $B_3^1=\{5\}$. For an initial condition, $f(\phi)=0$ (See Table 1). Since there exist three batches, three different states are considered at Stage 1. At Stage 2, we also considered three states because $\binom{3}{2}=3$. At the final stage, A1 produces an optimal solution value. The optimal schedule is $\sigma^*=((1,3,5),(2,4))$, and the optimal solution value is $z^*=1+3+5=9$.

Table 1. An example for A1

Stage (i)	*State* (V)	$f(V)$	$v(V)$	σ
1	{1}	1	1	((1),(2))
	{2}	2	2	((3),(4))
	{3}	3	3	((5), ϕ)
2	{1,2}	4	2	((1,3),(2,4))
	{1,3}	5	3	((1,5),(2))
	{2,3}	6	3	((3,5),(4))
3	{1,2,3}	9	3	((1,3,5),(2,4))

6. A HEURISTIC

In this section, we present a simple heuristic. The heuristic uses the Shortest Batch rule (SB, when a machine becomes available, an unscheduled job in the batch with a shortest total processing time is selected for processing) to find the batch sequence. Even though the heuristic is simple, it is intuitive and has practical implications.

6.1 Description

We formally describe the heuristic.

Heuristic SB.

0. Reindex batches so that $P_i \le P_{i+1}$ for $i = 1, 2, \cdots, b-1$.
1. Schedule all jobs in batches in index order. Assign jobs to machines according to their fixed machine-job assignment. When there exist ties, break them arbitrarily.
2. Output $\sum_{i=1}^{b} C_i$ and stop.

In Step 0, reindexing the batches requires $O(b \log b)$ time. Since all other operations require $O(n)$ time, the time requirement of SB is $O(b \log b + n)$.

6.2 Special Cases

We examine some special cases where heuristic SB generates an optimal schedule

Theorem 5. *If $n_i = 1$ for all $i \in B$ or $b = 1$, then heuristic SB generate an optimal schedule for problem $P \mid \mid \Sigma C_{B_i}$ with fixed machine-job assignment*

Proof. If $b = 1$, then any rule is optimal because switching jobs in the batch does not change the solution value. If $n_i = 1$, then each batch is processed by one machine. Since machine-job assignment is fixed, scheduling each machine can be performed separately. Then, the solution value for each machine can be summed to calculate C_i for the entire problem. Since each batch contains only one job, for each machine, the problem reduces to $1 \mid \mid \Sigma C_j$. Consequently, heuristic SB generates an optimal schedule.

The next theorem establishes a special case where processing times on each machine is *semi-ordered* such that $P_{i-1}^k \le P_i^k$ for $i = 2,3,\cdots,b;\quad k = 1,2,\cdots,m$.

Theorem 6. *If $P_{i-1}^k \le P_i^k$ for $i = 2, 3, \cdots, b$; $k = 1, 2, \cdots, m$, then heuristic SB generates an optimal schedule for $P \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.*

Proof. Let σ^H and σ^* be a schedule by heuristic SB and an optimal schedule, respectively. Notice that $P_1 \le P_2 \le \cdots \le P_b$ and thus, the batch sequence generated by SB is $(1,2,\cdots,b)$. Also, note that SPT rule (shortest processing time job first) generates an optimal schedule for problem $1 \mid \mid \sum C_j$. Since $P_{i-1}^k \le P_i^k$ for $i = 2, 3, \cdots, b$; $k = 1, 2, \cdots, m$, $C_i(\sigma^H) = \max_{k \in M}\{\sum_{j=1}^{i} P_j^k\}$ for all $i \in B$. Let $C_{[i]}(\sigma_k^*)$ be the completion time of ith completed batch in σ^* for $i \in B$ and $k \in M$. Then, $C_{[i]}(\sigma_k^*) \ge \sum_{j=1}^{i} P_j^k$ because $P_1^k \le P_2^k \le \cdots \le P_b^k$ and $C_{[i]}(\sigma^*) \ge \max_{k \in M}\{C_{[i]}(\sigma_k^*)\}$. Therefore, $C_{[i]}(\sigma^*) \ge C_i(\sigma^H)$, and we have the result.

