Customer Order Scheduling Problems with Fixed Machine-Job Assignment

Jaehwan Yang
Department of Business Administration, University of Incheon
177 Dohwa-dong, Namgu, Incheon, 402-749, Korea

Abstract

This paper considers a variation of customer order scheduling problems. The variation is the case where machine-job assignment is fixed, and the objective is to minimize the sum of the completion times of the batches. In customer order scheduling problems, jobs are dispatched in batches. While a machine can process only one job at a time, multiple machines can simultaneously process jobs in a batch. We first establish a couple of lower bounds. Then, we develop a dynamic programming (DP) algorithm that runs in exponential time on the number of batches when there exist two machines. For the same problem with arbitrary number of machines, we present two simple heuristics, which use simple scheduling rules such as shortest batch first and shortest makespan batch first rules. Finally, we empirically evaluate the heuristics.

1. **Introduction**

This work considers a scheduling problem where each job is part of some batch (customer order). The composition of the jobs in the batch is prespecified. While a machine can process only one job at a time, multiple machines can simultaneously process jobs in the batch. The completion time of the batch is the latest completion time of any job in the batch. A restriction on this environment is that *machine-job assignment is fixed*. Hence, jobs are processed by pre-assigned machines. The objective is to minimize the sum of the batch completion times.
This problem is introduced by Roemer and Ahmadi (1997) and Roemer *et al.* (2002). 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 because of the nature of processing. The customer order cannot be shipped to the customer unless the entire order is completed.
This example can be further extended to general customer order scheduling problems. 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. Usually, a different machines process a different product.
For the batch scheduling problem where the completion time is based on the latest completion time of a job in the batch, Julien and Magazine (1990) examine 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.* (1989) examine a similar problem where the batch processing order is not fixed. They develop an $O(n^{1/2})$ procedure. Baker (1988) considers a problem similar to Coffman *et al.* (1989).
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, 1985). Gupta *et al.* (1997) 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.* (2000) study single machine problems where each batch has one common job and one distinct job. Ding (1990), Liao (1996), and Yoon (2003) also study the similar problem.
In the batch (or customer order) scheduling problems that we consider in this work, there exist no setup times between different jobs or different batches. Blocher and Chhajed (1996) examine the problem with no restriction on machine-job assignment, minimizing the sum of batch completion times in a parallel machine environment. They show that the problem is NP-hard, develop several heuristic methods, and two lower bounds. Yang and Posner (2003) also consider the same problem, and develop three heuristics and find their tight lower bounds. Yang (2003) considers a variation of customer order scheduling problem where batch sequence is fixed. He establishes the complexity of the problem and develops a DP algorithm, which runs in pseudo-polynomial time. Yang (2004) also establishes the complexity of different customer order scheduling problems and summarizes the known complexity results.
Our problem focuses on the customer order scheduling problem with a restriction such that the machine-job assignment is fixed. Roemer and Ahmadi (1997) show that the problem is NP-hard for two machine case. An easier complexity proof is presented by Yang (2004).
We first introduce some notation. Next, we study the customer order scheduling problem with fixed machine-job assignment. The objective is minimization of sum of batch completion times. We first provide review on some preliminary results including complexity of the problem. Then, we establish two simple lower bounds for the problem. For the two machine case, we develop a DP algorithm, which runs in exponential time. Since the problem is unary NP-complete, we provide two simple and intuitive heuristics for the case where the number of machines is arbitrary. Finally, we provide the result of computational study.

2. Notation

We define some notation that is used in this work. Let

n	= number of jobs
N	= set of jobs = $\{1,2,K,n\}$
b	= number of batches
B	= set of batches = $\{1,2,K,b\}$
n_i	= number of jobs in batch i for $i \in B$
M	=set of machines= $\{1,2,K,m\}$
B_i	= set of jobs in batch i for $i \in B$

B_i^k = set of jobs assigned to machine k for $k \in M$ in batch $i \in B$
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 $i \in B$
σ_k = schedule of all jobs on machine k for $k \in M$
σ = schedule of all jobs
$C_i(\sigma_k)$ = 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 represent $C_i(\sigma)$ as C_i when there is no ambiguity. The standard classification scheme for scheduling problems (Graham *et al.*, 1979) is $\alpha_1 | \alpha_2 | \alpha_3$, 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.

3. Lower Bounds

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

Remark 1. (Blocher and Chhajed, 1996) *For problem* $P || \Sigma C_{B_i}$,

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

Observe that the left side of the inequality (1) is the optimal solution value to the Linear Programming (LP) relaxation of problem $P || \Sigma C_{B_i}$ 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. Since L1 is a lower bound for problem $P || \Sigma C_{B_i}$ and problem $P || \Sigma C_{B_i}$ with fixed machine-job assignment is a special case of $P || \Sigma C_{B_i}$, L1 is also a lower bound for $P || \Sigma C_{B_i}$ with fixed machine-job assignment.

