International Journal of Management Science
Vol 17, No 1, May 2011

Customer Order Scheduling in a Two Machine Flowshop

Jaehwan Yang*
College of Business Administration, University of Seoul

(Received: September 16, 2010 / Revised: May 3, 2011 / Accepted: May 3, 2011)

ABSTRACT

This paper considers a flowshop scheduling problem where a customer orders multiple products (jobs) from a production facility. The objectives are to minimize makespan and to minimize the sum of order (batch) completion times. The order cannot be shipped unless all the products in the order are manufactured. This problem was motivated by numerous real world problems encountered by a variety of manufacturers. For the makespan objective, we develop an optimal solution procedure which runs in polynomial time. For the sum of order completion time objective, we establish the complexity of the problem including several special cases. Then, we introduce a simple heuristic and find an asymptotically tight worst case bound on relative error. Finally, we conclude the paper with some implications.

Keywords: Flowshop, Customer Order Scheduling, Heuristics, Complexity Analysis

1. Introduction

This paper considers a customer order scheduling problem in a two machine flowshop environment. To the best of authors' knowledge, no previous research has considered this particular customer order scheduling problem in a flowshop environment.

In the customer order scheduling problem, each order has a set of products (jobs), called a batch, which needs to be processed. Only after all jobs in the batch are completed, the products in the order can be shipped to the customer. The composition of the jobs in the batch is prespecified. The completion time of the batch is the latest completion time of any job in the batch. Further, the objective is associated with the

* Corresponding author, E- mail: jyang@uos.ac.kr

completion time of the batches instead of the completion time of each job. Preemption of batches, i.e. processing a job in batch A and then processing a job in batch B, and maybe sometime later processing another job in batch A, is not allowed except for the case where it is specified. Also, no setup times are assumed between different jobs or different batches.

Several studies consider various customer order scheduling problems. 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. Julien and Magazine [15] study a single machine problem where the objective is to minimize the total batch completion time. A job-dependent setup time is assumed between two different types of jobs. They develop a dynamic programming algorithm that runs in polynomial time for the problem where there exist two types of jobs and the batch processing order is fixed. Coffman *et al.* [6] consider a similar problem where the batch processing order is not fixed. Baker [2] also considers a problem similar to Coffman *et al.* [6]. However, for one type of job, those jobs processed during the same production run are not available until the completion of the production run. Gupta *et al.* [12] 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.* [9] study single machine problems where each batch has one common job and one distinct job. Ding [7] and Liao [16] also examine similar problems.

Blocher and Chhajed [3] examine the customer order scheduling problem in the parallel machine environment where the objective is to minimize the sum of order completion times. They show that the recognition version of the problem is unary NP-complete for the three parallel machine case and is at least binary NP-complete for the two parallel machine case. Also, they develop several heuristic methods and two lower bounds. Blocher *et al*. [4] extend the problem to a job shop. Yang and Posner [19] consider the same problem with two parallel machines, and introduce three simple heuristics and find tight worst case bounds on relative errors. Yang [18] establishes the complexity of different customer order scheduling problems. When the machine-job assignment is fixed, Roemer and Ahmadi [17] show that the recog-nition version of the problem is unary NP-complete for the two parallel machine case with the objective of minimizing the sum of batch completion times. 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.

Recently, Blocher and Chhajed [5] consider a customer order scheduling problem in a two-stage supply chain which is somewhat similar to a flowshop environment when it has only one machine at stage 1. In the two-stage supply chain, there exist multiple processors for fabrications and a single processor for assembly. Their objective is to minimize the sum of order completion time. They assume that each order contains m+1 components where m is the number of processors in stage 1 and the other component is processed in an assembly machine after the m components are processed in stage 1. They suggest several lower bounds and heuristics for the problem. Since their problem is much more complicated, the lower bounds and heuristics they suggest may not provide direct implications for our problem.

We first introduce notation. Then, we establish some basic results of an optimal schedule for the problems with the both objective functions. For the problem with the objective of minimizing makespan, we develop a schedule which generates an optimal schedule. For the problem with the objective of minimizing total batch completion time, we establish complexity of some special cases. Also, we develop a series of lower bounds and introduce a simple but intuitive heuristic. We find the worst case bound on relative error and the bound is tight as the number of batches goes to infinity. Finally, we conclude the paper with possible future studies.

2. Notation

The decision variables in our models are

σ_k = schedule of all jobs or batches on machine k for $k \in \{1, 2\}$

σ = schedule of all jobs or batches $= (\sigma_1, \sigma_2)$

Other notation that is used in this work include

n = number of batches

N = number of jobs

B = set of batches $= \{1, 2, \cdots, n\}$

J = set of jobs $= \{1, 2, \cdots, N\}$

n_i = number of jobs in batch i for $i \in B$

M_k = machine k for $k \in \{1, 2\}$

p_j = total processing time of job j for $j \in J$

p_{kj} = processing time of job j for $j \in J$ on machine k for $k \in \{1, 2\}$

C_{ki} = completion time of batch i on machine k for $k \in \{1, 2\}$ and $i \in B$

$C_i(\sigma)$ = completion time of batch i in schedule σ for $i \in B$

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.* [11]) 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. In the first field, F means the flowshop machine structure, and the number after F indicates a fixed number of machines rather than an arbitrary number. Also, some restriction may be placed in the α_2 to describe the batch characteristic. Finally, we extend this scheme to provide for batch completion times by using C_{B_i} and for makespan for batch scheduling $C_{B_{max}}$ in the α_3 field. This notation is used to eliminate the confusion between our problem and the classical scheduling problem. For example, $F2 | | \sum C_{B_i}$ is the problem where there exist two machine in a flowshop and the objective is to minimize the total batch completion time.