The following theorem establishes a special case for the two parallel machines.

Theorem 7. *If $P_i^1 \ge P_i^2$ for all $i \in B$ (or $P_i^1 < P_i^2$ for all $i \in B$), then heuristic SB generates an optimal schedule for $P2 \mid \mid \sum C_{B_i}$ with a fixed machine-job assignment.*

Proof. Suppose that $P_i^1 \ge P_i^2$ for all $i \in B$. Let σ^H be a schedule by heuristic SB. Note that SPT rule generates an optimal schedule for problem $1 \mid \mid \sum C_j$. Hence, this is the minimum possible completion time for ith scheduled batch on machine 1 in any schedule. Since $C_i(\sigma^H) = \max\{C_i(\sigma_1^H), C_i(\sigma_2^H)\}$, heuristic SB generates an optimal schedule.

The result of Theorem 7 can be extended to the case of an arbitrary number of parallel machines. However, another rule is optimal for the problem.

Corollary 1. *If there exists $k \in M$ such that $P_i^k \ge \max_{\ell \in M}\{P_i^\ell\}$ for all $i \in B$, then shortest processing time order on machine k generates an optimal schedule for $P \mid \mid \sum C_{B_i}$ with a fixed machine-job assignment.*

Proof. Let σ^H be a schedule by heuristic SB. Since $P_i^k \ge \max_{\ell \in M}\{P_i^\ell\}$, $C_i(\sigma^H) = \sum_{j=1}^{i} P_j^k$ for all $i \in B$. Note that SPT rule generates an optimal sequence for problem $1 \mid \mid \sum C_j$. Hence, this is the minimum possible completion time for ith scheduled batch on machine k in any schedule. Since $C_i(\sigma^H) = \max_{k \in M}\{C_i(\sigma_k^H)\}$, heuristic SB generates an optimal schedule.

6.3 An Upper Bound on the Relative Error

Let σ^H be a schedule by heuristic SB and let z^{SB} be solution value of an SB schedule. We assume that $P_i \le P_{i+1}$ for $i = 1, 2, \cdots, b-1$. The following theorem establishes a worst case bound on the relative error for arbitrary number of machines.

Theorem 8. For $P \,|\,|\, \Sigma C_{B_i}$ with a fixed machine-job assignment, $z^{SB} / z^* < m$.

Proof. Note that $z^{SB} \le bP_1 + (b-1)P_2 + \cdots + 2P_{b-1} + P_b$. The equality holds only when all batches are processed on one machine. From Remark 1, $z^* \ge \{bP_1 + (b-1)P_2 + \cdots + 2P_{b-1} + P_b\} / m$. The equality holds only when for each batch, processing time is evenly distributed on machines 1 through m. However, the two inequalities can not occur for the same set of batches. Hence, $z^{SB} / z^* < m$.

7. A POLYNOMIAL TIME PROCEDURE FOR PROBLEM $P2 \,|\, P_i = 1 \,|\, \Sigma C_{B_i}$

In this section, we present a greedy type algorithm which finds an optimal schedule for the case where there exist two parallel machines and $P_i = \bar{P}$ for a constant $\bar{P}$ and all $i \in B$. The problem is formally written as $P2 \,|\, P_i = 1 \,|\, \Sigma C_{B_i}$ with a fixed machine-job assignment.

The algorithm uses a greedy rule to determine a batch sequence. Given set of batches $V = B$, a batch with the smallest makespan is scheduled first. Then, this batch is removed from V. For each remaining batch in V, the algorithm schedules the batch at the end of a current partial schedule and finds a batch which generates the smallest solution value. Then, this batch is also removed from V and it is scheduled to the end of the current partial schedule. The algorithm repeats until no more batches are left in V. Now, we formally describe the algorithm, which can be applied to the case of an arbitrary number of parallel machines.

