Industrial Engineering
& Management Systems
Vol 15, No 1, March 2016, pp.32-40
ISSN 1598-7248 │ EISSN 2234-6473 │

http://dx.doi.org/10.7232/iems.2016.15.1.032
© 2016 KIIE

A Special Case of Three Machine Flow Shop Scheduling

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

(Received: January 15, 2016 / Revised: March 4, 2016 / Accepted: March 7, 2016)

ABSTRACT

This paper considers a special case of a three machine flow shop scheduling problem in which operation processing time of each job is ordered such that machine 1 has the longest processing time, whereas machine 3, the shortest processing time. The objective of the problem is the minimization of the total completion time. Although the problem is simple, its complexity is yet to be established to our best knowledge. This paper first introduces the problem and presents some optimal properties of the problem. Then, it establishes several special cases in which a polynomial-time optimal solution procedure can be found. In addition, the paper proves that the recognition version of the problem is at least binary NP-complete. The complexity of the problem has been open despite its simple structure and this paper finally establishes its complexity. Finally, a simple and intuitive heuristic is developed and the tight worst case bound on relative error of 6/5 is established.

Keywords: Three Machine Flow Shop, Computational Complexity, Total Completion Time

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

1. INTRODUCTION

This paper considers a special case of a three machine flow shop scheduling problem in which operation processing time of each job is ordered such that machine 1 has the longest processing time, whereas machine 3, the shortest processing time. The objective of the problem is the minimization of the total completion time.

Although the flow shop scheduling problem has received much research attention over the last six decaades since the seminal work of Johnson (1954), there remain many variations of the problem of which com-plexity has not been established. Furthermore, many of them seem simple and easily solvable including the problem considered in this paper.

The flow shops are common in a variety of industries where a sequential processing of raw materials is required, and typical instances can be found in industries with assembly process including consumer electronics, automobiles, and toys. The problem considered in this paper is a simple version of the real world problems, but findings in ths paper can be extend for a more practical situations.

For a general introduction to the flow shop scheduling problem, see Baker and Trietch (2009) and Chen *et al.* (1999). In addition, for discussions about the complexity of the flow shop scheduling problem, see Garey *et al.* (1979), Lenstra *et al.* (1977), and Chen *et al.* (1999).

It is well known that the recognition version of two machine flow shop scheduling with the objective of minimizing the total completion time is unary NP-complete (Garey *et al.*, 1979). However, the two machine flow shop version of the problem in this paper, in which operation processing time of each job is ordered such that the processing time on machine 1 is no shorter than that on machine 2, is solvable in $O(n\log n)$ simply by ordering jobs in a nondecreasing order of processing times on machine 1 (Hoogeveen and Kawaguchi, 1999). The complexity of the two machine case implies that the three machine version of flow shop scheduling with the objective of minimizing the total completion time is also unary NP-complete. However, to the authors' knowledge, the complexity of the three machine version in which operation processing time of each job is ordered such that machine 1 has the longest processing time, whereas machine 3, the shortest processing time has been open. This paper establishes that the recognition version of this special case is at least binary NP-complete.

Sections 2 and 3 introduce the notations and present some optimal properties of the problem, respectively. Section 4 establishes several special cases in which a polynomial-time optimal solution procedure can be found. Section 5 establishes the complexity of the problem. In Section 6, a simple and intuitive heuristic is introduced and it is shown that the heuristic's worst case bound on relative error is 6/5 and the bound is tight. Finally, Section 7 concludes the paper by summarizing the results and discussing some avenues for future research.

2. NOTATION

The decision variables in our model are
σ = schedule of all jobs
σ_i = schedule of jobs on machine *i* for $i \in \{1, 2, 3\}$

Other notation includes
n = number of jobs
N = set of jobs $= \{1, 2, \cdots, n\}$
M = set of machines at stage 1 $= \{1, 2, \cdots, m\}$
M_i = machine *i* for $i \in M$
p_{ij} = processing time of job *j* on M_i for $i \in M$ and $j \in N$
$C_j(\sigma_i)$ = completion time of job *j* on M_i in σ for $i \in M$ and $j \in N$
$C_j(\sigma)$ = completion time of job *j* in σ for $j \in N$
$z(\sigma)$ = value of schedule σ.

If there is no confusion, then we replace $C_j(\sigma)$, $C_j(\sigma_i)$, and $z(\sigma)$ with C_j, C_{ij}, and z, respectively. We classify the problem according to the standard classification scheme for scheduling problems (Graham, Lawler, Lenstra, and Rinnooy Kan, 1979). In the three field notation of $\alpha_1 | \alpha_2 | \alpha_3$, α_1 describes the machine structure, α_2 gives job characteristics and restrictions, and α_3 defines the objective. Following the standard scheduling classification scheme in Graham, Lawler, Lenstra, and Rinnooy Kan (1979), we refer to the problem of minimizing the total completion time in a three machine flow shop with ordered operation processing time such that machine 1 has the longest processing time, where as machine 3, the shortest processing time as $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_j$.

