Industrial Engineering
& Management Systems
Vol 14, No 1, March 2015, pp.22-31
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2015.14.1.022

Hybrid Flow Shop with Parallel Machines at the First Stage and Dedicated Machines at the Second Stage

Jaehwan Yang*
College of Business Administration, University of Seoul, Seoul, Korea

(Received: September 29, 2014 / Accepted: December 29, 2014)

ABSTRACT

In this paper, a two-stage hybrid flow shop problem is considered. Specifically, there exist identical parallel machines at stage 1 and two dedicated machines at stage 2, and the objective of the problem is to minimize makespan. After being processed by any machine at stage 1, a job must be processed by a specific machine at stage 2 depending on the job type, and one type of jobs can have different processing times on each machine. First, we introduce the problem and establish complexity of several variations of the problem. For some special cases, we develop optimal polynomial time solution procedures. Then, we establish some simple lower bounds for the problem. In order to solve this NP-hard problem, three heuristics based on simple rules such as the Johnson's rule and the LPT (Longest Processing Time first) rule are developed. For each of the heuristics, we provide some theoretical analysis and find some worst case bound on relative error. Finally, we empirically evaluate the heuristics.

Keywords: Two-Stage Hybrid Flow Shop, Computational Complexity, Heuristic, Worst Case Bound on Relative Error

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

1. INTRODUCTION

In this paper, a two-stage hybrid flow shop problem is considered. Specifically, there exist identical parallel machines at stage 1 and two dedicated machines at stage 2, and the objective is to minimize makespan. After being processed by any machine at stage 1, a job must be processed by a specific machine at stage 2 depending on the job type. One type of jobs can have different processing times on each machine. If a job can be processed only by a subset of machines, the environment is called dedicated machines or machine eligibility (Ribas *et al.*, 2010). As in Yang (2013), we say that a machine is *dedicated* to a specific job type if a job which belongs to the particular type can be processed only by the dedicated machine. The job preemption is not allowed, and we assume no setup times between the two stages. Also, all jobs and machines are available at time zero.

A flow shop scheduling problem is called *hybrid* if the problem has more than one machine at least at one stage. We can find numerous real world examples of the hybrid flow shop scheduling problems. An example can be found in some flexible manufacturing systems where each production stage is either a flexible machine or a flexible manufacturing cell (Lee and Variaktarakis, 1994; Zijm and Nelissen, 1990). Another example can be found in the process industry where multiple servers (machines) are available at each stage (Brah and Hun-sucker, 1991). For the further review on general hybrid scheduling problems, see Ribas *et al.*(2010) and Ruiz and Vazquez-Rodriguez (2010).

If there exists single machine at each of the two stages, then the problem is solvable in $O(n \log n)$ by Johnson's algorithm (Johnson, 1954) where n is the number

of jobs. For other cases where either of the two stages has more than one machine, Hoogeveen *et al.* (1996) show that the hybrid flow shop scheduling problems with the objective of minimizing makespan are unary NP-complete both in the preemptive and the non-preemptive case. Regarding the solution procedures, researchers have developed several branch and bound algorithms and heuristic methods for the problems under various conditions (Brah and Hunsucker, 1991; Rajendran and Chaudihari, 1992; Deal and Hunsucker, 1991; Gupta, 1988; Gupta and Tunc, 1991; Gupta, *et al.*, 1997; Hunsucker, 1992).

In this paper, we consider a special type of the hybrid flow shop scheduling problem where there are identical parallel machines at stage 1 and two dedicated machines at stage 2. We can easily find examples where there exist dedicated machines at one of the stages in real world situations. For example, different products go through the same fabrication stage and depending specifications of the products, they are processed on different machines at final assembly. Lin and Liao (2003) consider a label sticker manufacturing company. At stage 1, there exists single high speed machine called calender which is used to glue the surface material and liner together to produce label sticker. Stage 2 has two types of cutting machines depending on cutting pattern required by specifications of the products. Some other examples can be found in process industry such as chemical and pharmaceutical industries.

Riane *et al.* (2002) first considered the hybrid flow shop scheduling problem where there exist two dedicated machines at stage 2 but with only one machine at stage 1. They develop three heuristics which run in polynomial time and one dynamic programming algorithm which runs in exponential time. They empirically evaluate their solution procedures. A similar problem is considered by Cheng *et al.* (2004). However, they assume setup times at stage 1 to switch processing from a job of one type to a job of another type. Each type of jobs need a dedicated machine and one type of jobs have the same processing time. For the case where there exist only two types of jobs, they develop an exact algorithm which runs in polynomial time. Lin and Lio (2003) consider a similar environment with the objective of minimizing weighted maximal tardiness. They develop a heuristic and empirically evaluate the heuristic. Yang (2010) establishes the complexity problem and shows that the decision version of the problem with one machine at stage 1 is unary NP-complete. Also, there exist some researches for problems with more than two dedicated machines at stage 2 but they assume only one machine at stage 1 (Wang and Liu, 2013).

In the next two sections, we introduce notation and present some basic results. Then, we consider some special cases based on the characteristics of the processing time at each stage. For some special cases, we develop optimal polynomial time solution procedures, and for some other cases, we establish their complexity. In Sections 5 and 6, we develop several lower bounds for the problem and three intuitive heuristics for the general case, respectively. The three heuristics are based on the simple rules such as the Johnson's rule and the LPT (Longest Processing Time first). Then, for each of the heuristics, we provide some theoretical analysis, find asymptotically tight worst case bound on relative error for one heuristic, and present some upper bounds for the other heuristics. Finally, we empirically evaluate the heuristics, summarize our work, and discuss the future research direction.

2. NOTATION

The decision variables are

σ = schedule of all jobs
σ_i = schedule of jobs on machine i for $i \in \{1, 2, \cdots, m+1, m+2\}$ where m is the number of machines at stage 1.