3. Prelimary Results

In this section, we establish some optimal properties for both problems $F2 | | C_{B_{max}}$ and $F2 | | \sum C_{B_i}$. First, note that there is no restriction on delaying jobs. Hence, for problems $F2 | | C_{B_{max}}$ and $F2 | | \sum C_{B_i}$, there exists an optimal schedule without inserted idle time.

Johnson's Rule (JR) in Johnson [14] is a seminary result in the flowshop scheduling with the objective of minimizing makespan and it is described as follows: (1) list the jobs and their times at each machine; (2) select the job with the shortest time, and if the job is for the first machine, then schedule the job first. Otherwise, schedule the job last. Break ties arbitrarily; (3) eliminate the job selected from further consi-

deration; (4) repeat steps 2 and 3 until all jobs have been scheduled.

In order to solve problems $F2||C_{B_{max}}$ and $F2||\sum C_{B_i}$, we need to modify JR slightly and introduce an algorithm called MJR (Modified Johnson's Rule). This algorithm is developed to solve a regular flowshop scheduling problem $F2||C_{max}$ not the batch scheduling problems which we eventually want to solve. However, it will contribute to solve problems $F2||C_{B_{max}}$ and $F2||\sum C_{B_i}$ later in the paper by determining a job schedule for each batch. The schedule generated by MJR is similar to that byJR except for that it does not have idle time on M_2 between completion time of the first job and start time of the last job. We now formally describe the algorithm.

Algorithm MJR

0. Apply JR to schedule jobs $j = 1, 2, \cdots, n$.
   Reindex jobs such that jobs are ordered in their index order.
1. Starting from the last job, if there exist some idle time before the job on M_2, then delay the preceding job on M_2 so that their exist no idle time before the job which we currently consider.
2. Repeat this process the preceding job until there exists no idle time between completion time of job 1 and start time of job n.
3. Calculate C_j for $j = 1, 2, \cdots, n$.
   Output C_{max} and stop.

Algorithm MJR runs in $O(n \log n)$ time. The resulting solution is a permutation schedule where there is no idle time on M_1 and M_2 after a job starts on each machine. Notice that a schedule which generated by using JR may have some idle time on M_2.

Since makespan generated by MJR is no greater than one by JR, MJR generates an optimal schedule for $F2||C_{max}$. For the batch scheduling problems, we will separtely apply MJR to each batch in order to generate a schedule for jobs in the batch. For notational convenience, we define some additional notation which will be used throughout the paper. We assume that jobs are scheduled in their index order after we apply MJR to jobs in each batch. First, we let C^i_{max} denote the completion time of

batch $i \in \{1, 2, \cdots, n\}$ when it starts at time 0 and its jobs are scheduled by MJR. Then, for each batch $i \in \{1, 2, \cdots, n\}$, we let

A_i = idle time of batch i on M_2 before the first job starts on M_2

$= C^i_{max} - \sum_{j=1}^{n_i} p_{2j}$

B_i = idle time of batch i on M_1 until completion time of the last job on M_2

$= C^i_{max} - \sum_{j=1}^{n_i} p_{1j}$

K_i = processing time of batch i when the both machines are busy

$= \{\sum_{j=1}^{n_i} (p_{1j} + p_{2j}) - A_i - B_i\}/2$.

The next lemma establishes that an important property of MJR.

Lemma 1: *For problems* $F2||C_{B_{max}}$ *and* $F2||\sum C_{B_i}$, *there exists an optimal schedule where jobs in each batch are scheduled by MJR.*

Proof: We prove for problem $F2||\sum C_{B_i}$ only. The proof is similar for the other case. Suppose that there does not exist an optimal schedule where jobs in each batch are scheduled by MJR. Consider an optimal schedule σ where jobs in some batches are not scheduled by MJR and so, it gives bigger makespan for those batches. Suppose that batch i is the first such batch and also suppose that batches are scheduled in their index order in σ. For notational con-venience, let $D_{i-1} = C_{2,i-1}(\sigma) - C_{1,i-1}(\sigma)$ for $i = 2, 3, \cdots, n$.

Case 1: $A_i \geq D_{i-1}$.

Suppose that we apply MJR to batch i and schedule their jobs after batch i-1. Since $A_i \geq D_{i-1}$ and MJR guarantees the smallest makespan, completion time of batch i will be smaller then $C_i(\sigma)$.

Case 2: $A_i < D_{i-1}$.

The condition implies that there is no idle time on M_2 between completion times of batches i-1 and i if we schedule jobs in batch i by MJR and put them after batch i-1. No idle time on M_2 means the smallest possible completion time of batch i.

From Cases 1 and 2, we can see that completion time of batch i would be no

greater than $C_i(\sigma)$ if we schedule jobs in batch i by MJR. We can repeat this argument for following batches in σ of which jobs are not scheduled by MJR. □

As a result of Lemma 1, we only consider those schedules where jobs are scheduled by MJR.

4. Problem $F2||C_{B_{max}}$

4.1 A Special Case

The JR indeed generates an optimal schedule for a special case where the batch preemption is allowed. The following theorem establishes problem $F2||C_{B_{max}}$ with batch preemption can be easily solved by using JR.