A *schedule* defines a job order for each machine, and in this paper, a *permutation schedule* is a schedule in which sequences of jobs for all $i = 1, 2, 3$ are identical. For $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_j$, jobs are available at the start of the planning process, and no preemptions are allowed. Lastly, the general analysis of the problem is similar to that in Yang (2013) and Yang (2015).

3. BASIC RESULTS

This section establishes some properties of an optimal schedule. The *inserted idle time* occurs if a machine is intentionally kept idle even if there is a job waiting. Since there are no restrictions that delay jobs, the following result is obtained.

Lemma 1. *For problem $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_j$, there exists an optimal schedule without inserted idle time on M_i for $i \in M$.*

Because of the following remark, we only consider a schedule with the same job sequence on the first two machines.

Remark 1. *For problem $F3 || \sum C_j$, there exists an optimal schedule in which the same job sequence occurs on the first two machines.*
Proof. The result can be found in Baker and Trietsch (2009). □

The following remark establishes that the optimal schedule for problem $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_i$ may not be a permutation schedule.

Remark 2. *For problem $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_j$, there exist an instance in which there is no optimal permutation schedule.*
Proof. Consider the four job problems in which $p_{11} = p_{21} = p_{31} = 3$, $p_{12} = p_{13} = p_{14} = 4$, and $p_{22} = p_{32} = p_{23} = p_{33} = p_{24} = p_{34} = 0$. Here the non-permutation optimal solution is $\sigma_1 = (1, 2, 3, 4)$, $\sigma_2 = (1, 2, 3, 4)$, and $\sigma_3 = (2, 1, 3, 4)$. The value of the optimal solution is 43. The best permutation schedule is $\sigma = (1, 2, 3, 4)$, and its solution value is 44. □

As a result of Remark 2, there is a need to consider a non-permutation schedule as well as a permutation schedule as a candidate for an optimal schedule. The next lemma establishes a useful property of an optimal schedule for the analysis of the problem.

Lemma 2. *For problem $F3 | p_{1j} \ge p_{2j} \ge p_{3j} | \sum C_j$, if there exists a permutation schedule in which (1) jobs are sequenced in nondecreasing p_{1j} and (2) for each job in N, there exists no delay between M_i and M_{i+1} for $i = 1, 2$, then this schedule is optimal.*

Proof. Note that a non-decreasing value of p_{1j} generates a schedule with the minimum $\sum_{j=1}^{n} C_j(\sigma_1)$. Because there is no delay between M_i and M_{i+1} for $i = 1, 2$, $C_j(\sigma) = C_j(\sigma_1) + p_{1j} + p_{2j}$ for all $j \in N$. Therefore, the result holds.□

4. SPECIAL CASES

This section introduces some special cases, and for each case, a polynomial-time optimal solution procedure is developed. The following result shows that if processing time on M_3 is shorter than the minimum processing time on M_2 for all jobs, then an optimal schedule can be easily obtained.

Theorem 1. *For problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$, *if* $p_{3j} \leq \min_{j \in N}\{p_{2j}\}$ *for all* $j \in N$, *then nondecreasing* p_{1j} *is optimal.*
Proof. Hoogeveen and Kawaguchi (1999) prove that a sequence with nondecreasing p_{1j} is an optimal solution procedure for problem $F2 | p_{1j} \geq p_{2j} | \sum C_j$. Because $p_{3j} \leq \min_{j \in N}\{p_{2j}\}$, the solution value of any schedule σ, $z(\sigma) = \sum_{j=1}^{n} C_j(\sigma_3) = \sum_{j=1}^{n} C_j(\sigma_2) + \sum_{j=1}^{n} p_{3j}$. In addition, a sequence with nondecreasing p_{1j} generates the minimum $\sum_{j=1}^{n} C_j(\sigma_2)$ for the problem. Therefore, the result holds. □

Corollary 1. *For problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$, *if* $p_{3j} = p_3$ *for all* $j \in N$, *then nondecreasing* p_{1j} *is optimal.*
Proof. Because $p_{3j} = p_3$, $p_3 \leq \min_{j \in N}\{p_{2j}\}$. From Theorem 1, the result holds. □

The next result establishes that if processing time on M_2 is less than half of the minimum processing time on M_1 for all jobs, then an optimal schedule can be easily found.

Theorem 2. *For problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$, *if* $2p_{2j} \leq \min_{j \in N}\{p_{1j}\}$ *for all* $j \in N$, *then nondecreasing* p_{1j} *is optimal.*
Proof. Because $2p_{2j} \leq \min_{j \in N}\{p_{1j}\}$, $p_{2j} + p_{3j} \leq \min_{j \in N}\{p_{1j}\}$ for all $j \in N$. This implies that a job completes at M_3 no later than the completion time of the following job on M_1. Hence, the solution value of any schedule σ, $z(\sigma) = \sum_{j=1}^{n} C_j(\sigma_3) = C_j(\sigma_1) + \sum_{j=1}^{n} (p_{2j} + p_{3j})$. Observe that a sequence with nondecreasing p_{1j} generates the minimum $\sum_{j=1}^{n} C_j(\sigma_1)$. Therefore, the result holds. □

The following theorem establishes that if processing time on M_1 is the same for all jobs, then an optimal schedule can be easily obtained.