Other notation includes

n = number of jobs
m = number of machines at stage 1
N = set of jobs = $\{1, 2, \cdots, n\}$
M = set of machines at stage 1 = $\{1, 2, \cdots, m\}$
M_i = machine i for $i \in M \cup \{m+1, m+2\}$
n_i = number of jobs processed on M_{m+i} for $i = 1,2$
N_i = set of jobs processed on M_{m+i} for $i = 1, 2$
p_{1j} = processing time of job j at stage 1 for $j \in N$
p_{ij} = processing time of job j on M_{m+i-1} for $i = 2, 3$ and $j \in N$
$C_j(\sigma_i)$ = completion time of job j on M_i in σ for $i \in M \cup \{m+1, m+2\}$ and $j \in N$
$C_j(\sigma)$ = completion time of job j in schedule σ for $j \in N$
σ^* = an optimal schedule
z^* = value of optimal schedule σ^*.

Observe that p_{ij} does not exist if job j does not belong to N_i for $i \in M$ and $j \in N$, and $n = n_1 + n_2$. Also, jobs in N_i can only be processed by M_{m+i} for $i \in \{1, 2\}$ at stage 2. When there exists no confusion, we replace $C_j(\sigma)$ and $C_j(\sigma_i)$ with C_j and C_{ij}, respectively. We let $[j]$ indicate the job in the jth position in schedule σ. For example, $p_{1[4]}$ is the processing time on M_1 of the fourth job in schedule σ. We classify our problem according to the standard classification scheme for scheduling problems (Graham *et al.*, 1979). In the three field notation of $\alpha_1 | \alpha_2 | \alpha_3$, α_1 describes the machine structure, α_2 gives the job characteristics and restrictions, and α_3 defines the objective. We extend this scheme to provide two-stage hybrid flow shop scheduling in the α_1 field as suggested by Gupta *et al.* (1997). Hence, we refer to the problem of minimizing makespan in a two-stage hybrid flow shop with arbitrary number of machines at stage 1 and one machine at stage 2 as $F2(P, 1) \| C_{max}$. Since our problem has dedicated machines at stage 1,

we denote the problem as $F2(P, 2d) \| C_{max}$ where "d" at α_1 denotes *dedicated machines*.

A *schedule* defines a job order for each machine, and in this paper, a *permutation schedule* is a schedule in which a sequence of jobs in N_i on any machine at stage 1 is a subsequence of the sequence on M_{m+i} at stage 2 for $i \in \{1, 2\}$. For $F2(P, 2d) \| C_{max}$, the jobs are available at the start of the planning process. Also, no preemptions are allowed.

3. PRELIMINARY RESULTS

In this section, we establish some properties of an optimal schedule. *Inserted idle time* occurs when a machine is intentionally kept idle even if there exists a waiting job. Since there are no restrictions that delay jobs, we have the following result.

Lemma 1. *For problem* $F2(P, 2d) \| C_{max}$, *there exists an optimal schedule without inserted idle time on* M_i *for* $i \in M \cup \{m+1, m+2\}$.

The following lemma establishes that there exists an optimal permutation schedule.

Lemma 2. *For problem* $F2(P, 2d) \| C_{max}$, *there exists an optimal permutation schedule.*

Proof. A simple pairwise interchange argument proves the result. □

As a result of Lemma 2, we only consider a permutation schedule, and hence, a schedule can be described by a job order.

The next result establishes that our problem can be easily solved by solving some other related problem. Consider problem $F2(2d, P) \| C_{max}$ which is a *mirror image* problem of $F2(P, 2d) \| C_{max}$ such that jobs at stage 1 on one problem is jobs at stage 2 on the other problem and vice versa. For convenience, we call this problem M-problem where M stands for *Mirror*. Then, we have the following result.

Lemma 3. *An optimal schedule for problem* $F2(P, 2d) \| C_{max}$ *can be easily obtained from an optimal schedule for its M-problem and vice versa.*

Proof. Consider an optimal schedule σ for problem $F2(2d, P) \| C_{max}$. Also consider an M-problem of problem $F2(2d, P) \| C_{max}$. As a schedule for the M-problem, consider schedule σ' where the sequence of jobs is reversed from σ, jobs at stage 1 are scheduled at stage 2, and jobs at stage 2 are scheduled at stage 1. Also, let σ' have no inserted idle time except for the first jobs at stage 1 so that all jobs are delayed as much as possible without increasing the makespan. Then, it can be seen that σ' is a feasible schedule for the M-problem and makespan of σ' is the same as that of σ.

We now show that σ' is an optimal schedule for the M-problem. Suppose that σ' is not an optimal schedule for the M-problem. Then, there must be a schedule with a smaller makespan for the M-problem and from this schedule, we can create a better schedule for problem $F2(2d, P) \| C_{max}$. This contradicts that σ is an optimal schedule. Therefore, σ' is an optimal schedule for the M-problem. □

Remark 1. *Note that jobs in* σ' *in the previous proof can be expedited so that it can become the schedule without any inserted idle time.*

4. COMPLEXTY AND SPECIAL CASES

In this section, we establish complexity of the problem under various conditions and examine some special cases. First, the general version of problem $F2(P, 2d) \| C_{max}$ is unary NP-complete since $F2(1, 2d) \| C_{max}$ is unary NP-complete (Yang, 2010). Recall that the Johnson's rule generates an optimal schedule for problem $F2 \| C_{max}$ (Johnson, 1954). We begin by the following two small theorems which establish complexity of some special cases of the problem.