Theorem 1: *The JR is optimal for problem $F2||C_{B_{max}}$ with batch preemption.*

Proof: Note thatJR is an optimal solution procedure for problem $F2||C_{max}$. Since the batch preemption is allowed, we can treat each job in batches as a job in problem $F2||C_{max}$ without associating the job with a batch. □

However, it sounds more realistic that in a two machine flowshop, if a job in a batch starts being processed on a machine, then all jobs in the batch should complete before a job in another batch starts being processed on the same machine. Also, the processing order of batches on each machine should be identical to one another. Hence, in a general version of problem $F2||C_{B_{max}}$, we assume that there does not exist batch preemption. Then, it can be seen that JR does not guarantee an optimal schedule if batch preemption is not allowed.

4.2 An Optimal Solution Procedure for Problem $F2||C_{B_{max}}$

In this section, we introduce a solution procedure which generates an optimal schedule for $F2||C_{B_{max}}$. After jobs in each batch are scheduled by MJR, we treat each schedule of a batch as a job and apply JR to solve the whole problem, $F2||C_{B_{max}}$.

Specifically, when we apply JR to batches, we treat A_i as processing time on M_1 and B_i as processing time on M_2. We formally describe Algorithm MJRB (Modified Johnson's Rule for Batch scheduling) which generates an optimal schedule for problem $F2||C_{B_{max}}$.

Algorithm MJRB

0. Apply MJR to each batch i for $i = 1, 2, \cdots, n$.
   Calculate $A_i, B_i,$ and K_i for $i = 1, 2, \cdots, n$.
1. Apply JR to schedule batches by treating A_i as processing time on M_1 and B_i as processing time on M_2, respectivly for $i \in B$.
2. Calculate C_i for $i = 1, 2, \cdots, n$.
   Output $C_{B_{max}}$ and stop.

If the number of jobs of all batches is N, then MJRB runs in $O(N \log N)$ time. The resulting solution is a permutation schedule where there is no idle time on M_1. The next theorem proves that MJRB generates an optimal schedule for $F2||C_{B_{max}}$.

Theorem 2: *MJRB generates an optimal schedule for problem* $F2||C_{B_{max}}$.

Proof: From Lemma 1, we only need to prove that applying JR to each batch of which jobs are already scheduled by MJR generates an optimal schedule for problem $F2||C_{B_{max}}$. Suppose that there exists a schedule which is generated by MJRB and is not an optimal schedule. Let σ and z^{σ} be this schedule and the solution value, respectively. Also, let z^* is an optimal solution value for the problem. Now, we suppose another schedule which is identical to σ except that $K_i = 0$ for all $i = 1, 2, i = 1, 2, \cdots, n$. Let σ' be this new schedule. In σ', let us treat a batch in σ as a job. Then, σ' is an optimal schedule for an instance of problem $F2||C_{max}$ where $p_{1i} = A_i$ and $p_{2i} = B_i$ for $i = 1, 2, \cdots, n$. Let $z^{\sigma'}$ be makespan of σ'. Since σ' is an optimal schedule for the instance of problem $F2||C_{max}$, $z^* \geq z^{\sigma'} + \sum_{i=1}^{n} K_i$. Observe that $z^{\sigma} = z^{\sigma'} + \sum_{i=1}^{n} K_i$ since only difference between σ and σ' is extra processing time K_i for each batch i for $i = 1, 2, \cdots, n$. Contradiction to the assumption that σ is not an

optimal sche-dule. □

The following remark shows that using JR in Step 0 of MJRB instead of MJR may not generate an optimal schedule.

Remark 1: *If JR is used to schedule jobs in batches instead of MJR in MJRB, then MJRB does not guarantee an optimal schedule for problem* $F2||C_{B_{max}}$.

Proof: Consider an instance of problem $F2||\sum C_{B_{max}}$. Let $n=2$, $n_1=2$, and $n_2=1$. Also let $p_{11}=p_{21}=1$, $p_{12}=2$, $p_{22}=1$, $p_{31}=2$, and $p_{32}=3$. If we apply JR to each batch separately, then $A_1=B_1=1$, $A_2=2$, and $B_2=3$. Note that $A_1=B_1$. So, if we apply MJRB to the two batches, then any batch order is possible. If a resulting schedule is $\sigma=(1, 2)$, then the solution value is 8. However, an optimal schedule is $\sigma^*=(2, 1)$ and the solution value is 7 (see Figure 1). □

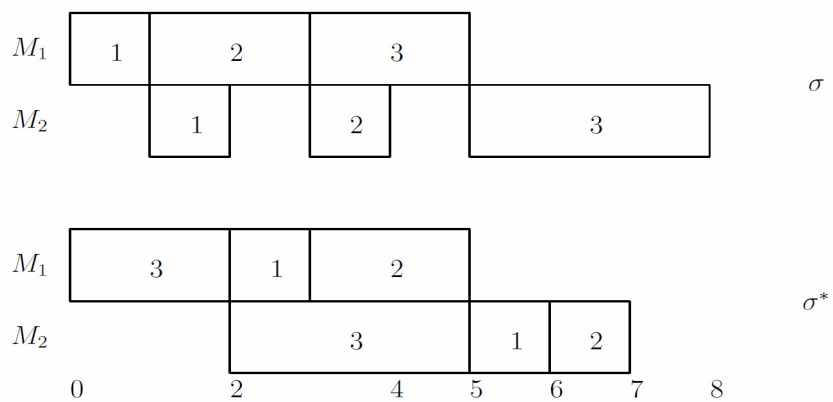

Figure 1. Schedules for Remark 1

5. Problem $F2||\sum C_{B_i}$

5.1 Basic Results

In this section, we present some preliminary results for problem $F2||\sum C_{B_i}$. First, we have the following property for an optimal schedule.