Theorem 3. *For problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$, *if* $p_{1j} = p_1$ *for all* $j \in N$, *then nondecreasing* p_{2j} *is optimal.*
Proof. Because $p_{1j} = p_1$ for all $j \in N$, a sequence with nondecreasing p_{2j} generates a schedule in which there is no delay between M_i and M_{i+1} for $i = 1, 2$ for all jobs. From Lemma 2, the result holds. □

A similar result can be established for the case in which processing time on M_2 is the same as follows.

Theorem 4. *For problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$, *if* $p_{2j} = p_2$ *for all* $j \in N$, *then nondecreasing* p_{1j} *is optimal.*
Proof. Since $p_{2j} = p_2$ for all $j \in N$, a sequence with nondecreasing p_{1j} generates a schedule in which there is no delay between M_i and M_{i+1} for $i = 1, 2$ for all jobs. From Lemma 2, the result holds. □

5. COMPLEXITY

This section establishes that the recognition version of problem $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$ is at least binary NP-complete. Specifically, we show that problem is binary NP-complete by using the reduction from the following binary NP-complete problem.

Even-Odd Partition (Garey and Johnson, 1979): Given a set of 2ℓ positive integers $A = \{a_1, a_2, \cdots, a_{2\ell}\}$ such that $a_1 < a_2 < \cdots < a_{2\ell}$, does there exist a partition of A into two subsets A_1 and A_2 such that $\sum_{a_j \in A_1} a_i = \sum_{a_j \in A_2} a_i$, and such that for each $1 \leq j \leq \ell$, A_1 contains exactly one of $\{a_{2j-1}, a_{2j}\}$?

Theorem 5. *The recognition version of problem* $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$ *is at least binary NP-complete.*

Proof. For notational convenience, let $b = \left(\sum_{j=1}^{2\ell} a_j\right)/2$, $r_j = 2^{(j-1)/2}b + a_j$ for $j = 1, 3, \cdots, 2\ell - 1$, $r_j = 2^{j/2-1}b + a_j$ for $j = 2, 4, \cdots, 2\ell$, $s_j = 2^{2\ell-(j+1)/2}b$ for $j = 2\ell+1, 2\ell+3, \cdots, 4\ell-1$, $s_j = 2^{2\ell-j/2}b$ for $j = 2\ell+2, 2\ell+4, \cdots, 4\ell$, $L = 2^{\ell} \times$

$100b$, and $W = 2\ell L - \sum_{j=1}^{2\ell} s_{2\ell+j}$. Given an instance of Even-Odd Partition, we construct the following instance of problem $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$:

$$
\begin{aligned}
& n = 4\ell + 2, \\
& m = 3, \\
& p_{1j} = L, \quad j = 1, 2, \cdots, 2\ell, \\
& p_{2j} = L - b, \ j = 1, 2, \\
& p_{2j} = L - r_j, \ j = 3, 4, \cdots, 2\ell, \\
& p_{3j} = L - s_j, \ i = 1, 2, 3, \ j = 2\ell+1, 2\ell+2, \cdots, 4\ell, \\
& p_{i,2\ell+j} = L - s_j, \ i = 1, 2, 3, \ j = 2\ell+1, 2\ell+2, \cdots, 4\ell, \\
& p_{1,4\ell+1} = L - \varepsilon, \\
& p_{2,4\ell+1} = L - b\sum_{j=1}^{\ell} 2^{j-1} - b + \varepsilon, \\
& p_{3,4\ell+1} = 0, \\
& p_{1,4\ell+2} = 2L - b\sum_{j=1}^{\ell} 2^{j-1} - b, \\
& p_{i,4\ell+2} = 0, \ i = 2, 3, \\
& z = 2\ell W + (2\ell+3)\ell L - 2(\ell+3)b - \sum_{j=1}^{\ell} (\ell + j + 3.5)2^j b \\
& \quad - \sum_{j=1}^{\ell} (\ell - j)(a_{2j-1} + a_{2j}) - (\ell+1)\varepsilon,
\end{aligned}
$$

where ε is a small positive number.

The recognition version of problem $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$ is in NP because $\sum C_j$ can be calculated in polynomial time.

Now it is proved that there exists a solution to Even-Odd Partition if and only if there exists a solution to problem $F3 | p_{1j} \geq p_{2j} \geq p_{3j} | \sum C_j$ with $\sum C_j \leq z$.