Theorem 1. *If* $m = 2$ *and* $N_1 (or\, N_2) = \varnothing$, *then the problem is still unary NP-complete.*

Proof. If $m = 2$ and $N_1 (or\, N_2) = \varnothing$, then the problem becomes $F2(P2, 1) \| C_{max}$, and problem $F2(P2, 1) \| C_{max}$ is unary NP-complete (Hoogeveen *et al.*, 1996). □

Theorem 2. *If* $p_{m+1,j} = p_{m+2,j} = 0$ *for all* $j \in N$ *and there exists a fixed number of machines at stage 1, then the problem is binary NP-complete.*

Proof. If $p_{m+1,j} = p_{m+2,j} = 0$ for all $j \in N$, then the problem is identical to $Pm \| C_{max}$. Hence, the result holds. □

Theorem 3. *If* $p_{m+1,j} = p_{m+2,j} = p$ *for all* $j \in N$ *and* $p \leq \min_{j \in N} \{p_{1,j}\}$ *and there is a fixed number of machines at stage 1, then the problem is binary NP-complete.*

Proof. Since $p \leq \min_{j \in N} \{p_{1j}\}$, any schedule which generates the shortest completion at stage 1 is optimal. Hence, the problem is identical to $Pm \| C_{max}$ and hence, the result holds. □

We now introduce a simple algorithm which generates an optimal schedule for a special case of problem $F2(2d, P) \| C_{max}$ where processing times at stage 2 are identical.

Algorithm A1

1. Sort jobs in N_i in an increasing order of their processing time at stage 1 for $i = 1, 2$.
   Schedule the jobs on their dedicated machines at

stage 1.
2. Schedule jobs on the first available machine at stage 2 in their completion time order at stage 1 without any inserted idle time.
3. Calculate and output C_{max}, and stop.

Algorithm A1 runs in $O(n \log n)$ time. The resulting solution is a permutation schedule where there is no inserted idle time on any machine.

Theorem 4. *For problem $F2(2d, P) \| C_{max}$, if processing times at stage 2 are identical, then algorithm A1 generates an optimal schedule.*

Proof. Observe that the first job scheduled by Algorithm A1 generates the smallest completion time since it schedules jobs in an increasing order of their processing time at stage 1 and all processing times at stage 2 are identical. Also, observe that *k*th job at stage 2, which is selected to be scheduled on M_i for $i \in \{1, 2\}$ should have the smallest completion time at stage 2 in any partial schedule of k jobs in N for $k \in \{1, 2, \cdots, n\}$ since the A1 schedules jobs at stage 1 in an increasing order of their processing time and it also schedules jobs at stage 2 in their completion time order at stage 1 without inserted idle time. This implies that the A1 generates the smallest completion time for *n*th job, and therefore, the A1 generates an optimal schedule for the given problem. □

Corollary 1. *For problem $F2(P, 2d) \| C_{max}$, if processing times at stage 1 are identical, then it is solvable in a polynomial time.*

Proof. An M-problem for $F2(P, 2d) \| C_{max}$ can be constructed and this problem can be solvable in polynomial time by using Algorithm A1. By reversing the job sequence and switching stages of jobs, an optimal schedule for problem $F2(P, 2d) \| C_{max}$ can be obtained from Lemma 3. □

5. LOWER BOUNDS

We establish several lower bounds on the value of a schedule. These lower bounds are used in the analysis of the heuristics and the computational experiment. For notational convenience, we define some useful notation. Let $P_1 = p_{11} + p_{12} + \cdots + p_{1n}$ and $P_i = p_{i1} + p_{i2} + \cdots + p_{in}$ where $p_{ij} > 0$ only if job $j \in N_{i-1}$ for $i = 2, 3$. All bounds are based on the condition that there is no wait for M_{m+1} and M_{m+2}. The first bound assumes that each job is processed as quickly as possible on M_{m+1} and M_{m+2}.

Lemma 4.

$$C_{max} \geq z^{L1} = \max\{\min_{j \in N_1}\{p_{1j}\} + P_2, \min_{j \in N_2}\{p_{1j}\} + P_3\}.$$

Proof. For job $j \in N_i$,

$$C_{[j]} \geq \min_{j \in N_i}\{p_{1j}\} + \sum_{k=1}^{j} p_{m+i[k]}$$

where $p_{m+i[k]} = 0$ if *k*th job is not processed on M_{m+i} for $i \in \{1, 2\}$. Then, the result follows. □

The next bound assumes that each job is processed as quickly as possible on M_i for $i \in M$.

Lemma 5.

$$C_{max} \geq z^{L2} = \{\frac{\sum_{j=1}^{n} p_{1j}}{m} + \min\{\min_{j \in N_1}\{p_{2j}\}, \min_{j \in N_2}\{p_{3j}\}\}.$$

Proof. The last job at stage 1 should complete no sooner than $\sum_{j=1}^{n} p_{1j}/m$. Then, the result follows. □

Finally, the next bound assumes that an optimal makespan is no smaller than the processing time of the biggest job.

Lemma 6.

$$C_{max} \geq z^{L3} = \max_{j \in N}\{p_{1j} + p_{m+1,j} + p_{m+2,j}\}.$$

Proof. The optimal makespan is no smaller than the biggest job in *N*. Then, the result follows. □

We use z^{L1}, z^{L2}, and z^{L3} to bound z^* from below.

6. HEURISTICS

In this section, we introduce three intuitive heuristics which are all based on the simple scheduling rules such as the Johnson's rule, the LPT rule and the List Scheduling (LS) rule. Note that the LS rule schedules the first available job on the first available machine.

First, the two heuristic procedures we develop use the following well known rule called Johnson's rule (Johnson, 1954), which is optimal for problem $F2 \| C_{max}$. The following is a brief description of the Johnson's rule (Lee and Vairaktarakis, 1994).

Johnson's Rule: Job *i* precedes job *j* if $\min\{p_{1i}, p_{2j}\} \leq \min\{p_{1j}, p_{i2}\}$.

We begin by describing the first heuristic, Heuristic H1. The H1 is simple and intuitive and uses the Johnson's rule. Furthermore, it has the asymptotically tight worst case bound on relative error of 3-2/*m*, which seems to be the best known bound for the problem $F2(P2d, 1) \| C_{max}$. We formally describe the H1 as follows.