Lemma 2: *For problem* $F2||\sum C_{B_i}$, *there exists an optimal schedule without batch preemption.*

Proof: Follows from an adjacent job interchange argument. □

As a result of Lemma 2, we only consider those scheduled without batch preemption.

5.2 Complexity

In this section, we establish the complexity of problem $F2||\sum C_{B_i}$ and its two special cases. We first establish complexity of problem $F2||\sum C_{B_i}$.

Theorem 3: *The recognition version of problem* $F2||\sum C_{B_i}$ *is unary NP-complete.*

Proof: Note that the recognition version of problem $F2||\sum C_j$ is unary NP-complete (Garey *et al.* [8]). By letting $n_i = 1$ for all $i \in B$, problem $F2||\sum C_j$ reduces to problem $F2||\sum C_{B_i}$. Hence, $F2||\sum C_j$ is a special case of $F2||\sum C_{B_i}$. □

The next two theorems establish complexity of two special cases of problem $F2||\sum C_{B_i}$.

Theorem 4: *The recognition version of problem* $F2|A_i = A|\sum C_{B_i}$ *is unary NP-complete.*

Proof: Note that the recognition version of problem $F2|p_{1j} = p_1|\sum C_j$ is unary NP-complete (Hoogeveen and Kawaguchi [13]). By letting $n_i = 1$ for all $i \in B$, problem $F2|p_{1j} = p_1|\sum C_j$ reduces to problem $F2|A_i = A|\sum C_{B_i}$. Hence, pro-blem $F2|p_{1j} = p_1|\sum C_j$ is a special case of problem $F2|A_i = A|\sum C_{B_i}$. □

The next result establishes that even if B_i is identical for all batches, the problem is still unary NP-complete. We use the reduction from the following unary NP-complete problem.

Numerical Matching with Target Sums (Garey and Johnson [8]): We are given two

disjoint sets of $F = \{f_1, f_2, \cdots, f_q\}$ and $G = \{g_1, g_2, \cdots, g_q\}$, a target vector $(h_1, h_2, \cdots, h_q)$ where $\sum_{i=1}^{q}(f_i + g_i) = \sum_{i=1}^{q} h_i$. The problem is to partition $F \cup G$ into q disjoint sets $T_1, T_2, \cdots, T_q$ such that each T_i contains exactly one element from each of F and G and $f_i + g_i = h_i$ for $i = 1, 2, \cdots, q$.

Theorem 5: *The recognition version of problem* $F2 \mid B_i = B \mid \sum C_{B_i}$ *is unary NP-complete.*

Proof: Given an instance of Numerical Matching with Target Sums, we construct the following instance of $F2 \mid B_i = B \mid \sum C_{B_i}$:

$$
\begin{array}{lcll}
N & = & 6q+7, & \\
n & = & 3q+4, & \\
n_i & = & 2, & i = 1, 2, \cdots, 3q+3 \\
n_{3q+4} & = & 1, & \\
m & = & 2, & \\
p_{1j} & = & 0, & j = 1, 2, \cdots, q, \\
p_{1j} & = & L, & j = q+1, q+2, \cdots, 2q, \\
p_{1j} & = & 0, & j = 2q+1, 2q+2, \cdots, 3q, \\
p_{1j} & = & L, & j = 3q+1, 3q+2, \cdots, 4q, \\
p_{1j} & = & 0, & j = 4q+1, 4q+2, \cdots, 5q, \\
p_{1j} & = & 3L, & j = 5q+1, 5q+2, \cdots, 6q, \\
p_{1j} & = & L, & j = 6q+1, 6q+2, 6q+3, \\
p_{1j} & = & 2L, & j = 6q+4, 6q+5, 6q+6, \\
p_{1,6q+7} & = & 0, & \\
p_{2j} & = & L - g_j, & j = 1, 2, \cdots, q, \\
p_{2j} & = & L, & j = q+1, q+2, \cdots, 2q, \\
p_{2j} & = & L - f_j, & j = 2q+1, 2q+2, \cdots, 3q, \\
p_{2j} & = & L, & j = 3q+1, 3q+2, \cdots, 4q, \\
p_{2j} & = & h_j, & j = 4q+1, 4q+2, \cdots, 5q, \\
p_{2j} & = & L, & j = 5q+1, 5q+2, \cdots, 6q,
\end{array}
$$

$$
\begin{aligned}
p_{2j} &= 2L, && j = 6q+1,\ 6q+2,\ 6q+3,\\
p_{2j} &= L, && j = 6q+4,\ 6q+5,\ \cdots,\ 6q+7,\\
z &= (7.5q^2 + 36.5q + 22)L - \textstyle\sum_{j=1}^{q} f_j - 2\sum_{j=1}^{q} g_j,
\end{aligned}
$$

where $L = 2\max_{1\le j\le q}\{h_j\}+1$, $g_j > 2\sum_{j=1}^{q} f_j$ for $j = 1, 2, \cdots, q$.

The recognition version of problem $F2 \mid B_i = B \mid \sum C_{B_i}$ is in NP since we can calculate $\sum C_{B_i}$ in polynomial time.