Algorithm GR.

```
0. Set $F_i^k = 0$ for $i = 0, 1, 2, \cdots, b$; $k = 1, 2, \cdots, m$.
   Set $V = B$ and $i = \ell = 1$.
1. Set $\ell = \arg\min_{q \in V} \{F_{i-1}^1 + P_q^1, F_{i-1}^2 + P_q^2, \cdots, F_{i-1}^m + P_q^m\}$.
   Set $V = V \setminus \{\ell\}$.
   Set $F_i^k = F_{i-1}^k + P_\ell^k$ for $k = 1, 2, \cdots, m$.
2. Set $C_i = \max_{k \in M} \{F_i^k\}$.
   If $i < b$, then set $i = i + 1$, and go to Step 1.
   Otherwise, output $\sum_{i=1}^{b} C_i$ and stop.
```

Note that GR is also a heuristic for $P \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment. We evaluate the performance of GR along with SB in the next section.

The following remark finds special cases where algorithm GR generates an optimal schedule for $P \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.

Theorem 9. *Theorems 5, 6, 7, and 8, and Corollary 1 also hold for algorithm GR.*

Proof. The proof is similar to those of Theorems 5, 6, 7, and 8, and Corollary 1.

Let σ^H be a schedule by algorithm GR. We assume that batches are reindexed according to the selection order in Step 1. The following lemma establishes a property of a schedule generated by algorithm GR for $P2 \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.

Lemma 4. For $P2 \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment, $C_i(\sigma^H) \geq C_{i-1}(\sigma^H)$ for $i = 2, 3, \cdots, b$.

Proof. Suppose that there exists a batch $i \in B$ such that $C_i(\sigma^H) < C_{i-1}(\sigma^H)$. From Lemma 3, if $P_i^1 > 0$ and $P_i^2 > 0$, then $C_i(\sigma^H) > C_{i-1}(\sigma^H)$. Hence, $P_i^1 = 0$ or $P_i^2 = 0$. Without loss of generality, we assume that $P_i^1 = 0$. Then, $C_{i-1}(\sigma_2^H) > C_i(\sigma_1^H)$. However, Step 1 of GR must select batch i first before batch $i-1$ because choosing batch i first generates a smaller solution value, a contradiction. Therefore, $C_i(\sigma^H) \geq C_{i-1}(\sigma^H)$ for $i = 2, 3, \cdots, b$.

As a result of Lemma 4, we assume that the selection order in Step 1 is the same as the completion order in σ^H which is generated by GR.

Next, we define some additional notation used in this section (see Yang and Posner [17]). For $k, i \in B$, suppose k is the last batch to complete before i in schedule σ. Let

$$\delta_i(\sigma) = \begin{cases} C_i(\sigma) - C_k(\sigma) & \text{if only one machine processes batch } i \\ |C_i(\sigma_1) - C_i(\sigma_2)| & \text{if both machines process batch } i. \end{cases}$$

For each $i \in B$, $\delta_i(\sigma)$ is the absolute difference between completion time of batch i on machine 1 and machine 2 in σ. If batch i is processed on only one machine, then $\delta_i(\sigma)$ is the difference between the completion time of batch i and the completion time of the last batch to complete before i (see Figure 1). If batch i is the first batch to complete, then we assume $C_k(\sigma) = 0$. We use $\delta_i(\sigma)$ to provide a description of the completion time of batch i. If $G \subseteq B$ is the set of batches that complete no later than batch i, then

$$C_i(\sigma) = \frac{\sum_{\ell \in G} P_\ell + \delta_i(\sigma)}{2}.$$

Notice that since P_ℓ is known for $\ell = 1, 2, \cdots, i$, $C_i(\sigma)$ only depends on the size of $\delta_i(\sigma)$.

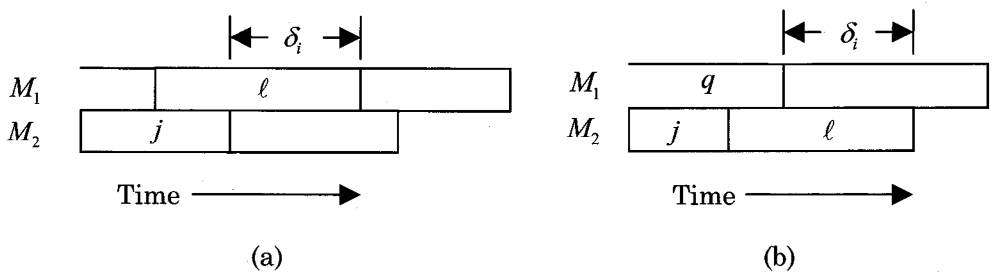

Figure 1. Examples of δ_i where the last two jobs processed in batch i are j and ℓ : (a) batch i is processed by both machines; (b) batch i is processed only by M_2 and q is the last job to complete in batch k.