($\Rightarrow$) We assume without loss of generality that if there exists a solution to Even-Odd Partition, then elements are indexed such that $\sum_{j=1}^{\ell} a_{2j-1} = \sum_{j=1}^{\ell} a_{2j} = b$ for $j = 1, 2, \cdots, \ell$. Consider permutation schedule σ shown in Figure 1, where

$$
\begin{aligned}
\sigma = (& 2\ell+1, 2\ell+2, \cdots, 4\ell, 1, 3, \cdots, 2\ell-1, \\
& 4\ell+1, 2, 4, \cdots, 2\ell, 4\ell+2)
\end{aligned} \tag{1}
$$

and there is no inserted idle time. Note that in σ, there is no idle time on M_3 between jobs $2j-1$ and $2j+1$ and between jobs $2j$ and $2j+2$ for $j = 1, 2, \cdots, \ell-1$ because of the structure of processing times of the jobs. In addition, because jobs $2\ell + j$ and $2\ell + j + 1$ are identical for $j = 1, 2, \cdots, \ell-1$, there exists no idle time between these two jobs. Finally, since there exists a solution to Even-Odd Partition, there exists no idle time on M_3 between jobs $2\ell-1$ and $4\ell+1$ and between jobs 2ℓ and $4\ell+2$. Observe that the completion time of job 4ℓ on M_1, $C_{4\ell}(\sigma_1) = \sum_{j=2\ell+1}^{4\ell} p_{1j} = 2\ell L - \sum_{j=1}^{2\ell} s_{2\ell+j} = W$. Then, we calculate the total completion time as follows.

$$
\begin{aligned}
\sum_{j=1}^{4\ell+2} C_j &= \sum_{j=1}^{\ell} C_{2j-1} + \sum_{j=1}^{\ell} C_{2j} + \sum_{j=2\ell+1}^{4\ell} C_j + C_{4\ell+1} + C_{4\ell+2} \\
&= \sum_{j=1}^{\ell} \left\{ W + (j+2)L - b - \sum_{k=1}^{j} r_{2k-1} \right\} \\
&+ \sum_{j=1}^{\ell} \left\{ W + (\ell + j + 3)L - b - \sum_{k=1}^{j} r_{2k} - \varepsilon \right\} \\
&+ \sum_{j=1}^{\ell} \{ (4j+3)L - 2\sum_{k=1}^{j} s_{2\ell+2k} - 3s_{2\ell+2j} \} \\
&+ \left\{ W + (\ell+2)L - b - \sum_{j=1}^{\ell} r_{2j-1} \right\} \\
&+ \left\{ W + (2\ell+3)L - b - \sum_{j=1}^{\ell} r_{2j} - \varepsilon \right\} \\
&= 2(\ell+1)W + \ell(4\ell+11)L - 2(\ell+1)b \\
&- \sum_{j=1}^{\ell} (\ell - j + 1)(r_{2j-1} + r_{2j}) \\
&- \sum_{j=1}^{\ell} (r_{2j-1} + r_{2j}) \\
&- \sum_{j=1}^{\ell} \{ 4(\ell - j + 1)s_{2\ell+2j} + 3s_{2\ell+2j} \} - (\ell+1)\varepsilon
\end{aligned}
$$

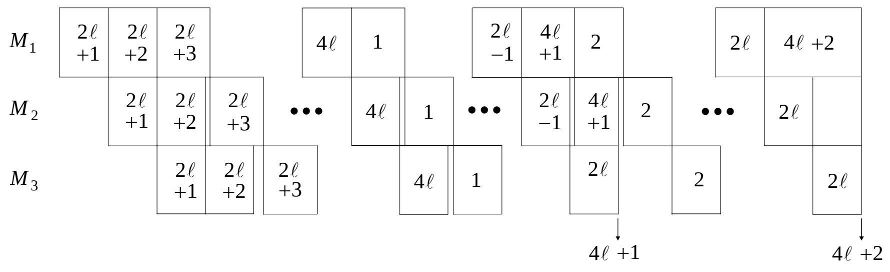

Figure 1. An example of an optimal schedule.

$$= 2\ell W + \ell(4\ell+11)L - 2(\ell+1)b$$
$$-\sum_{j=1}^{\ell}(\ell-j+2)(2^j b + a_{2j-1} + a_{2j})$$
$$-\sum_{j=1}^{\ell}\{4(\ell-j+1)2^{\ell-j} + 2^{\ell-j}3\}b - (\ell+1)\varepsilon$$
$$= 2\ell W + \ell(4\ell+11)L - 2(\ell+3)b - \sum_{j=1}^{\ell}(\ell+j+3.5)2^j b$$
$$-\sum_{j=1}^{\ell}(\ell-j)(a_{2j-1}+a_{2j}) - (\ell+1)\varepsilon$$
$$= z.$$

($\Leftarrow$) Let $I_1 = \{2\ell+1, 2\ell+2, \cdots, 4\ell\}$. In addition, if there exists a solution to Even-Odd Partition, then we assume that elements in set A_1 and A_2 are partitioned into sets I_2 and I_3, respectively and that they are reindexed such that $I_2 = \{1, 3, \cdots, 2\ell-1\}$ and $I_3 = \{2, 4, \cdots, 2\ell\}$. Because each job in I_1 has an identical processing time on each of the three machines and its processing time increases with an increase in the index, it can be seen that in an optimal schedule, jobs in I_1 are processed in their index order. Here an optimal schedule is described first. Then, we show that a schedule does not have a solution value $\le z$ unless it is optimal.

From Lemma 2, we try to find a permutation schedule satisfying the two conditions in the lemma. If a schedule that meets these conditions cannot be found, then it is the best to find a schedule with the least violation of the two conditions in terms of the solution value.