We prove that there exists a solution to Numerical Matching with Target Sums if and only if there exists a solution to $F2 \mid B_i = B \mid \sum C_{B_i}$ and $\sum C_{B_i} \le z$.

For each batch, we first apply MJR and determine A_i, B_i, and K_i for $i = 1, 2, \cdots, n$. Since there exists an optimal schedule where batches are scheduled by MJR, we only use the following results to prove the complexity:

$$
\begin{aligned}
A_i &= g_j, && i = 1, 2, \cdots, q,\\
A_i &= f_j, && i = q+1, q+2, \cdots, 2q,\\
A_i &= 3L - h_j, && i = 2q+1, 2q+2, \cdots, 3q,\\
A_i &= L, && i = 3q+1, 3q+2, 3q+3,\\
A_{3q+4} &= 0,\\
B_i &= L, && i = 1, 2, \cdots, 3q+4,\\
K_i &= L - g_i, && i = 1, 2, \cdots, q,\\
K_i &= L - f_i, && i = q+1, q+2, \cdots, 2q,\\
K_i &= h_i, && i = 2q+1, 2q+2, \cdots, 3q,\\
K_i &= 2L, && i = 3q+1, 3q+2, 3q+3,\\
K_i &= 0.
\end{aligned}
$$

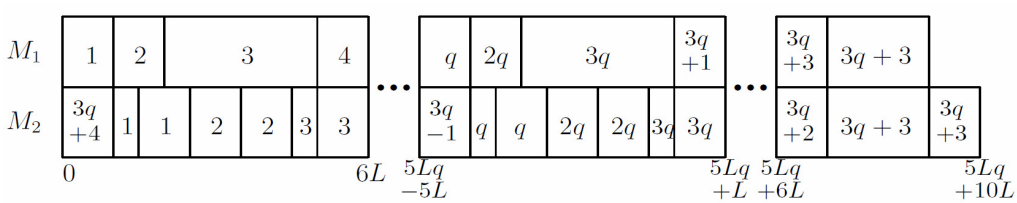

Figure 2. An Example of an Optimal Schedule

($\Rightarrow$) We assume without loss of generality that if there exists a solution to Numerical Matching with Target Sums, then elements are indexed such that $f_i + g_i = h_i$ for $i = 1, 2, \cdots, q$.

Consider the permutation batch schedule σ shown in Figure 2, where

$$\sigma = (3q+2, 1, q+1, 2q+1, 2, q+2, 2q+2, \cdots, q, 2q, 3q, 3q+1, 3q+2, 3q+3) \quad (1)$$

and there exists no inserted idle time. To determine the total completion time, we first calculate total completion time of batches $1, 2, \cdots, 3q$. Since $f_i + g_i = h_i$ for $i = 1, 2, \cdots, q$ by assumption,

$$\sum_{i=1}^{3q} C_i(\sigma) = (7.5q^2 + 21.5q)L - \sum_{j=1}^{q} f_j - 2\sum_{j=1}^{q} g_j. \quad (2)$$

Since completion time of batch $3q$ is $(5q+1)L$, completion time of the batches $3q+1$, $3q+2$, $3q+3$, and $3q+4$ is

$$\sum_{i=3q+1}^{3q+3} C_i(\sigma) = (15q+21)L + L = (15q+22)L. \quad (3)$$

Hence, from (2) and (3), we have

$$\sum_{i=3q+1}^{3q+3} C_i(\sigma) = (7.5q^2 + 36.5q + 22)L - \sum_{j=1}^{q} f_j - 2\sum_{j=1}^{q} g_j. \quad (4)$$

($\Leftarrow$) Let $I_1 \in \{1, 2, \cdots, q\}$, $I_2 = \{q+1, q+2, \cdots, 2q\}$, $I_3 = \{2q+1, 2q+2, \cdots, 3q\}$, and $I_4 = \{3q+1, 3q+2, 3q+3\}$. Since all batches in I_4 are identical, we assume that in an optimal schedule, batches in I_4 are processed in index order. In an optimal schedule, let the batches in I_1, I_2, and I_3 be processed in the order $v_1, v_2, \cdots, v_{3q}$. For notational convenience, let $v_1 = 1, v_2 = 2, \cdots, v_{3q} = 3q$. First, we describe an optimal schedule. Then, we show that a schedule does not have a solution value $\leq z$ unless it is optimal.

We first consider the batch order in an optimal solution. Since batch $3q+4$ has zero processing time on M_1 and $K_i = 0$, it is optimal to process batch $3q+4$ first. Also, batches in I_4 has big processing time on M_2, it will be optimal to process them last. Then, the second batch to process is either from I_1, I_2, or I_3. If we process a batch from I_3 first, then it produces idle time of more than $2L$ on M_2 and also finish later than the other two choices. A batch from I_1 or I_2 will not create any idle time on M_2 but a batch from I_1 will give the smallest completion time. Hence, we process batch 1 from I_1 next.

Now, if we process a batch from I_3 next, then it still creates an idle time on M_2 by $5L-(3L-g_1+h_{2q+1})$ and completion time will be $6L$. Instead, if process either a batch from I_1 or I_2, then it does not create idle time and completion time will be smaller. Suppose we process a batch from I_1 again. Then, completion time of batch 2 will be $4L-g_1-g_2$. Since $h_j < L/2$ for all $j = 1, 2, \cdots, q$, it is optimal to process a batch from I_3 next. Then, completion time of batch $2q+1$ is $6L$ but it produces idle time on M_2 by $5L-(5L-g_1-g_2+h_1)$. Recall that we assume that $g_j > 2\sum_{j=1}^{q} f_j$ for $j = 1, 2, \cdots, q$ and it implies that $h_1 < g_1+g_2$. This idle time at least delay the last three batches by $g_1+g_2-h_1$ compared to the case where there exist no idle time.