The next theorem establishes that GR finds an optimal schedule for $P2 | P_i = 1 | \Sigma C_B$ with a fixed machine-job assignment.

Theorem 10. *Algorithm GR produces an optimal batch sequence for* $P2 \mid P_i = 1 \mid \Sigma C_{B_i}$ *with a fixed machine-job assignment in* $O(b^2)$ *time.*

Proof. Since $P_i = \bar{P}$ for all $i \in B$, from (4), the solution value of this problem is

$$z = \frac{b\bar{P} + (b-1)\bar{P} + \cdots + \bar{P} + \sum_{i=1}^{b} \delta_i}{2} = \frac{b(b+1)\bar{P}}{4} + \frac{\sum_{i=1}^{b} \delta_i}{2}. \tag{5}$$

Note that the first term of (5) is fixed regardless of the batch sequence. Hence, an optimal schedule of the problem produces the minimum value of $\sum_{i=1}^{b} \delta_i$. For notational convenience, let $\Delta_i = P_i^1 - P_i^2$ for $i = 1, 2, \cdots, b$.

We create a new batch scheduling problem with b batches where completion times of the batches are equal to the δ_i's of the original problem for $i = 1, 2, \cdots, b$. Let $\hat{P}_i$ be processing time of batch $i \in B$ in the new problem. Set $\hat{P}_i^1 = \Delta_i$ and $\hat{P}_i^2 = 0$ if $\Delta_i \geq 0$, and set $\hat{P}_i^1 = 0$ and $\hat{P}_i^2 = \Delta_i$ if $\Delta_i < 0$ for all $i \in B$.

Then, the new problem becomes $P2 \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment where $n_i = 1$ for all $i \in B$ and either $\hat{P}_i^1 = 0$ or $\hat{P}_i^2 = 0$ for all $i \in B$. From Theorem 5, heuristic SB finds an optimal schedule for this problem. We use the completion order of this schedule to generate a batch sequence for the original problem. Notice that $\sum_{i=1}^{b} \delta_i$ depends only on Δ_i's. Since we use Δ_i's from the original problem to generate the new problem, a solution value in the new problem must be the same as $\sum_{i=1}^{b} \delta_i$ in the original problem. Hence, we have a schedule with minimum $\sum_{i=1}^{b} \delta_i$ for the original problem and thus, this schedule is optimal for the original problem.

Now, we show that the batch sequence achieved by applying heuristic SB to the new problem is actually the same batch sequence achieved by applying algorithm GR to the original problem. Suppose that batch i completes first in an optimal schedule for the new problem. Note that $\Delta_i = \min_{r \in B}\{\Delta_r\}$. Hence, GR also selects batch i first. The second batch to complete in the optimal schedule for the new problem is batch $\arg\min_{r \in B \setminus \{i\}}\{\mid \Delta_i + \Delta_r \mid\}$, which is also the second choice by GR. We can repeat this argument for the rest of batches in B. Consequently, GR finds an optimal batch sequence to $P2 \mid P_i = 1 \mid \Sigma C_{B_i}$ with a fixed machine-job assignment.

Step 1 requires $O(b)$ time. In Step 1, finding a partial schedule requires $O(\sum_{i=0}^{b-1}(b-i)) = O(b^2)$ time. Since all the other operations require $O(b)$ time, the time requirement of GR is $O(b^2)$. Therefore, GR finds an optimal batch sequence for $P2 \mid P_i = 1 \mid \Sigma C_{B_i}$ with a fixed machine-job assignment in $O(b^2)$.