Heuristic H1
1. Apply the Johnson's rule for jobs in each of N_1 and N_2 separately, and schedule jobs in N_i on M_i and M_{m+i}

for $i = 1, 2$.
Let σ^i be a schedule for N_i for $i = 1, 2$.
2. Find the first available machine among M, and let M_k and T_k be this machine and the earliest time M_k is available, respectively.
3. Find the first available job, job r which starts later than T_k at stage 1.
If there is no such job, then go to Step 4. Otherwise, schedule job r on M_k and adjust the remaining schedule without changing the order of jobs on M_{m+i} so that there is no inserted idle time for $i = 1, 2$.
4. Calculate and output C_{max}, and stop.

Heuristic H1 runs in $O(n \log n)$ time. The resulting solution is a permutation schedule where there is no inserted idle time on any machine. Let σ^{H1} be the schedule generated by the H1 and z^{H1} be the cost of schedule σ^{H1}.

The next heuristic uses a simple scheduling called the LPT to determine the scheduling order of all jobs. The LPT rule schedules a job with the longest processing time on the first available machine, and it is well known to generate a good approximate schedule in a parallel machine environment.

Heuristic LP
1. Determine the scheduling order of the jobs at stage 1 according to the LPT rule on processing time of jobs at stage 1 without any inserted idle.
2. Schedule jobs at stage 2 in the completion order jobs at stage 1.
3. Calculate and output C_{max}, and stop.

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

Finally, we present Heuristic H2 which also uses the Johnson's rule to determine the sequence of jobs in N_i for $i \in \{1, 2\}$. Then, the H2 assumes that jobs on M_{m+1} and M_{m+2} complete at the same time to determine job sequence at stage 1. Notice that the H2 generates the same optimal solution as the Algorithm A1, which is to solve problem $F2(2d, P) \| C_{max}$, if processing times at stage 2 are identical. Hence, we can consider the H2 as an extended version of the Algorithm A1 for problem $F2(P, 2d) \| C_{max}$.

Heuristic H2
1. Let $T = \max\{P_1 + P_2, P_1 + P_3\}$ which is an upper bound for C_{max}.
2. Apply the Johnson's rule for jobs in each of N_1 and N_2 separately, and schedule jobs in N_i on M_{m+i} for $i = 1, 2$ without inserted idle time except for the first job on each machine such that the last job on each machine completes at time T.
3. Schedule jobs at stage 1 backward, i.e., starting from the last job scheduled on either M_{m+1} or M_{m+2}, such that waiting time of each job between the two stages can be minimized.
4. Expedite all jobs such that the first jobs on M_i for $i \in M$ starts at time 0 and there exists no inserted idle time.
5. Calculate and output C_{max}, and stop.

Heuristic H2 runs in $O(n \log n)$ time. The resulting solution is a permutation schedule where there is no inserted idle time on any machine. Let σ^{H2} be the schedule generated by the H2 and z^{H2} be the cost of schedule σ^{H2}.

7. ANALYSIS OF HEURISTICS

In this section, we analyze the heuristics for the general case of the problem. We first analyze heuristic H1, and for convenience, we define some notation. For a schedule σ, let $I_j(\sigma)$ be the idle time on M_{m+i} before job j and after the previous job in N_i, if any, for any $j \in N_i$. Also, we notify that the following analysis on the H1 is based on one of the analysis procedures in Chen (1995). The following lemma establishes that the idle time of a job is no larger than the job's processing time at stage 1.

Lemma 7. *For any* $j \in N$. $I_j(\sigma^{H1}) \le p_{1j}$.

Proof. Observe that schedule σ which is generated by Step 1 of the H1 always generates a smaller idle time for each job than σ^{H1} does. Hence, we can prove this lemma by using σ. Note that the Johnson's rule generates a permutation schedule and there is no inserted idle time on M_{m+i} for $i \in \{1, 2\}$. Hence, $I_j(\sigma) \le p_{1j}$. □

The following result establishes the asymptotically tight for the H1.

Theorem 5. $z^{H1}/z^* \le 3 - 2/m$, *and the bound is asymptotically tight.*

Proof. Without loss of generality, we suppose that $C_{max}(\sigma_{m+1}) \ge C_{max}(\sigma_{m+2})$. If $I_j(\sigma^{H1}) = 0$ for all $j \in N_1$, then σ^{H1} does not have any idle time on M_{m+1}. Hence, it can be seen that $C_{max}(\sigma^{H1}) \le 2C^*_{max}$, and the result holds. Thus, we suppose $I_j(\sigma^{H1}) > 0$ for some $j \in N_1$.

We let $N_1 = \{1, 2, \cdots, n_1\}$ and reindex jobs in N_1 such that jobs in σ^{H1}_{m+1} is scheduled in their index order. Then, $N_2 = \{n_1 + 1, n_1 + 2, \cdots, n\}$. Further, we let $k = \max\{j \in N_1 : I_j(\sigma^{H1}) > 0\}$, and $Q_1 = \{j \in N_1 : p_{1j} < p_{m+1,j}\}$ and $Q_2 = N \setminus Q_1$. Then, we consider the following two cases.

Case 1. $k \in Q_1$.

Since jobs are first scheduled by the Johnson's rule, $\{1, 2, \cdots, k\} \subseteq Q_1$. Then, we have $I_j(\sigma^{H1}) \le p_{1j} < p_{m+1,j}$ from

Lemma 7. Thus,

$$\begin{aligned} C_{max}(\sigma^{H1}) &= \sum_{j\in N_1}(I_j(\sigma^{H1}) + p_{m+i,j}) \\ &= \sum_{I_j(\sigma^{H1})>0} I_j(\sigma^{H1}) + \sum_{j\in N_1} p_{m+i,j} \\ &\le \sum_{j=1}^{k} p_{1i} + \sum_{j\in N_1} p_{m+i,j} \le 2\sum_{j\in N_1} p_{m+1,j} \\ &\le 2C^*_{max} \le (3-2/m)C^*_{max}. \end{aligned}$$

Case 2. $k \in Q_2$.