Alternatively, suppose we process a batch from I_2 as a third batch instead of an additional batch from I_1. Then, completion time of batch $q+1$ will be $4L-g_1-f_1$. If we process a batch from I_3 next due to the similar reason, then completion time of batch $2q+1$ is $6L$ but it may of may not produce idle time on M_2 because $\Delta = 5L-(5L-g_1-f_1+h_1)$ can be either 0, positive, or negative. Even though completion time of batch $q+1$ will be slightly bigger than that of batch 2, idle time on M_2 or delaying of batch $2q+1$ will be smaller and so increase the solution value by smaller amount. Hence, it should be optimal to process a batch from I_2 as a third batch. Now, we have the similar situation as when batch 1 is about to be scheduled. Hence, it is optimal to schedule batches in I_1, I_2, and I_3 in the order of 2, $q+2$, $2q+2$, 3, $q+3$, $2q+3, \cdots, q, 2q, 3q$.

Further, it is optimal that $f_i + g_{q+i} = h_{2q+i}$ for $i = 1, 2, \cdots, q$. If $f_i + g_{q+i} < h_{2q+i}$ for any $i \in \{1, 2, \cdots, q\}$, then it creates delay of at least $h_{2q+i} - (f_i + g_{q+i})$ for the following

three batches and it will increase the solution value by at least $3\{h_{2q+i}-(f_i+g_{q+i})\}$. Alternatively, if $f_i+g_{q+i}>h_{2q+i}$ for any $i\in\{1, 2, \cdots, q\}$, then it creates idle time of $h_{2q+i}-(f_i+g_{q+i})$ on M_2 before batch $2q+i$ is processed on M_2. Since $\sum_{i=1}^{q}(f_i+g_i)=\sum_{i=1}^{q}h_i$, this idle time will create a situation where $f_l+g_{q+l}<h_{2q+l}$ for some $l\in B$. Hence, eventually it will contribute to increase the solution value at least by $3\{h_{2q+i}-(f_i+g_{q+i})\}$.

Observe that the optimal schedule is the same as the schedule in (4). Hence, the solution value is $\sum_{i=3q+1}^{3q+3}C_i(\sigma)=(7.5q^2+36.5q+22)L-\sum_{j=1}^{q}f_j-2\sum_{j=1}^{q}g_j$. This is only possible if $f_i+g_{q+i}=h_{2q+i}$ for $i=1, 2, \cdots, q$, i e., there exists a solution to Numerical Matching with Target Sums. □

Remark 2: *Problem* $F2\,|\,p_{2j}=p_2\,|\sum C_j$ *is solvable in* $O(nlogn)$ *(Hogeveen and Kawacuchi [13]).*

5.3 Lower Bounds for $F2\,||\sum C_{B_i}$

We establish a series of lower bounds on the value of a schedule for problem $F2\,||\sum C_{B_i}$. The lower bounds are used in the analysis of a heuristic. These bounds are based on the condition that there is no wait for M_2. Before we start introducing the bounds, we let $[i]$ indicate the batch in the *i*th position in given schedule. The first bound assumes that each job is processed as quickly as possible on M_1.

Lemma 3:

$$\sum_{i=1}^{n}C_i\geq nA_{[1]}+\sum_{i=1}^{n}(n-i+1)B_{[i]}.$$

Proof: Since jobs are processed without delaying on M_1,

$$C_{[i]}\geq A_{[1]}+\sum_{k=1}^{i}(K_{[k]}+B_{[k]})$$

for $i\in B$. Since batches are ordered such that $A_i+B_i+2K_i\leq A_{i+1}+B_{i+1}+2K_{i+1}$ for

$i = 1, 2, \cdots, n-1$ (or indexed in shortest processing time first (SPT) order),

$$\sum_{i=1}^{n} C_i \geq nA_{[1]} + \sum_{i=1}^{n}\sum_{k=1}^{j}(K_{[k]} + B_{[k]}) = nA_{[1]} + \sum_{i=1}^{n}(n-i+1)B_{[i]}. \quad \square$$

The next bound assumes that each job is processed as quickly as possible on M_2.

Lemma 4:

$$\sum_{i=1}^{n} C_i \geq \sum_{i=1}^{n}(n-i+1)(A_{[i]} + B_{[i]} + 2K_{[i]}) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i.$$

Proof: Since jobs are processed without delaying on M_2,

$$C_{[i]} \geq \sum_{k=1}^{i}(A_{[k]} + K_{[k]}) + B_{[i]}.$$

Since batches are ordered in SPT order,

$$\sum_{i=1}^{n} C_i \geq \sum_{i=1}^{n}\sum_{k=1}^{i}(A_{[k]} + K_{[k]}) + \sum_{i=1}^{n} B_{[i]} = \sum_{i=1}^{n}(n-i+1)(A_{[i]} + K_{[i]}) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i. \quad \square$$

We combine the two lower bounds in Lemmas 3 and 4 to obtain the following lower bound.