First, note that processing time of job $2\ell+2$ on M_1 is $2L - b\sum_{j=1}^{\ell}2^{j-1} - b$, which is substantially longer than that of all other jobs. Hence, it can be seen that it is optimal to schedule job $4\ell+2$ last.

Next, observe that processing times of jobs in I_1 on M_1 are shorter than other jobs by at least b. Hence, we start constructing an optimal schedule by scheduling all jobs in I_1 first in their index order. Note that because each job in I_1 has the identical processing time on all three machines and the processing time increases with an increase in the index, each job in I_1 can be processed without any delay between M_i and M_{i+1} for $i = 1, 2$.

If some other jobs are scheduled before or between jobs in I_1, then the solution value increases in comparison to a schedule in which all jobs in I_1 are scheduled first. This is because jobs in I_1 have shorter processing times on M_1 than other jobs. Recall that except for jobs $4\ell+1$ and $4\ell+2$, jobs in I_2 and I_3 have the same processing time on M_1, that is, L. Also, recall that $p_{1,4\ell+1} = L - \varepsilon \approx L$. First, consider job $4\ell+1$ which has the shortest total processing time as job $4\ell+2$ does. Note that only way to schedule job $4\ell+1$ without any delay between M_i and M_{i+1} for $i = 1, 2$ is to process job $4\ell+1$ first. However, in comparison to a schedule that schedules job $4\ell+1$ after jobs in I_1, as in schedule (1), this increases the solution value by about $\sum_{j=1}^{2\ell}(p_{1,4\ell+1} - p_{1,2\ell+j}) = 2\sum_{j=1}^{\ell}2^{\ell-j}b$, which is the sum of processing time gaps on M_1 between job $4\ell+1$ and each job in I_1. Now, if job $4\ell+1$ is scheduled after job $2\ell+2$, then the delay between M_2 and M_3 for job $4\ell+1$ is $p_{2,2\ell+2} + p_{3,2\ell+2} - (p_{1,4\ell+1} + p_{3,4\ell+1}) = 2L - 2^{\ell-1}b - (2L - \sum_{j=1}^{\ell}2^{j-1}b) = \sum_{j=1}^{\ell}2^{j-2}b$. In addition, there is an increase in the solution value from the early processing of the job with a longer processing time on M_1 than jobs in I_1. Hence, it is optimal to schedule job $4\ell+1$ after jobs in I_1 if a schedule with no delay or a shorter delay between M_2 and M_3 cannot be found.

Next, consider job $2\ell-1$ or 2ℓ with the shortest total processing time among remaining jobs except for jobs $4\ell+1$ and $4\ell+2$. Note that the best way to schedule, for example, job $2\ell-1$ without any delay between M_i and M_{i+1} for $i = 1, 2$ is to process job $2\ell-1$ after job $2\ell+2$. However, in comparison to a schedule that schedules job $2\ell-1$ after jobs in I_1, as in schedule (1), this increases the solution value by about $2\sum_{j=2}^{\ell}2^{\ell-j}b = \sum_{j=1}^{\ell-1}2^{\ell-j}b$, as in the case of job $4\ell+1$. There is another way to schedule $2\ell-1$ with a small delay between M_2 and M_3, that is, $a_{2\ell-1}$. In other words, job $2\ell-1$ can be scheduled after job $2\ell+4$ with a delay of $a_{2\ell-1}$ between M_2 and M_3, but then this increases the solution value by about $2\sum_{j=3}^{\ell}2^{\ell-j}b = \sum_{j=2}^{\ell-1}2^{\ell-j}b$ in comparison to a schedule, in which job $2\ell-1$ is processed after jobs in I_1, as in schedule (1). Note that this increase is smaller than that for scheduling job $2\ell-1$ after job $2\ell+2$. In addition, if job $2\ell-1$ is scheduled after job $2\ell+6$, then the delay between M_2 and M_3 is $(2^{\ell-1} + \varepsilon - 2^{\ell-2})b$. Hence, it is optimal to schedule job $2\ell-1$ after jobs in I_1 if a schedule with no delay or a shorter delay between M_2 and M_3 cannot be found.

This procedure can be repeated to calculate the increase in the solution value or the delay between M_2

and M_3 for all remaining jobs in I_2 and I_3.

Now consider schedule (1). Note that the completion time of job $2\ell-1$ in (1) is $C_{2\ell-1} = W + (\ell+2)L - \sum_{j=1}^{\ell} r_{2j-1} = W + (\ell+2)L - \sum_{j=1}^{\ell} 2^{j-1}b - \sum_{j=1}^{\ell} a_{2j-1}$. This implies that job $2\ell-1$ is delayed after it is scheduled on M_2 by $r_{2\ell-1} - \sum_{j=1}^{\ell-1} r_{2j-1} = 2^{\ell-1}b + a_{2\ell-1} - \sum_{j=1}^{\ell-1}(2^{j-1}b + a_{2j-1})$ before it is processed on M_3. Note that this delay is smaller than $\sum_{j=2}^{\ell-1} 2^{\ell-j}b$, which is the increase in the solution value for scheduling job $2\ell-1$ after job $2\ell+4$. In addition, this delay is shorter than $(2^{\ell-1}+\varepsilon-2^{\ell-2})b$, which is the delay for scheduling job $2\ell-1$ after job $2\ell+6$. Similarly, job 2ℓ in schedule (1) has almost the same delay as job $2\ell-1$.