8. COMPUTATIONAL STUDY OF HEURISTICS SB AND GR

We empirically evaluate SB and GR by comparing solution values generated by heuristics with z^{L3} which is generated by L3. Recall that L3 is the tightest lower bound among L1, L2, and L3 on the optimal solution value. Also, we compare the performance of z^{L1} and z^{L2}. Even though GR generates an optimal schedule for $P2 \mid P_i = 1 \mid \Sigma C_{B_i}$ with a fixed machine-job assignment, it is a heuristic for $P2 \mid \mid \Sigma C_{B_i}$ with a fixed machine-job assignment. As performance indicators of SB and GR, we use upper bounds on relative errors z^{SB}/z^{L3} and z^{GR}/z^{L3}, respectively. The computational experiment is performed for the two, three, and five parallel machine cases. We also observe the impact of different factors such as b, n_i, p_j and $E(n)$ on the performances of SB and GR, where $E(\cdot)$ is the expectation operator.

For each problem instance, $n_i \sim DU[1,\bar{n}]$ and $p_i \sim DU[p^{LB},p^{UB}]$, where $\bar{n}$, p^{LB}, and p^{UB} are parameters and where $DU[\ell,u]$ represents a discrete random variable uniformly distributed between ℓ and u. For a given set of test problems, b is fixed. It follows that $E(n_i) = (1+\bar{n})/2$ and $E(n) = bE(n_i) = b(1+\bar{n})/2$.

We generate 1,050 test problems under 35 conditions. To test the effects of varying $E(n)$, we consider three different values of $E(n)$: 16, 100, and 2500. To determine whether different combinations of b and n_i have an impact on the performance of the heuristics, we consider five different combinations of b and n_i for a given value of $E(n)$. Also, to test the effects of varying the number of parallel machines, we consider three different cases: two parallel machines, three parallel machines, and five parallel machines. The impact of varying the number of machines is considered when $E(n)$ =2500.

Table 2. Design for the Computational Study

$p_j \sim$	$DU[1,99]$										$DU[25,75]$		$DU[40,60]$	
$E(n)$	16		100		2500						100		100	
m	2		2		2		3		5		2		2	
	b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$
	1	31	1	199	1	4999	1	4999	1	4999	1	199	1	199
	2	15	4	49	10	499	10	499	10	499	4	49	4	49
	4	7	10	19	50	99	50	99	50	99	10	19	10	19
	8	3	25	7	250	19	250	19	250	19	25	7	25	7
	16	1	100	1	2500	1	2500	1	2500	1	100	1	100	1

It is also possible that the standard deviation of the p_j's may affect the performance of the heuristics. Consequently, when $E(n)$ =100, we consider three different distributions of p_j: $p_j \sim DU[1,99]$, $p_j \sim DU[25,75]$, and $p_j \sim DU[40,60]$. Their standard deviations are 28.88, 14.43, and 5.77, respectively. For each combination of the different factors, we solve 30 problems. Table 2 presents a summary of the design for the computational study.

The results for the cases where $p_j \sim DU[1,99]$ are presented in Tables 3 and 4. The average relative error is the average ratio of the solution value of a heuristic to z^{L3}. Since each design point has 30 replications, the average relative error is calculated over 30 test problems. When $b=1$, all average relative errors are equal to 1 because both of the heuristics produce an optimal schedule (Theorems 5 and 9). Also, when $\bar{n}=1$, both of the heuristics generate an optimal schedule and errors are due to the use of z^{L3} instead of the optimal value (Theorems 5 and 9).

We now summarize the results of our study. First, observe that L1 and L2 do not dominate each other in Table 3. For a fixed number of jobs, L1 performs better than L2 as b increases. Also, it seems that L1 performs better as the number of jobs increases.

The results in Table 4 indicate that both of the heuristics perform better as $E(n)$ increases. Note that in some cases, the sum of the number of problems where $z^{SB} \le z^{GR}$ and the number of problems where $z^{SB} \ge z^{GR}$ is greater than 30. This is due to the fact that both of the algorithms can generate the same solution value. Heuristic GR consistently performs better than SB except for the case where $E(n)$ =2500 and $b=250$, but GR does not dominate SB. For $E(n)$ =2500, SB performs slightly better as the number of batches increases, but the reverse is true for the cases where $E(n)=16$ and $E(n)=100$. Also, the best performance of SB occurs when $E(n)=2500$.