Lemma 5:

$$\sum_{i=1}^{n} C_i \geq \frac{1}{2}\left[\sum_{i=1}^{n}(n-i+1)(A_i + B_i + 2K_i) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i\right].$$

Proof: By adding up two lower bounds from Lemmas 3 and 4,

$$2\sum_{i=1}^{n} C_i \geq \sum_{i=1}^{n}\sum_{i=1}^{n}(n-i+1)(A_{[i]} + B_{[i]} + 2K_{[i]}) + nA_{[1]} + \sum_{i=1}^{n} B_i$$

$$\geq \sum_{i=1}^{n}(n-i+1)(A_i + B_i + 2K_i) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i.$$

The last inequality is due to the fact that batches are in SPT order. □

Let

$$z^L = \sum_{i=1}^{n} C_i \geq \frac{1}{2}[\sum_{i=1}^{n}(n-i+1)(A_i + B_i + 2K_i) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i].$$

We use z^L to provide a lower bound for z^*.

5.4 Heuristic

In this section, we introduce and analyze the worst case behavior of a heuristic. This heuristic is presented in Gonzalez and Sahni [10] to solve problem $F2||\sum C_j$. We apply the heuristic to problem $F2||\sum C_{B_i}$ after we apply MJR first. The heuristic processes batches in SPT order and does not allow any inserted idle time. Consequently, it is a reasonable choice for managers who are interested in maximizing machine utilization.

Heuristic GS

0. Apply Algorithm MJR to each batch *i* for $i = 1, 2, \cdots, n$.
   Calculate A_i, B_i, and K_i for $i = 1, 2, \cdots, n$.
1. Reindex jobs so that $A_i + B_i + 2K_i \leq A_{i+1} + B_{i+1} + 2K_{i+1}$ for $i = 1, 2, \cdots, n-1$.
2. Schedule job 1 such that $C_{11} = A_1 + K_1$ and $C_{21} = A_1 + K_1 + B_1$.
   Schedule batches $2, 3, \cdots, n$ in index order on M_1 and M_2 such that
   $C_{1i} = C_{1,i-1} + A_i + K_i$ and $C_{2i} = \max\{C_{2,i-1}, C_{11}\} + K_i + B_i$ for $j = 2, 3, \cdots, n$.
3. Calculate C_i for $i = 1, 2, \cdots, n$.
   Output $\sum_{i=1}^{n} C_i$ and stop.

Heuristic GS runs in $O(n\log n)$ time. The resulting solution is a permutation schedule where there is no inserted idle time on M_1. Let σ^{GS} be the schedule generated by Heuristic GS and z^{GS} be the cost of schedule σ^{GS}. We analyze Heuristic

GS and find an upper bound on the relative error.

Let

$$\alpha = \min_{i \in B}\{A_i, B_i\},$$
$$\beta = \max_{i \in B}\{A_i, B_i\}.$$

To analyze the worst case behavior of the heuristic, we follow the approach used by Hoogeveen and Kawaguchi [13]. Our analysis is more complicated because we deal with batches instead of jobs.

Lemma 6:

$$|B_k - A_{k+1}| \le (\beta - \alpha)(A_{k+1} + B_{k+1} + 2K_{k+1})/(\beta + \alpha)$$
$$for\ k = 1, 2, \cdots, n-1.$$

Proof: We first consider the case where $|B_k - A_{k+1}| = B_k - A_{k+1}$. Suppose that $|B_k - A_{k+1}| > (\beta - \alpha)(A_{k+1} + B_{k+1} + 2K_{k+1})/(\beta + \alpha)$ for some k. Since $\ge A_k + B_k + 2K_k$, we have $(B_k - A_{k+1})(\beta + \alpha) > (\beta - \alpha)(A_k + B_k + 2K_k)$. Then, $2\alpha B_k > (\beta - \alpha)A_k + 2(\beta + \alpha)A_{k+1} + (\beta - \alpha)2K_k \ge (\beta - \alpha)\alpha + (\beta + \alpha)\alpha + 2(\beta - \alpha)K_k \ge 2\beta\alpha$. Contradiction. The similar argument holds for the other case. □

Theorem 6:

$$z^{GS}/z^* \le 2\beta/(\alpha + \beta).$$

Further, this bound is asymptotically attainable.

Proof: By the construction of GS, $C_1 = A_1 + K_1 + B_1$ and

$$\begin{aligned}
C_i &= \max\{C_{1i}, C_{2,i-1} + K_i\} + B_i \\
&= \max\{C_{1,i-1} + A_i, C_{2,i-1}\} + K_i + B_i \\
&\le \max\{C_{1,i-1} + A_i, \max\{C_{2,i-2} + K_{i-1}, C_{1,i-1}\} + B_{i-1}\} + K_i + B_i \\
&\le \max\{C_{1,i-1}, C_{2,i-2} + K_{i-1}\} + \max\{A_i, B_{i-1}\} + K_i + B_i \\
&\le A_1 + \sum_{k=1}^{i-1}(\max\{B_k, A_{k+1}\} + K_k) + K_i + B_i
\end{aligned}$$

for $i = 2, 3, \cdots, n$. Using the equality $2\max\{B_k, A_{k+1}\} = B_k + A_{k+1} - |B_k - A_{k+1}|$, we have

$$2C_i \le 2A_1 + 2\sum_{k=1}^{i-1}(\max\{B_k, A_{k+1}\} + K_k) + 2(K_i + B_i)$$
$$= \sum_{k=1}^{i} A_k + \sum_{k=1}^{i} B_k + 2\sum_{k=1}^{i} K_k + A_1 + B_i + \sum_{k=1}^{j-1} |B_k - A_{k+1}|$$

for $i \in B$. For notational convenience, we let $W = \sum_{i=1}^{n}(n-i+1)(A_i + B_i + 2K_i)$. Then, we obtain the following inequality by adding up C_i for $i = 1, 2, \cdots, B$.