Observe that if jobs $2\ell-1$ and $4\ell+1$ are scheduled as in schedule (1), then it is possible that job 2ℓ is scheduled right after job $4\ell+1$ without any delay between M_2 and M_3. However, this increases the delay of job $4\ell+2$ between M_2 and M_3 by about $r_{2\ell} = 2^{\ell-1}b + a_{2\ell}$ because $p_{3,2\ell} = L - 2^{\ell-1}b - a_{2\ell}$. Note that the delay between M_2 and M_3 created by scheduling job 2ℓ right before job $4\ell+2$, as in schedule (1), is only $r_{2\ell} - \sum_{j=1}^{\ell-1} r_{2j} = 2^{\ell-1} + a_{2\ell} - \sum_{j=1}^{\ell-1}(2^{j-1} + a_{2j})$. Hence, in an optimal schedule, job 2ℓ is not scheduled after job $4\ell+1$.

This argument can be repeated for all remaining jobs in I_2 and I_3. In addition, the same argument can be used to show that jobs in I_2 and jobs in I_3 are separated by job $4\ell+1$ in the job sequence.

As for a sequence among jobs in I_2, scheduling job 1 right after job 4ℓ does not create any delay between M_2 and M_3. If job 3 is scheduled right after job 4ℓ instead of job 1, then it generates a delay of a_1 for job 3 between M_2 and M_3. In addition, it can be seen that it creates a delay of $r_1 = b + a_1$ for jobs $5, 7, \cdots, 2\ell-1, 4\ell+1$ in comparison to a schedule in which job 3 is scheduled right after job 1, as in schedule (1). Alternatively, if job 3 is scheduled right after job 1, which is scheduled right after job 4ℓ, then a delay of $r_3 - r_1 = 2b + a_3 - (b + a_1) = b + a_3 - a_1$ is created for job 3, which is smaller than the increase in the solution value for the other case. Hence, it is better to schedule job 3 after job 1. This argument can be repeated for all remaining jobs in I_2 to determine an optimal order for jobs in I_2. Similarly, the same argument can be used to sequence jobs in I_3. Therefore, it is optimal to schedule jobs in I_2 and I_3 as in schedule (1).

Job $4\ell+1$ in an optimal schedule should be scheduled as early as possible because it has a slightly shorter processing time on M_1 than jobs in I_2 and I_3 as long as there is no delay between M_i and M_{i+1} for $i = 1, 2$. Hence, it is better to schedule job $4\ell+1$ after job $2\ell-1$ than to schedule it after job 2ℓ. Observe that the processing time of job $4\ell+1$ on M_2 is $L - \sum_{j=1}^{\ell} 2^{j-1}b - b + \varepsilon$. Hence, job $4\ell+1$ can be processed without any delay between M_2 and M_3 only if job $4\ell+1$ is scheduled after job $2\ell-1$ and $\sum_{j=1}^{\ell} a_{2j-1} \ge b$. Similarly, the processing times of job $4\ell+2$ on M_1 and M_2 are $2L - \sum_{j=1}^{\ell} 2^{j-1}b - b$ and 0, respectively. Hence, job $4\ell+2$ can be processed without any delay between M_2 and M_3 only if job $4\ell+2$ is scheduled after job 2ℓ and $\sum_{j=1}^{\ell} a_{2j} \ge b$. These two conditions can be satisfied simultaneously only if $\sum_{j=1}^{\ell} a_{2j} = \sum_{j=1}^{\ell} a_{2j-1} = b$.

Observe that the optimal schedule is the same as schedule (1). Hence, the solution value is the same as that for schedule (1). This is possible only if $\sum_{j=1}^{\ell} a_{2j} = \sum_{j=1}^{\ell} a_{2j-1} = b$. That is, there exists a solution to Even-Odd Partition. □

6. HEURISTICS

Since problem $F3 \mid p_{1j} \ge p_{2j} \ge p_{3j} \mid \sum C_j$ is at least binary NP-complete, it is impossible to develop a polynomial time optimal solution procedure. Hence, it would be valuable to develop heuristics which can generate reasonably good solution values. In this section, we introduce a simple and intuitive heuristic, H and analyze the performance of the heuristic analytically. Specifically, we show that the worst case bound on relative error of the H is 6/5 and the bound is tight.

6.1 Description of the Heuristic

Heuristic H is based on a simple rule. It processes jobs in the SPT (Shortest Processing Time first) order of p_{1j} for $j = 1, 2, \cdots, n$, and as a result it ensures that each job can be processed as early as possible. We briefly describe the H1 as follows.

Heuristic H
0. Reindex jobs so that $p_{1j} \le p_{1,j+1}$ for $j = 1, 2, \cdots, n-1$.
1. Schedule jobs $1, 2, \cdots, n$ as early as possible without

any unnecessary idle time in their index order.