Since jobs are first scheduled by the Johnson's rule, $\{k+1, k+2, \cdots, n_1\} \subseteq Q_2$. Let T be the starting time of job k at stage 1. Let R be a set of jobs in N which start earlier than job k at stage 1. Then, $\sum_{j\in R} p_{1j} \ge mT$. Since job $k \in Q_2$ and from Lemma 5, we have

$$C^*_{max} \ge \frac{\sum_{j\in N} p_{1j}}{m} \ge T + \frac{p_{1k} + \sum_{j=k+1}^{n_1} p_{m+1,j}}{m}.$$

Observe that $C_{max}(\sigma^{H1}) = T + p_{1k} + p_{m+1,k} + \sum_{j=k+1}^{n_1} p_{m+1,j}$. Hence,

$$\begin{aligned} &C_{max}(\sigma^{H1}) - C^*_{max} \\ &\le p_{1k} + p_{m+1,k} + \sum_{j=k+1}^{n_1} p_{m+1,j} - \frac{p_{1k} + \sum_{j=k+1}^{n_1} p_{m+1,j}}{m} \\ &= p_{m+1,k} + \frac{(m-1)(p_{1k} + \sum_{j=k+1}^{n_1} p_{m+1,j})}{m} \\ &= p_{m+1,k} - \frac{(m-1)(p_{m+1,k} + \sum_{j=1}^{k} p_{m+1,j})}{m} \\ &\quad + \frac{(m-1)(p_{1k} + p_{m+1,k} + \sum_{j\in N} p_{m+1,j})}{m} \\ &\le \frac{(m-1)(p_{1k} + p_{m+1,k} + \sum_{j\in N} p_{m+1,j})}{m} \le \frac{2(m-1)C^*_{max}}{m}. \end{aligned}$$

In order to prove that the bound is asymptotically tight, consider the instance where there are n jobs, $|N_1| = n$, and $|N_2| = 0$. For the case where $m = 2$, we can construct the instance where $p_{1,1} = n$, $p_{1,2} = p_{1,3} = \cdots = p_{1,n} = 1$, and $p_{3,1} = p_{3,2} = \cdots = p_{3,n} = 1$. The smallest processing time of any job is 1 and processing time for all jobs on M_3 is 1. Hence, any job sequence can be a schedule by the H1. Then, $\sigma^{H1} = (1, 2, \cdots, n)$, and the solution value is $z^{H1} = 2n$. An optimal schedule is $\sigma^* = (2, 3, \cdots, n, 1)$ where job 1 is scheduled on M_1 and the rest of jobs are scheduled on M_2, and the solution value is $z^* = n+1$. The relative error, $z^{H1}/z^* = 2n/(n+1)$ goes to $3 - 2/m = 2$ as $n \to \infty$. Therefore, the bound is asymptotically tight for the case where $m = 2$.

For the case where $m > 2$, the construction of an instance is complicated and we use the instance suggested by Chen (1995). Note that the instance in Chen (1995) is for problem $F2(P, 1)\,||\,C_{max}$. As an example, we present the instance where $m = 3$. Consider the instance where there are $n = 3k + 5$ jobs for $k \ge 2$, $|N_1| = n$, and $|N_2| = 0$. Also, $p_{1,1} = 0$, $p_{1,2} = p_{1,3} = p_{1,4} = k$, $p_{1,5} = 3k$, $p_{1,6} = p_{1,7} = \cdots = p_{1,n} = 1$, and $p_{4,1} = p_{4,2} = \cdots = p_{4,n} = 1$. As in the previous example where $m = 2$, any job sequence can be a schedule by the H1. Then, $\sigma^{H1} = (1, 2, \cdots, n)$, and the solution value is $z^{H1} = k + 3k + (n - m - 1) = 4k + (n - 4) = 7k + 1$. An optimal schedule is $\sigma^* = (1, 5, 6, 7, 8, \cdots, 3k + 4, 3k + 5, 2, 3, 4)$ where job 3 and 5 are scheduled on M_1, jobs 2 and 3 are scheduled on M_2, job 4 is scheduled on M_3 and the rest of jobs are scheduled on M_2 and M_3 so that jobs on M_1, M_2, and M_3 are finished at $4k$, and the solution value is $z^* = n = 3k + 5$. The relative error, $z^{H1}/z^* = (7k + 1)/(3k + 5)$ goes to $3 - 2/m = 7/3$ as $n \to \infty$. Therefore, the bound is asymptotically tight for the case where $m = 3$. □

Next, we analyze the LP, and the following theorem establishes that heuristic LP has the worst case bound of $7/3 - 1/(3m)$.

Theorem 6. $z^{LP}/z^* \le 7/3 - 1/(3m)$.

Proof. Note that for problem $P\,||\,C_{max}$, $C_{max}(LPT)/C_{max}(OPT) = 4/3 - 1/(3m)$ (Graham, 1969) where $C_{max}(LPT)$ and $C_{max}(OPT)$ are solution values of the LPT schedule and an optimal schedule, respectively.

Without loss of generality, suppose that the last job at stage 1 completes on M_1. Then, we have

$$C_{max}(\sigma_1^{LP}) \le (\frac{4}{3} - \frac{1}{3m})C^*_{max}.$$

After completion of the last job at stage 1, there should not be any idle time at stage 2. Hence,

$$C_{max}(\sigma^{LP}) \le C_{max}(\sigma_1^{LP}) + \max\{P_{m+1}, P_{m+2}\}.$$

Then, from Lemma 4,

$$\begin{aligned} C_{max}(\sigma^{LP}) &\le C_{max}(\sigma_1^{LP}) + z^1 \\ &\le (\frac{4}{3} - \frac{1}{m})C^*_{max} + C^*_{max} \\ &\le (\frac{7}{3} - \frac{1}{m})C^*_{max}. \end{aligned}$$ □

Remark 2. *There exists an instance of problem* $F2(P, 2d)\,||\sum C_j$ *whose bound on* $z^{LP}/z^* = 2 - 1/(3m)$.