$$2\sum_{j=1}^{n} C_i \le (W + nA_1 + \sum_{i=1}^{n} B_i) + \sum_{i=1}^{n-1}\sum_{k=1}^{i} |B_k - A_{k+1}|. \tag{5}$$

Now, using Lemma 6, inequality (5) becomes

$$2\sum_{j=1}^{n} C_i \le (W + nA_1 + \sum_{i=1}^{n} B_i) + \{W - n(A_1 + B_1)\}(\beta - \alpha)/(\beta + \alpha)$$
$$\le W\{2\beta/(\beta + \alpha)\} + \sum_{i=1}^{n} B_i + nA_1\{2\alpha/(\beta + \alpha)\}.$$

Note that

$$z^L = \sum_{i=1}^{n} C_i \ge \frac{1}{2}[\sum_{i=1}^{n}(n-i+1)(A_i + B_i + 2K_i) + n\min_{i\in B}\{A_i\} + \sum_{i=1}^{n} B_i]$$
$$= \frac{1}{2}(W + \sum_{i=1}^{n} B_i + \min_{i\in B}\{A_i\}).$$

Then, since $\alpha A_1 \le \beta \min_{i\in B}\{A_i\}$, we have

$$\frac{z^{GS}}{z^*} \le \frac{z^{GS}}{z^L}$$

$$\leq \frac{W(2\beta/(\beta+\alpha)) + \sum_{i=1}^{n} B_i + n\,A_1(2\alpha/(\beta+\alpha))}{W + \sum_{i=1}^{n} B_i + n \min_{i\in B}\{A_i\}}$$

$$\leq \frac{2\beta/(\beta+\alpha)(W + \sum_{i=1}^{n} B_i + n \min_{i\in B}\{A_i\})}{W + \sum_{i=1}^{n} B_i + n \min_{i\in B}\{A_i\}}$$

$$= 2\beta/(\beta+\alpha).$$

Now, we show that this bound is asymptotically attainable. Consider the instance where there are $2n$ batches with processing times $A_i = \beta$, $B_i = \alpha$, and $K_i = 0$ for $i = 1, 2, \cdots, n$ and $A_i = \alpha$, $B_i = \beta$, and $K_i = 0$ for $i = n+1, n+2, \cdots, 2n$. Since $A_i + B_i + 2K_i$ are equal for all batches, any batch sequence an be generated by the heuristic. Suppose that $\sigma^{GS} = (1, 2, \cdots, 2m)$. Then the solution value $z^{GS} = 2\beta n^2 + (\beta + 2\alpha)n$. An optimal schedule $\sigma^* = (n+1, 1, n+2, 2, \cdots, 2n, n)$ has solution value $z^* = (\beta+\alpha)n^2 + (\beta+2\alpha)n$. The relative error goes to $2\beta/(\beta+\alpha)$ as $n \to \infty$. □

6. Discussion and Further Research

We have explored problems $F2||C_{B_{max}}$ and $F2||\sum C_{B_i}$. While there exist several studies in the customer order scheduling problems in parallel and job shop environments, there has not been any research on flowshop environment. For problem $F2||C_{B_{max}}$, we develop an schedule which generates an optimal schedule, and the algorithm is based on JR. For the problem with the objective of minimizing the sum of order completion time, we establish complexity of some special cases. Also, we develop a series of lower bounds and introduce a simple but intuitive heuristic. Then, we find the worst case bound on relative error and the bound is tight as the number of batches goes to infinity.

There are several extensions of our research that might be considered. We can first extend to multiple machine cases where the number of machines are more than 2. Also, we can study different machine speeds such as proportional and unrelated

machines. We also leave this to future research.

References

[1] Ahmadi, R., U. Bagchi, and T. A. Roemer, "Coordinated Scheduling of Customer Orders for Quick Response," *Naval Research Logistics* 52 (2005), 493-512.

[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 Research Logistics* 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] Blocher, J. D. and D. Chhajed, "Minimizing Customer Order Lead-iime in a Two-stage Assembly Supply Chain," *Decision Science* 29 (1998), 951-981.

[6] 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.

[7] 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.

[8] Garey, M. R., D. S. Johnson, and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," *Mathematics of Operations Research* 1 (1976), 117-129.

[9] 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.

[10] Gonzalez, T. and S. Sahni, "Flowshop and job shop schedules: Complexity and approximation," *Operations Research* 26 (1978), 36-52.

[11] 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.

[12] Gupta, J. N. D., J. C. Ho, and A. A. van der Veen, "Single Machine Hierarchical Scheduling with Customer Orders and Multiple Job Classes," *Annals of Opera-*

tions Research 70 (1997), 127-143.

[13] Hoogeveen, J. A. and T. Kawaguchi, "Minimizing total completion time in a two-machine flowshop: analysis of special cases," *Mathematics of Operations Research* 24 (1999), 887-910.

[14] Johnson, S. M., "Optimal Two- and Three-Stage Production with Setup Times Included," *Naval Research Quarterly* 1 (1954), 61-68.

[15] 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.

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

[17] Roemer, T. A. and R. Ahmadi, "Complexity of Scheduling Customer Orders," Working Paper, Anderson School at UCLA (1997), USA.

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

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