Table 3. Relative Performance of the Lower Bounds when the Number of Machines is Two

$p_j \sim DU[1,99]$			Average lower bound values			Number of problems	
$E(n)$	b	$\bar{n}$	L1	L2	L3	$z^{L1} \geq z^{L2}$	$z^{L1} \leq z^{L2}$
16	1	31	283	402	402	0	30
	2	15	455	593	603	2	28
	4	7	799	906	930	9	21
	8	3	873	943	1204	14	16
	16	1	2391	1879	2572	26	4
100	1	199	2273	3106	3106	0	30
	4	49	5216	5886	5997	6	24
	10	19	10153	10277	10763	13	17
	25	7	19975	19045	20809	22	8
	100	1	82675	50791	84067	30	0
2500	1	4999	57122	76717	76717	0	30
	10	499	267783	280836	290406	13	17
	50	99	1186883	1118412	1213671	26	4
	250	19	5532894	4903452	5551524	30	0
	2500	1	52362228	27124896	52389868	30	0

Table 4. Performance of the Heuristics when the Number of Machines is Two

$p_j \sim DU[1,99]$			Average relative error bounds		Number of problems	
$E(n)$	b	$\bar{n}$	Heuristic SB	Heuristic GR	$z^{SB} \leq z^{GR}$	$z^{SB} \geq z^{GR}$
16	1	31	1.0000	1.0000	30	30
	2	15	1.0473	1.0331	24	30
	4	7	1.0727	1.0481	15	30
	8	3	1.1011	1.0681	6	25
	16	1	1.0762	1.0762	30	30
100	1	199	1.0000	1.0000	30	30
	4	49	1.0624	1.0337	11	30
	10	19	1.0775	1.0467	3	27
	25	7	1.0860	1.0435	5	25
	100	1	1.0163	1.0163	30	30
2500	1	4999	1.0000	1.0000	30	30
	10	499	1.0711	1.0385	4	26
	50	99	1.0327	1.0338	6	24
	250	19	1.0327	1.0338	19	11
	2500	1	1.0008	1.0008	30	30

The results in Table 5 indicate that L1 performs better when the number of machines is more than two. However, it is still the case that L1 and L2 do not dominate each other.

Table 5. Relative Performance of the Lower Bounds when the Number of Machines Varies

$p_j \sim DU[1,99]$ $E(n)=2500$			Average lower bound values			Number of problems	
m	b	$\bar{n}$	L1	L2	L3	$z^{L1} \geq z^{L2}$	$z^{L1} \leq z^{L2}$
2	1	4999	57122	76717	76717	0	30
	10	499	267783	280836	290406	13	17
	50	99	1186883	1118412	1213671	26	4
	250	19	5532894	4903452	5551524	30	0
	2500	1	52362228	27124896	52389868	30	0
3	1	4999	38447	57745	57745	0	30
	10	499	190767	203850	212041	11	19
	50	99	854338	776280	869644	30	0
	250	19	3881820	3267555	3895012	30	0
	2500	1	34057480	1247740	34977832	30	0
5	1	4999	26474	42262	42262	0	30
	10	499	119171	127730	133557	10	20
	50	99	538635	482388	551230	30	0
	250	19	2424896	1903940	2436828	30	0
	2500	1	20994628	4746149	21014076	30	0

Table 6. Performance of the Heuristics when the Number of Machines Varies

$p_j \sim DU[1,99]$ $E(n)=2500$			Average relative error bounds		Number of problems	
m	b	$\bar{n}$	Heuristic SB	Heuristic GR	$z^{SB} \leq z^{GR}$	$z^{SB} \geq z^{GR}$
2	1	4999	1.0000	1.0000	30	30
	10	499	1.0711	1.0385	4	26
	50	99	1.0327	1.0338	6	24
	250	19	1.0327	1.0338	19	11
	2500	1	1.0008	1.0008	30	30
3	1	4999	1.0000	1.0000	30	30
	10	499	1.1314	1.0706	1	29
	50	99	1.0848	1.0456	1	29
	250	19	1.0528	1.0385	9	21
	2500	1	1.0019	1.0019	30	30
5	1	4999	1.0000	1.0000	30	30
	10	499	1.1543	1.0980	1	29
	50	99	1.1190	1.0595	0	30
	250	19	1.0767	1.0421	0	30
	2500	1	1.0039	1.0039	30	30