Proof. We first consider the case where m is an even

number. Consider the instance where there are $n = 2m+1$ jobs with processing times $p_{1,2j-1} = p_{1,2j} = 2m - j$ for $j = 1, 2, \cdots, m/2$, $p_{1,m+2j-1} = p_{1,m+2j-1} = m + j$ for $j = 1, 2, \cdots, m/2-1$, $p_{1,2m-1} = p_{1,2m} = p_{1,2m+1} = m$, $p_{m+1,j} = 0$ for $j = 1, 2, \cdots, 2m$, $p_{m+1,2m+1} = 2m$. In an optimal schedule, jobs $2m-1$, $2m$, $2m+1$ are scheduled on the same machine at stage 1 so that the sum of their processing time is $3m$. Similarly, one job with processing time $2m - j$ and another job with processing time $m + j$ are processed on the same machine at stage 1 for $j = 1, 2, \cdots, m/2-1$. Also, two jobs with processing time $2m - m/2$ are processed on the same machine at stage 1. Note that on each machine, two jobs are scheduled and the total processing time on each machine is $3m$. Finally, we assume that job $2m+1$ is the first job to be scheduled. Then, the optimal solution value is $z^* = 3m$. In a schedule by the LP, one job with processing time $2m - j$ and another job with processing time $m + j - 1$ is processed on the same machine at stage 1 so that the sum of their processing time is $3m-1$. On each machine, schedule a bigger job first so that it complies to the LPT rule. Furthermore, we assume without loss of generality that one remaining job which is scheduled last in the schedule is job $2m+1$. Then, the solution value is $z^{LP} = 3m-1+3m = 6m-1$. The relative error is $(6m-1)/(3m) = 2-1/(3m)$. An instance where m is an odd number can be similarly constructed. □

Remark 3. *The worst case error bound for the H1, $3-2/m$ is less than that for the LP, $7/3-1/(3m)$ only if $m < 2.5$. Hence, for the case where $m \geq 3$, the LP performs better than the H1 in terms of the worst case bound.*

The following theorem establishes that heuristic H2 has the worst case bound of $3-1/m$. Recall that the LS rule schedules the first available job on the first available machine.

Theorem 7. $z^{H2}/z^* \leq 3-1/m$.

Proof. Note that for problem $P \| C_{max}$, $C_{max}(LS)/C_{max}(OPT) = 2-1/m$ (Graham, 1969) where $C_{max}(LS)$ and $C_{max}(OPT)$ are solution values of the LS schedule and an optimal schedule, respectively.

Without loss of generality, suppose that the last job at stage 1 completes on M_1. Then, we have

$$C_{max}(\sigma_1^{LS}) \leq (2-\frac{1}{m})C_{max}^*.$$

After completion of the last job at stage 1, there should not be any idle time at stage 2. Hence,

$$C_{max}(\sigma^{LS}) \leq C_{max}(\sigma_1^{LS}) + \max\{P_{m+1}, P_{m+2}\}.$$

Then, from Lemma 4,

$$\begin{aligned} C_{max}(\sigma^{LS}) &\leq C_{max}(\sigma_1^{LS}) + z^{L1} \\ &\leq (2-\frac{1}{m})C_{max}^* + C_{max}^* \\ &\leq (3-\frac{1}{m})C_{max}^*. \ \square \end{aligned}$$

8. COMPUTATIONAL STUDY

To study the practical value of the procedures we develop, heuristics H1, LP, and H2 are empirically evaluated. Because finding z^* is computationally intensive, we use lower bound value $\max\{z^{L1}, z^{L2}, z^{L3}\}$ for the experiments from Lemmas 4, 5, and 6. For notational convenience, let $\overline{z}^L = \max\{z^{L1}, z^{L2}, z^{L3}\}$. The performance indicators for H1, LP, and H2 are the upper bounds on the relative errors, $(z^{H1} - \overline{z}^L)/\overline{z}^L$, $(z^{LP} - \overline{z}^L)/\overline{z}^L$, and $(z^{H2} - \overline{z}^L)/\overline{z}^L$, respectively.

This computational study compares the performance of LP, H1, and H2 under various conditions and examines the effects of various factors such as m, n, n_1, n_2, and p_{ij}. For each problem instance, $p_{ij} : DU[p^{LB}, p^{UB}]$ for $i \in M$ and $j \in N$ where p^{LB} and p^{UB} are parameters and where $DU[\ell, u]$ is a discrete random variable uniformly distributed between ℓ and u. For a given set of test problems, n, n_1, n_2, and m are fixed.

We generate 1,080 test problems under 36 conditions. To test the effects of varying the number of jobs n, three different values of n are considered: 10, 50, and 100. To determine the effect of the gap between n_1 and n_2, we consider three different combinations of n_i's for each value of n. They are $n_1 = 0.5n$, $n_1 = 0.6n$, and $n_1 = 0.7n$. For these cases, we fixed m as 2. Finally, in order to test the effects of different number of machines, three values of m are considered: 2, 5, 10. For these cases, we fixed n as 50.

It is also possible that the standard deviation of the p_{ij}'s may affect heuristic performance. Consequently, we consider three different distributions for p_{ij}, $i \in \{1, 2\} : p_{ij} : DU[1, 99]$, $p_{ij} : DU[25, 75]$, and $p_{ij} : DU[40, 60]$. The standard deviations are 28.88, 14.43, and 5.77, respectively.