2. Calculate C_j for $j = 1, 2, \cdots, n$.

   Output $\sum_{j=1}^{n} C_j$ and stop.

Heuristic H runs in $O(n \log n)$ time. The resulting solution is a permutation schedule where there is no inserted idle time on M_i for $i = 1, 2, 3$. Let σ^H be the schedule generated by Heuristic H and z^H be the cost of schedule σ^H.

6.2 A Lower Bound

We establish a lower bound on the value of a schedule and the lower bounds is used in the analysis of the heuristic. The following bound assumes that each job is processed as quickly as possible on M_i for $i = 1, 2, 3$. We let $[j]$ indicate the job in the jth position in schedule σ and z^L be the cost of the lower bound.

Lemma 3.

$$\sum_{j=1}^{n} C_{[j]} \geq z^L = \sum_{j=1}^{n} (n-j) p_{1j} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j}).$$

Proof. If job $j = 1$, then $C_{[1]} = p_{1[1]} + p_{2[1]} + p_{3[1]}$. Since each job is processed on each machine as quickly as possible, for each job $j \in \{2, 3, \cdots, n\}$, $C_{[j]} \geq \sum_{k=1}^{j} p_{1[k]} + p_{2[j]} + p_{3[j]}$. Hence, if we add up $C_{[j]}$ for jobs $1, 2, \cdots, n$, then

$$\sum_{j=1}^{n} C_{[j]} \geq \sum_{j=1}^{n} (n-j+1) p_{1[j]} + \sum_{j=1}^{n} (p_{2[j]} + p_{3[j]})$$
$$= \sum_{j=1}^{n} (n-j) p_{1[j]} + \sum_{j=1}^{n} (p_{1[j]} + p_{2[j]} + p_{3[j]}).$$

Therefore, $z^L = \sum_{j=1}^{n} (n-j) p_{1[j]} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j})$. □

6.3 Analysis of the Heuristic

This section analyzes the H. Specifically, we establish the worst case bound on the relative error for the H and prove that the bound is tight. First, we assume $p_{1j} \leq p_{1,j+1}$ for $j = 1, 2, \cdots, n-1$. The following thoerem establishes that for the H, there exists a tight worst case bound on relative error.

Theorem 6. $z^H / z^* \leq 6/5$ *and this bound is tight.*

Proof. Since job 1 is the first job to be processed, $C_1(\sigma^H) = p_{11} + p_{21} + p_{31}$. Also, since there is no waiting time before M_i for $i = 1, 2, 3$ in σ^H and $p_{1j} \geq p_{2j} \geq p_{3j}$ for $j = 2, 3, \cdots, n$,

$$C_j(\sigma^H) \leq \sum_{k=1}^{j-1} p_{1k} + \max\{p_{2j}, p_{3,j-1}\} + p_{3j}. \tag{2}$$

If we add up $C_j(\sigma^H)$ in (2) for jobs $1, 2, \cdots, n$, then

$$\sum_{j=1}^{n} C_j(\sigma^H) \leq \sum_{j=1}^{n} (n-j+1) p_{1j} + \sum_{j=1}^{n} \max\{p_{2j}, p_{3,j-1}\}$$
$$+ \sum_{j=1}^{n} p_{3j} \leq \sum_{j=1}^{n} (n-j+1) p_{1j} + \sum_{j=1}^{n} p_{2j} + \sum_{j=1}^{n-1} p_{3j} + \sum_{j=1}^{n} p_{3j}. \tag{3}$$

From Lemma 3 and since jobs are processed in their index order,

$$z^L = \sum_{j=1}^{n} (n-j) p_{1j} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j})$$
$$= \sum_{j=1}^{n} (n-j+1) p_{1j} + \sum_{j=1}^{n} (p_{2j} + p_{3j}). \tag{4}$$

Then, from (3) and (4)

$$z^H / z^L = \frac{\sum_{j=1}^{n} C_j(\sigma^H)}{z^L}$$
$$\leq \frac{\sum_{j=1}^{n} (n-j+1) p_{1j} + \sum_{j=1}^{n} p_{2j} + \sum_{j=1}^{n-1} p_{3j} + \sum_{j=1}^{n} p_{3j}}{\sum_{j=1}^{n} (n-j+1) p_{1j} + \sum_{j=1}^{n} (p_{2j} + p_{3j})}$$
$$= 1 + \frac{\sum_{j=1}^{n-1} p_{3j}}{\sum_{j=1}^{n} (n-j) p_{1j} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j})}$$
$$= 1 + \frac{\sum_{j=1}^{n-1} p_{3j}}{\sum_{j=1}^{n-1} (n-j-1) p_{1j} + \sum_{j=1}^{n-1} p_{1j} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j})}$$
$$= 1 + \frac{\sum_{j=1}^{n-1} p_{3j}}{\sum_{j=1}^{n-2} (n-j-1) p_{1j} + \sum_{j=1}^{n-2} p_{1j} + \sum_{j=1}^{n-1} p_{1j} + \sum_{j=1}^{n} (p_{1j} + p_{2j} + p_{3j})}$$

$$\leq 1 + \frac{\sum_{j=1}^{n-1} p_{3j}}{\sum_{j=1}^{n-2}(n-j-2)p_{1j} + 3\sum_{j=1}^{n-1} p_{1j} + \sum_{j=1}^{n}(p_{2j}+p_{3j})}$$
$$\leq 6/5. \tag{5}$$

The inequality (5) holds since $\sum_{j=1}^{n-1} p_{1j} \geq \sum_{j=1}^{n-1} p_{2j} \geq \sum_{j=1}^{n-1} p_{3j}$ and $p_{1n} \geq p_{1,n-1}$.