From Table 6, observe that GR consistently performs better than SB as the number of machines increases, but SB and GR still do not dominate each other. Also, it seems that the average relative error of the two heuristics increases as the number of machines increases. We suspect that this may be due to the use of z^{L3} instead of the optimal value. Note that L1 performs much better than L2 when the number of machines is more than two, and performance of lower bound L1 (splitting jobs into pieces of any size and processed, simultaneously if desired, on more than two machines without considering machine-job assignment) may become worse as the number of machines increases.

Finally, we compare the average relative error of the two heuristics when the standard deviation (s.d.) of p_j changes in Table 7. When $b=4$ or $b=10$ in Table 7, the average relative error bounds for SB and GR increases slightly as the standard deviation of p_j decreases. But, when $b=25$ or $b=100$ in Table 7, the average relative error for SB and GR decreases slightly as the standard deviation of p_j decreases. In Table 8, we compare the number of test problems where $z^{SB} \le z^{GR}$ and $z^{SB} \ge z^{GR}$.

Table 7. Sensitivity of the Error Bounds to Processing Time Variance when the Number of Machines is Two

$E(n)$ =100		$p_j \sim DU[1,99]$ ($s.d.$ = 28.28)		$p_j \sim DU[25,75]$ ($s.d.$ = 14.43)		$p_j \sim DU[40,60]$ ($s.d.$ = 5.77)	
b	$\bar{n}$	SB	GR	SB	GR	SB	GR
1	199	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	49	1.0624	1.0337	1.0601	1.0351	1.0584	1.0361
10	19	1.0775	1.0467	1.0750	1.0422	1.7710	1.0438
25	7	1.0860	1.0435	1.0774	1.0423	1.0766	1.0426
100	1	1.0163	1.0163	1.0129	1.0129	1.0106	1.0106

Table 8. Sensitivity of the Number of Problems when SB and GR are Best to Processing Time Variance when the Number of Machines is Two

$E(n)$ =100		$p_j \sim DU[1,99]$ ($s.d.$ = 28.28)		$p_j \sim DU[25,75]$ ($s.d.$ = 14.43)		$p_j \sim DU[40,60]$ ($s.d.$ = 5.77)	
b	$\bar{n}$	$z^{SB} \le z^{GR}$	$z^{SB} \ge z^{GR}$	$z^{SB} \le z^{GR}$	$z^{SB} \ge z^{GR}$	$z^{SB} \le z^{GR}$	$z^{SB} \ge z^{GR}$
1	199	30	30	30	30	30	30
4	49	11	30	9	30	12	30
10	19	3	27	3	27	3	28
25	7	5	25	4	26	6	24
100	1	30	30	30	30	30	30

Observe the number of test problems with $z^{SB} \le z^{GR}$ increases slightly as the standard deviation of p_j decreases, except when $b = 25$. The results in Tables 7 and 8 suggest that in general, GR performs better than SB regardless of the change in the standard deviation of p_j.

We may suggest that managers in practice use the both heuristics and apply the best result for the real world problems since SB and GR do not dominate each other and their computational requirement is minimal.

9. DISCUSSION AND FURTHER RESEARCH

We have explored problem $P \mid \mid \Sigma C_{B_i}$ with fixed machine-job assignment. This problem is simpler than many batch scheduling problems but has several practical implications. Producing different types of products on a different machine is a practical assumption, which is true in many real world manufacturing facilities. Also, due to logistics cost, products are shipped out to customer as a whole order not as an individual product. Hence, the objective concerned with batch completion times instead of job completion times is also a suitable assumption. Even though heuristics SB and GR are simple, they are intuitive and provide practical insight to managers. Also, for small sets of orders and machines, the DP algorithm can be applied to obtain an optimal schedule.

In this work, the worst case bound on the relative error of the heuristics SB and GE are not tight. Hence, future research can find their tight worst case bounds. Also, the both heuristics can be analyzed further to provide more information on its performance.