The mean relative error is the arithmetic mean of the ratio of $(z^H - \overline{z}^L)$ to $\overline{z}^L$ where z^H is the value of the heuristic. The mean relative error is calculated over 30 instances of each problem type. The program is implemented in C and run on a desktop PC with double processors with 2GHz for each plus 6GB RAM.

In Table 1, we present the mean relative error when n = 10 while m is fixed as 2. The result indicates that the H2 performs best and the LP performs worst. This is interesting because in terms of the worst case bound, the LP is the best heuristic and the H2 is the worst one according to our analysis of the heuristics in the previous section. As the standard deviation of p_{ij} increases, the average relative error bound increases for all heuristics. With a given standard deviation of p_{ij}, as $n_1 - n_2$ dec-

Table 1. Heuristic Performance when $n = 10$ and $m = 2$ (*out of 30 problems)

$n = 10$		Mean relative error			# of best performance*		
p_{ij} :	(n_1, n_2)	LP	H1	H2	LP	H1	H2
	(5, 5)	0.30126	0.06496	0.04009	0	14	25
DU[1, 99]	(6, 4)	0.24952	0.05926	0.03839	1	17	22
	(7, 3)	0.24356	0.02790	0.02149	1	21	20
	(5, 5)	0.26233	0.00415	0.00056	0	21	29
DU[25, 75]	(6, 4)	0.20087	0.00130	0.00067	0	25	29
	(7, 3)	0.12518	0.00040	0.00019	0	28	28
	(5, 5)	0.24900	0.00207	0.00024	0	21	30
DU[40, 60]	(6, 4)	0.16488	0.00001	0.00004	0	29	29
	(7, 3)	0.08694	0.00001	0.00003	0	29	29

Table 2. Heuristic Performance when $n = 50$ and $m = 2$ (*out of 30 problems)

$n = 50$		Mean relative error			# of best performance*		
p_{ij} :	(n_1, n_2)	LP	H1	H2	LP	H1	H2
DU[1, 99]	(25, 25)	0.17312	0.04646	0.03092	0	12	24
	(30, 20)	0.13675	0.04074	0.01943	0	16	25
	(35, 15)	0.11115	0.01103	0.00393	0	21	26
DU[25, 75]	(25, 25)	0.16793	0.00459	0.00184	0	22	27
	(30, 20)	0.08986	0.00070	0.00076	0	29	29
	(35, 15)	0.04206	0.00000	0.00064	0	30	29
DU[40, 60]	(25, 25)	0.15187	0.00380	0.00106	0	19	30
	(30, 20)	0.06343	0.00001	0.00002	0	29	29
	(35, 15)	0.02626	0.00001	0.00002	0	29	29

Table 3. Heuristic Performance when $n = 100$ and $m = 2$ (*out of 30 problems)

$n = 100$		Mean relative error			# of best performance*		
p_{ij} :	(n_1, n_2)	LP	H1	H2	LP	H1	H2
DU[1, 99]	(50, 50)	0.11954	0.02049	0.01757	0	17	21
	(60, 40)	0.05429	0.01381	0.00113	0	15	27
	(70, 30)	0.04507	0.00263	0.00090	0	23	27
DU[25, 75]	(50, 50)	0.11428	0.00409	0.00220	0	20	23
	(60, 40)	0.03279	0.00058	0.00037	0	29	28
	(70, 30)	0.01408	0.00061	0.00032	0	29	28
DU[40, 60]	(50, 50)	0.09921	0.00384	0.00120	0	16	29
	(60, 40)	0.02444	0.00000	0.00001	0	30	29
	(70, 30)	0.00824	0.00000	0.00001	0	30	29

reases, the mean relative error bound seems to increase for most of the cases. Notice that some portion of the error is due to using a lower bound for z^*. Hence, the performance of the heuristics is likely to be better than those indicated in the tables. Similar patterns can be found in Tables 2 and 3 where $n = 50$ and $n = 100$, respectively.

In comparison of the cells from Tables 1, 2 and 3, it seems that the performance of the heuristics drastically improves as n increases. This improvement is consistent for the three heuristics.

The last three columns in Tables 1, 2 and 3 indicate that the H2 is overall the best choice among the heuristics compared when $n = 10$ and $n = 50$. In only six cases out of 27 cases, the H1 performs slightly better than the H2. However, it seems noticeable the perfor-

mance of the H1 seems as good as that of the H2 or it is slightly better than the H2 when n = 100. The heuristic LP is the by far the worst heuristic among the heuristics compared even though it has the best worst case bound when $m \geq 3$. Based on the comparison of the three heuristics, it seems that the using Johnson's rule is critical when a simple heuristic is developed for the problem.

In Table 4, we present the mean relative error with varying number of machines while n is fixed as 50 and $p_{ij} : DU[1, 99]$. The results indicate that the performance of the three heuristics improves as the number of machines increases. However, the improvement in performance of the LP is small compared to the other two heuristics. Finally, it seems that the performances of the H1 and the H2 are almost identical when m = 5 and m = 10.

Since the computation time of each of the heuristics is negligible, for each instance managers in real worlds may choose the best heuristic by comparing the results of all three heuristics.

9. SUMMARY AND FURTHER RESEARCH

In this paper, we consider the hybrid flow shop scheduling problem where there exist identical parallel machines at stage 1 and two dedicated machines at stage 2. We establish complexity for some of the special cases of the problem, and for other special cases, we develop optimal solution procedures which run in polynomial time. The three heuristics based on the simple rules such as the Johnson's rule and the LPT rule are examined. For each of the heuristics, we provide some theoretical analysis and find a worst case bound on relative error. Especially, we extend heuristic H, which is for the case with identical parallel machines at stage 1 and one machine at stage 2 by Chen (1995), for the case where there exist identical parallel machines at stage 1 and two dedicated machines at stage 2, and we find an asymptotically tight bound on relative error. Then, we develop the two additional heuristics which are based on simple scheduling rules, the LPT rule and the Johnson's rule, respectively. Finally, the heuristics are empirically evaluated.