Now, we consider another lower bound L2.

Lower Bound L2.

```
0. Set $\hat{B}_i^k = B_i^k$ if $B_i^k \ne \phi$ otherwise $\hat{B}_i^k = \phi$ for $i \in B$ and $k \in M$.
   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 $i = 1,2,\mathrm{K},b-1$.
2. Create a new set of batches $\hat{B}_i$ such that
   $\hat{B}_i = \hat{B}_i^1 \cup \hat{B}_i^2 \cup \Lambda \cup \hat{B}_i^m$ 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 $O(mb)$ 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.

Remark 2. *L2 is a lower bound of an optimal solution value for* $P || \Sigma C_{B_i}$ *with fixed machine-job assignment.*

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 $i \in B$, $C_i(\sigma^*) = \max\{C_i(\sigma_1^*), C_i(\sigma_2^*), \mathrm{K}, C_i(\sigma_m^*)\}$. Let $C_{[i]}(\sigma^*)$ be completion time of i th completed batch for $i \in B$ and $k \in M$. Then, for each $i \in B$ and $k \in M$, $C_{[i]}(\sigma_k^*) \ge C_i(\sigma_k^{L2}) = P_1^k + P_2^k + \Lambda + P_i^k$. Hence,

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

4. Preliminary Results

4.1 General Results

In this section, we provide reviews on some of preliminary results for this problem.

Lemma 1. (Yang, 2004) *For scheduling problems with regular measures, there exists an optimal schedule without inserted idle time.*

As a result of Lemma 1, we only consider a schedule with this property. 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, 2004) *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 , 1996) *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 (1997).

Theorem 1. *The recognition version of* $P \| \Sigma C_{B_i}$ *with a fixed machine-job assignment is unary NP-complete.*

5. DP Algorithm

Now, we present a DP algorithm for $P2 \| \Sigma C_{B_i}$ 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 $P2 \| \Sigma C_{B_i}$ with fixed machine-job assignment in $O(b2b^2)$ time.

Let $f(V)$ be minimum total completion time of the batches in set V for $V \subseteq B$. 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. As an initial condition, we let $f(\phi)=0$. Define $f(V)$ recursively as follows:

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

Observe that $\max\{\sum_{i\in V} P_i^1, \sum_{i\in V} P_i^2\}$ 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 $V \subseteq B$ and $1 \leq |V| \leq b-1$. To record the optimal choice at each iteration, let $v(V)$ represent the last batch in a schedule which corresponds to $f(V)$. Then,

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

We now formally describe a DP procedure that finds an optimal schedule for $P2 \| \Sigma C_{B_i}$ with a fixed machine-job assignment.

Algorithm B1.

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

The following theorem verifies the optimality of Algorithm B1.

Theorem 2. *Algorithm B1 produces an optimal job-machine assignment to* $P2 \| \Sigma C_{B_i}$ *with a fixed machine-job assignment in* $O(b2^b)$ *time.*

Proof. Breaking ties arbitrarily in Step 1 does not change the optimal solution value because given $V \subseteq B$, the last batch completes at $\max\{\sum_{i\in V} P_i^1, \sum_{i\in V} P_i^2\}$ 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 B1, consider the determination of $f(V)$ in (1) for any set $V \subseteq B$. For all $V' \subseteq V$ such that $|V'| = V|-1$, $f(V')$ is known. Recall that for any $V \subseteq B$, 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 $V' \subseteq V$ 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 $\sum_{i=1}^b C_i(\sigma^*) = f(B)$.

Let $\binom{n}{k}$ be the number of ways of choosing $k \leq n$ objects from a collection of n distinct objects without regard to order. Step 0 requires $O(b)$ time. Step 1 considers $\binom{b}{i}$ different sets of batches for $i = 1,2,\ldots,b$. Also, for each set, i different candidates are compared. Thus, Step 1 requires $O(i \cdot \binom{b}{i})$ time. Step 1 repeats for $i = 1,2,\ldots,b$. Steps 3, 4, 5, and 6 require constant time, and they repeat b times. Step 7 requires constant time. Now,

$$\begin{aligned}\sum_{i=1}^{b}\left(i,\binom{b}{i}\right) &= 1\cdot\binom{b}{1} + 2\cdot\binom{b}{2} + \Lambda + b\cdot\binom{b}{b} \\ &= \frac{1+(b-1)}{2}\binom{b}{1} + \frac{2+(b-2)}{2}\binom{b}{2} + \binom{b}{b} \\ &\quad + \Lambda + \frac{(b-1)+1}{2}\binom{b}{b-1} + b \\ &= b\left[\frac{\left(\sum_{i=1}^{b}\binom{b}{i}\right)-1}{2}\right] + b = b\left[\frac{(2^b-1)-1}{2}\right]+b \\ &= b2^{b-1}.\end{aligned}$$

Therefore, B1 finds a solution in $O(\sum_{i=1}(i\cdot\binom{b}{i})) = O(b2^b)$ time. Because the number of elementary operations and size of all values are bounded by a exponential function of the input length, B1 runs in exponential time. □

6. An Example

We now illustrate the algorithm with an 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, B1 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 B1.

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))

7. Heuristics

In this section, we present two simple heuristics. First, heuristic SB 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 determine the batch sequence. We assume that the order of job is arbitrary.

Heuristic SB