We now prove that the bound is tight. Consider the instance where there are $n = 2$ jobs with processing times $p_{11} = 1$, $p_{21} = p_{31} = 0$, and $p_{12} = p_{22} = p_{32} = 1$. Since $p_{11} = p_{12}$, any job sequence can be generated by the H. Suppose that $\sigma^H = (2, 1)$. Then, the solution value is $z^H = 3 + 3 = 6$. An optimal schedule $\sigma^* = (1, 2)$ has solution value $z^* = 1 + 4 = 5$. Therefore, the relative error is 6/5 and this shows that the bound is tight. □

7. A SUMMARY AND FUTURE RESEARCH

This paper explores problem $F3 \mid p_{1j} \geq p_{2j} \geq p_{3j} \mid \sum C_j$. Specifically, the paper establishes the complexity of $F3 \mid p_{1j} \geq p_{2j} \geq p_{3j} \mid \sum C_j$, which has been open, and presents a simple polynomial-time solution procedure for several of its special cases. The complexity of the problem has been open despite its simple structure and this paper finally establishes its complexity. With this result, future research can be focused more on developing heuristics.

Also, a simple and intuitive heuristic, Heuristic H is introduced and it is shown that the H's worst case bound on relative error is 6/5 and the bound is tight.

Table 1 summarizes the work. The blank space in Additional Job Characteristics column implies no job restrictions. All cases assume problem $F3 \mid p_{1j} \geq p_{2j} \geq p_{3j} \mid \sum C_j$. The table lists the theorem or corollary showing each complexity.

Future research may consider some other special cases of problem $F3 \| \sum C_j$. For instance, the complexity of problem $F3 \mid p_{1j} \leq p_{2j} \leq p_{3j} \mid \sum C_j$ is open to our best knowledge even though structure of the problem is as simple as the problem considered in this paper. A good starting point of the analysis can be problem $F2 \mid p_{1j} \leq p_{2j} \mid \sum C_j$, which can be solved in $O(n^2 \log n)$ (Hoogeveen and Kawaguchi, 1999). In addition, additional heuristics should be developed for our problem because its complexity is established as binary NP-complete.

The findings in ths paper can be used in a variety of industries where a sequential processing of raw materials is required, and typical instances can be found in industries with assembly process including consumer electronics, automobiles, and toys. Even thoug the problem considered in this paper is simple, the results can be extend for a more practical situations

ACKNOWLEDGMENTS

This work was supported by the 2015 Research Fund of the University of Seoul.

REFERENCES

Baker, K. R. and Trietsch, D. (2009), *Flow shop scheduling*, Principles of sequencing and scheduling, John Wiley and Sons, 225-249.

Chen, B., Potts, C. N., and Woeginger, G. J. (1993), *A review of machine scheduling: complexity*, algorithms and approximability, Handbook of Combinatorial Optimization, Springer US, 1493-1641.

Garey, M. R. and Johnson, D. S. (1979), *Computers and intractibility: a guide to the theory of NP-completeness*, New York: W. H. Freeman and Co.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976), The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, **1**, 117-129.

Table 1. Complexity of problem $F3 \mid p_{1j} \geq p_{2j} \geq p_{3j} \mid \sum C_j$

Additional Job Characteristics	Complexity Results	
	Binary NP-Comp.	Theorem 5
$p_{1j} = p_1$ for all $j \in N$	$O(n \log n)$	Theorem 3
$p_{2j} = p_2$ for all $j \in N$	$O(n \log n)$	Theorem 4
$p_{3j} = p_3$ for all $j \in N$	$O(n \log n)$	Corollary 1
$p_{3j} \leq \min_{j \in N}\{p_{2j}\}$ for all $j \in N$	$O(n \log n)$	Theorem 1
$2p_{2j} \leq \min_{j \in N}\{p_{1j}\}$ for all $j \in N$	$O(n \log n)$	Theorem 2

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.

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

Johnson, S. M. (1954), Optimal two- and three-stage production with setup times included, *Naval Research Quarterly*, 61-68.

Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. (1977), Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, **1**, 343-362.

Yang, J. (2013), No Tardiness Rescheduling with Order Disruptions, *Industrial Engineering and Management Systems*, **12**, 51-62.

Yang, J. (2015), Hybrid Flow Shop with Parallel Machines at the First Stage and Dedicated Machines at the Second Stage, *Industrial Engineering and Management Systems*, **14**, 22-31.