We summarize our work in Table 5. A blank in the number of machines column implies arbitrary number machines. Also, a blank in the Job Characteristics column implies that there are no job restrictions. We also list the theorem or reference where each complexity is found.

The problem has several important practical implications for the real-world situations. But, so far little research has been done for the problem. The worth considering are more general cases of the problem such as including different objective functions. Also, heuristics must be improved and their performance should be studied.

Table 4. Heuristic Performance with Varying # of Machines (*out of 30 problems)

n = 50, $p_{ij} : DU[1, 99]$		Mean relative error			# of best performance*		
m	(n_1, n_2)	LP	H1	H2	LP	H1	H2
2	(25, 25)	0.17312	0.04646	0.03092	0	12	24
	(30, 20)	0.13675	0.04074	0.01943	0	16	25
	(35, 15)	0.11115	0.01103	0.00393	0	21	26
5	(25, 25)	0.07170	0.00012	0.00003	0	29	30
	(30, 20)	0.06400	0.00002	0.00048	0	29	28
	(35, 15)	0.05488	0.00002	0.00018	0	29	28
10	(25, 25)	0.06883	0.00012	0.00003	0	29	30
	(30, 20)	0.06390	0.00002	0.00048	0	29	28
	(35, 15)	0.05488	0.00002	0.00018	0	29	28

Table 5. Complexity of Two-stage Hybrid Flow Shop with Identical Parallel Machines at Stage 1 and Two Dedicated Machines at Stage 2

# of Mach.	Job Characteristics	Complexity Results	
		Unary NP-comp.	Yang (2010)
	$N_1 (or\ N_2) = \varnothing$	Unary NP-comp.	Thm. 1
m	$p_{m+1,j} = p_{m+2,j} = 0,\ \forall j \in N$	Binary NP-comp.	Thm. 2
m	$p_{m+1,j} = p_{m+2,j} = p,\ \forall j \in N$, $p \leq \min_{j \in N}\{p_{1j}\}$	Binary NP-comp.	Thm. 3
	$p_{1j} = p_1$ for all $j \in N$	$O(n \log n)$	Cor. 1

ACKNOWLEDGMENT

This work was supported by the 2013 Research Fund of the University of Seoul.

REFERENCES

Brah, S. A. and Hunsucker, J. L. (1991), Branch and bound algorithm for a flowshop within multiple processors, *European Journal of Operational Research*, **51**, 88-99.

Chen, B. (1995), Analysis of Classes of Heuristics for Scheduling a Two-Stage Flow Shop with Parallel Machines at One Stage, *Journal of the Operational Research Society*, **46**, 23-244.

Cheng, T. C. E., Kovalyov, M. Y., and Chakhlevich, K. N. (2012) Batching in a two-stage flowshop with dedicated machines in the second stage, *IIE Transactions*, **36**, 87-93.

Deal, D. E. and Hunsucker, J. L. (1991), The two-stage flowshop scheduling problem with M machines at each stage, *Journal of Information and Optimization Sciences*, **12**, 407-417.

Graham, R. L. (1969), Bounds on Multiprocessing Timing Anomalies, *SIAM J. Appl. Math.*, **17**, 416-429.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy, Kan A. H. G. (1979), Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, **5**, 287-326.

Gupta, J. N. D. (1988), Two-stage, hybrid flowshop scheduling problem, *Journal of the Operational Research Society*, **39**, 359-364.

Gupta, J. N. D., Hairi, A. M. A., and Potts, C. N. (1997), Scheduling a two-stage hybrid flow shop with parallel machines at the first stage, *Annals of Operations Research*, **69**, 171-191.

Gupta, J. N. D. and Tunc, E. A. (1991), Schedules for a two-stage hybrid flowshop with parallel machines at the second stage, *International Journal of Production Research*, **29**, 1489-1502.

Hoogeveen, J. A., Lenstra, J. K., and Veltman, B. (1996), Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard, *European Journal of Operational Research*, **89**, 172-175.

Hunsucker, J. L. and Shah, J. R. (1992), Performance of priority rules in a due date flowshop, *OMEGA*, **20**, 73-89.

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

Lee, C.-Y. and Variaktarakis, G. L. (1994), Minimizing makespan in hybrid flowshops, *Operation Research Letters*, **16**, 149-159.

Lin, H.-T. and Liao, C.-J. (2003), A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics*, **86**, 133-143.

Rajendran, C. and Chaudhari, D. (1992), Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan, *International Journal of Production Economics*, **27**, 137-143.

Riane, F., Artiba, A., and Elmaghraby, S. E. (2002), Sequencing a hybrid two-stage flowshop with dedicated machines, *International Journal of Production Research*, **40**, 4353-4380.

Ribas, I., Leisten, R., and Framiñan, J. M. (2010), Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective, *Computers and Operations Research*, **37**, 1439-1454.

Ruiz, R. and Vazguez-Rodriguez., J. A. (2010), The hybrid flow shop scheduling problem, *European Journal of Operational Research*, **205**, 1-18.

Wang, S. and Liu, M. (2013), A heuristic method for two-stage hybrid flow shop with dedicated machines, *Computers and Operations Research*, **40**, 438-450.

Yang, J. (2010), A new complexity for the two-stage hybrid flow shop scheduling problem with dedicated machines, *International Journal of Production Research*, **48**, 1531-1538.

Yang, J. (2013), A two-stage hybrid flow shop with dedicated machines at the first stage, *Computers and Operations Research*, **40**, 2836-2843.

Zijm, W. H. M. and Nelissen, E. H. L. B. (1990), Scheduling a flexible machining centre, *Engineering Costs and Production Economics*, **19**, 249-258.