```
0. Reindex batches so that $P_i \le P_{i+1}$ for $i = 1,2,\ldots,b-1$.
1. Schedule all jobs in batches in index order. When there
   exists ties, break it 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 $O(b \log b + n)$.

Now, we introduce the next heuristic. Heuristic SM first calculates makespan of each batch with given machine-job relationship. Then, schedule batches with smallest makespan.

Heuristic SM

```
0. Set $F_q = 0$ for $q = 1,2,\ldots,b$.
1. Calculate $F_i = \max\{P_i^1, P_i^2\}$ for $i = 1,2,\ldots,b$.
2. Reindex batches so that $F_i \le F_{i+1}$ for $i = 1,2,\ldots,b-1$.
3. Schedule all jobs in batches in index order.
4. Output $\sum_{i=1}^{b} C_i$ and stop.
```

Steps 0 and 1 require $O(b)$ time. In Step 2, reindexing batches requires $O(b \log b)$ time. Since all other operations require $O(n)$ time, the time requirement of SM is $O(b \log b + n)$.

The next remark finds some special cases where heuristics SM and SB generates an optimal schedule.

Remark 3. *If* $n_i = 1$ *for all* $i \in B$ *or* $b = 1$, *then heuristics SM and SB generate an optimal schedule for problem* $P \| \Sigma C_{B_i}$ *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 and then the solution value for each machine can be added up to calculate ΣC_{B_i} for the entire problem. Recall that each batch contains only one job. Then, for each machine, the problem reduces to $1 \| \Sigma C_i$. Consequently, heuristic SB is optimal. Also, if $n_i = 1$, then heuristic SM generates the same schedule as heuristic SB. □

8. Computational Study

We empirically evaluate heuristics SB and SM by comparing solution values generated by heuristics to an optimal LP solution value z^{L1} and a lower bound L2 z^{L2}. Also, we compare the performance of z^{L1} and z^{L2}. For notational convenience, we let $z^{L*} = \max\{z^{L1}, z^{L2}\}$. As performance indicators of SB and SM, we use upper bounds on relative errors z^{SB}/z^{L*} and z^{SM}/z^{L*}, respectively.

We observe the impact of different factors such as b, n_i, and $E(n)$ on the performances of SB and SM, where $E(\cdot)$ is the expectation operator. For each problem instance, $m = 2$, $n_i \sim DU[1, \bar{n}]$ and $p_i \sim DU[1,99]$, where $\bar{n}$ is a parameter and where $DU[\lambda, u]$ represents a discrete random variable uniformly distributed between λ 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 450 test problems under 15 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 $E(n)$. For each combination of the different factors, we solve 30 problems. Table 2 presents a summary of the design for the computational study.

Table 2. Design for the Computational Study.

$E(n)=16$		$E(n)=100$		$E(n)=2500$	
b	$\bar{n}$	b	$\bar{n}$	b	$\bar{n}$
1	31	1	199	1	4999
2	15	4	49	10	499
4	7	10	19	50	99
8	3	25	7	250	19
16	1	100	1	2500	1

The results are presented in Table 3. The average relative error bound is the average ratio of the solution value of a heuristic to z^{L*}. Since each design point has 30 replications, the average relative error is calculated over 30 test problems. When $b = 1$, all average relative error bounds are equal to 1 because the both heuristics produce an optimal schedule (Remark 4). Also, when $\bar{n} = 1$, the both heuristics generate an optimal schedule and errors are due to the use of z^{L*} instead of the optimal value (Remark 4).

We now summarize the results of our study. First, L1 and L2 do not dominate each other. For a given $E(n)$, L1 performs better than L2 as b increases. Also, it seems that L1 performs better as the number of jobs increases. The results indicate that both heuristics perform better as the number of jobs increases. While $E(n)$ is small, SM performs well compared to SB, but as $E(n)$ increases, performance of SB is getting better than SM. For a given $E(n)$, heuristic SB performs slightly better than heuristic SM as b increases except for $E(n)=16$.
Since SB and SM do not dominate each other and the computational load is minimal, we may suggest that managers in practice use the both heuristics and pick the best result.