The results of this paper can be used to develop solution procedures for more complex and realistic applications. There are several extensions of our research that might be considered. One extension is to study our problem with different machine speeds such as proportional and unrelated parallel machines. Also, different shop environments can be considered, 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 processing orders at an internet shopping mall, $\sum w_i T_{B_i}$ is a more realistic objective.

REFERENCES

[1] Ahmadi, R., U. Bagchi, and T. A. Roemer, “Coordinated Scheduling of Customer Orders for Quick Response,” Working Paper, Anderson School of Management at UCLA, USA 2004.

[2] Baker, K. R., “Scheduling the Production of Components at a Common Facility,” *IIE Transactions* 20 (1988), 32-35.

[3] Blocher, J. D. and D. Chhajed, “The Customer Order Lead Time Problem on Parallel Machines,” *Naval Res. Logist.* 43 (1996), 629-654.

[4] Blocher, J. D., D. Chhajed, and M. Leung, “Customer Order Scheduling in a General Job Shop Environment,” *Decision Science* 29 (1998), 951-981.

[5] Coffman, E. G., A. Nozari, and M. Yannakakis, “Optimal Scheduling of Products with Two Subassemblies on a Single Machine,” *Operations Research* 37 (1989), 426-436.

[6] Ding, F. Y., “A Pairwise Interchange Solution Procedure for a Scheduling Problem with Production of Components at a Single Facility,” *Computers and Industrial Engineering* 18 (1990), 325-331.

[7] Gerodimos, A. E., C. A. Glass, and C. N. Potts, “Scheduling the Production of Two-Component Jobs on a Single Machine,” *European Journal of Operational Research* 120 (2000), 250-259.

[8] Graham, R. L., E. L. Lawler, J. K. Lenstra, and A .H. G. Rinnooy Kan, “Optimization and Approximation in Deterministic Machine Scheduling: A Survey,” *Annals of Discrete Mathematics* 5 (1979), 287-326.

[9] Gupta, J. N. D., J. C. Ho, and A. A. van der Veen, “Single Machine Hierarchical Scheduling with Customer Orders and Multiple Job Classes,” *Annal of Operations Research* 70 (1997), 127-143.

[10] Jordan, C., *Batching and Scheduling: Models and Methods for Several Problem Classes,* Springer, New York 1996.

[11] Julien, F. M. and M. J. Magazine, “Scheduling Customer Orders: An Alternative Production Scheduling Approach,” *Journal of Manufacturing and Operations Management* 3 (1990), 177-199.

[12] Liao, C. J., “Optimal Scheduling of Products with Common and Unique Components,” *International Journal of Systems Science* 27 (1996), 361-366.

[13] Potts, C. N. and L. N. Van Wassenhove, “Integrating Scheduling with Batching and Lot-sizing: a Review of Algorithm and Complexity,” *Journal of Operational Research Society* 43 (1992), 395-406.

[14] Roemer, T. A., and R. Ahmadi, “The Complexity of Scheduling Customer Orders,” Working Paper, Anderson School of Management at UCLA, USA

1997.

[15] Santos, C. and M. Magazine, "Batching in Single Operation Manufacturing Systems," *Operations Research Letters* 4 (1985), 99-103.

[16] Webster, C. and K. R. Baker, "Scheduling Groups of Jobs on a Single Machine," *Operations Research* 43 (1995), 692-703.

[17] Yang, J. and M. E. Posner, "Scheduling Parallel Machines for the Customer Order Problem," *Journal of Scheduling* 8 (2005), 49-74.

[18] Yang, J., "Scheduling Parallel Machines for the Customer Order Problem with Fixed Bach Sequence," *Journal of the Korean Institute of Industrial Engineers* 29 (2003) 304-311.

[19] Yang, J., "The Complexity of Customer Order Scheduling Problems on Parallel Machines," *Computers and Operations Research* 32 (2005), 1921-1939.

[20] Yoon, S. H., "Fabrication Scheduling of Products with Common and Unique Components at a Single Facility," *Journal of the Korean Operations Research and Management Science Society* 28 (2003), 105-114.