9. Discussion and Further Research

We have explored problem $P\,||\,\Sigma C_{B_i}$ with fixed machine-job assignment. We provide a lower bound and for two machine case, we develop a DP algorithm, which runs in exponential time. We also develop two heuristics and perform the computational study. Even though heuristics are very simple, they are intuitive and provide practical insight to the managers in the real world field.
For further research, we want to explore the characteristics of the heuristics further. For example, we want to establish the worst case bounds on the relative errors for those heuristics considered. Also, we like to expand our research to the problem with the arbitrary number of machines.

References

Baker, K. R. (1988), Scheduling the Production of Components at a Common Facility. *IIE Trans.* **20**, 32-35.
Blocher, J. D., and Chhajed D. (1996), The Customer Order Lead Time Problem on Parallel Machines, Naval Res. Logist. 43, 629-654.
Coffman, E. G., Nozari A., and Yannakakis M. (1989), Optimal Scheduling of Products with Two Subassemblies on a Single Machine, Opns. Res. 37, 426-436.
Ding, F.Y. (1990), " A Pairwise Interchange Solution Procedure for a Scheduling Problem with Production of Components at a Single Facility," Computers and I.E., 18, 325-331.
Gerodimos, A. E., Glass C. A., and Potts C. N. (2000), Scheduling the Production of Two-Component Jobs on a Single Machine, Euro. J. Opnl. Res. 120, 250-259.
Graham, R.L., Lawler E.L., Lenstra J.K., and Rinnooy Kan A.H.G. (1979), Optimization and Approximation in Deterministic Machine Scheduling: A Survey. Annals of Discrete Mathematics 5, 287--326.
Gupta, J. N. D., Ho J. C., and van der Veen A. A. (1997), Single Machine Hierarchical Scheduling with Customer Orders and Multiple Job Classes. Ann. Oper. Res 70, 127-143.
Julien, F. M., and Magazine M. J. (1990), Scheduling Customer Orders: An Alternative Production Scheduling Approach, J. Mfg. Oper. Mgt. 3, 177-199.
Liao, C.J. (1996), Optimal Scheduling of Products with Common and Unique Components," International Journal of Systems Science, 27, 361-366.
Roemer, T.A., and Ahmadi R. (1997) The Complexity of Scheduling Customer Orders, Working Paper, Anderson School of Management at UCLA, USA.
Ahmadi, R., Bagchi U., and Roemer T.A. (2002) Coordinated Scheduling of Customer Orders for Quick Response, Working Paper, Anderson School of Management at UCLA, USA
Santos, C., and Magazine M. (1985) Batching in Single Operation Manufacturing Systems, OR Lett. 4, 99-103.
Yang, J. and Posner, M. E. (2003), Scheduling Parallel Machines for the Customer Order Problem, Working Paper, Department of Industrial Welding and Systems Engineering, The Ohio State University.
Yang, J. (2003), Scheduling Parallel Machines for the Customer Order Problem with Fixed Bach Sequence, Journal of the Korean Institute of Industrial Engineers 29, 304-311.
Yang, J. (2004), The Complexity of Customer Order Scheduling Problems on Parallel Machines, to appear in Computers and Operations Research.
Yoon, S. H. (2003), Fabrication Scheduling of Products with Common and Unique Components at a Single Facility, Journal of the Korean Operations Research and Management Science Society, 28 105-114.

Table 3. Results of Computational Study

			Number of problems		Avg. relative error bounds		Number of problems	
$E(n)$	b	$\bar{n}$	$z^{L1} \geq z^{L2}$	$z^{L1} \leq z^{L2}$	z^{SM}/z^{L^*}	z^{SM}/z^{L^*}	$z^{SB} \leq z^{SM}$	$z^{SB} \geq z^{SM}$
16	1	31	0	30	1.000	1.000	30	30
	2	15	1	29	1.051	1.044	27	30
	4	7	8	22	1.124	1.106	19	28
	8	3	19	11	1.229	1.234	16	18
	16	1	26	4	1.233	1.233	30	30
100	1	199	0	30	1.000	1.000	30	30
	4	49	9	21	1.119	1.117	19	25
	10	19	16	14	1.142	1.138	14	17
	25	7	24	6	1.148	1.151	18	12
	100	1	30	0	1.102	1.102	30	30
2500	1	4999	0	30	1.000	1.000	30	30
	10	499	19	11	1.138	1.135	13	17
	50	99	27	3	1.080	1.135	19	11
	250	19	30	0	1.043	1.065	29	1
	2500	1	30	0	1.021	1.021	30	30