7.Flow-shop Scheduling Problem with Weighted Work-In-Process

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

Abstract

This paper considers a new flow-shop scheduling problem where a different WIP (work-in-process) state has different weight on the duration time.

For the two machine case, the recognition version is NP-Complete in the strong sense. Several special cases are solved by different polynomial time algorithms. Finally, we develop a heuristic and provide an upper-bound on relative error which is tight in limit.

1. Introduction.

The scheduling literature considers a wide variety of problems and objectives. One important objective that has not received much attention is the work-in-process cost (WIP) associated with value that is added during the production process. In the production process, as labor and material are added to the product, the value of the product and hence the WIP costs increase. Hence, a factory may be able to reduce the total WIP costs by keeping WIP inventory at the earlier stage of manufacturing process instead of keeping inventory at the later stage.

Minimizing WIP costs is an important criterion for many manufacturing facilities. Conway et al. (1988) describes the role of WIP in serial production lines and investigates the behavior of lines buffered with storage and explores the distribution and quantity of WIP inventory that accumulates.

Any scheduling problem with the objective of minimizing total completion time minimizes the average WIP inventory during the entire manufacturing process of jobs. In this case, the WIP cost remains the same throughout the manufacturing process.

With the minimum-wait objectives, the cyclic sequencing problem minimizes the average WIP inventory of partially finished jobs subject to the constraint that the jobs have to be produced at the maximum throughput rate. A difference from the regular scheduling problem with the objective of minimizing total completion time is that the problem recognizes the WIP cost only when jobs are not processed by a machine. However, the WIP cost remains the same for the entire manufacturing process. For surveys on the problem, see Kamoun and Srikandarajah (1993) and Matsuo (1990).

In this paper, we consider a new flow-shop scheduling problem where a different WIP state has different weight on the duration time. The value is added while a raw material is processed through the flow-shop. We consider the two machine flow-shop case for this paper.

The two machine flow-shop problem with the objective of minimizing total completion time is a well known problem. The recognition version of this problem is known to be NP-Complete in the strong sense (Garey et al. ,1979). Several studies are done and most of them are focused on developing efficient algorithms (Wang et al., 1996;Hoogeveen and Kawaguchi, 1999;Croce et al., 1996, 2002).

In the next section, we describe the problem

and introduce some notation. Then, we establish complexity of the problem. Also, some preliminary results and solutions for special cases are presented. Finally, we develop a heuristic and provide an upper-bound on relative error which is tight in limit.

2. Description of Problem

We assume that all jobs are available at time zero. Let N be a set of jobs $N = \{1,2,\mathrm{K}\ ,n\}$ where n is the number of jobs. Following the standard scheduling classification schedule of Graham et al. (1979), we refer to the problem of minimizing the WIP cost in a flow shop as $F2 \| \sum WIP_j$ where WIP_j is *work in process cost* for job j. In a two machine system, there are four different types of WIP costs:

- Type 1: before a job is put into the first machine (value of raw material)
- Type 2: a job is being processed by machine 1 (value of raw material + added components at machine 1)
- Type 3: after a job is processed by machine 1 but before the job is put into machine 2 (value of raw material + added components at machine 1 + labor and depreciation of machine 1)
- Type 4: a job is being processed by machine 2 (value of raw material + added components at machine 1 + labor and depreciation of machine 1 + added components at machine 2)

Now, we assign different weight (value) to each WIP inventory. Let w_i be weight for Type i inventory for $i = 1,2,3,4$. Then, we have Remark 1.

Remark 1 *For problem* $F2 \| \sum WIP_j$,

$$w_1 \le w_2 \le w_3 \le w_4 .$$

A *schedule* defines a job order for each machine and a *permutation schedule* is a schedule in which every machine has the same job order and no preemption is allowed. Let T_{1j} be waiting time of job j before it starts processing on machine 1 and T_{2j} be waiting time before job j starts processing on machine 2 for $j = 1,2,\mathrm{K}\ ,n$, respectively. The completion time of job j is $C_j = T_{1j} + p_{1j} + T_{2j} + p_{2j}$ where p_{ij} is processing time of job j on machine i for $i = 1,2$. While the actual weight might vary based on the job, most of the jobs on a flow line are similar. Consequently, one reasonable model is that the *value added* from a given operation is proportional to the time spent on the machine. Thus, the WIP cost for job j is

$$WIP_j = w_1 T_{1j} + w_2 p_{1j} + w_3 T_{2j} + w_4 p_{2j}. \quad (1)$$

Note that $w_2 p_{1j}$ and $w_4 p_{2j}$ are fixed regardless of job sequence.

3. Complexity of Problem $F2 \| \sum WIP_j$

The following remark establishes that problem $F2 \| \sum C_j$ is a special case of problem $F2 \| \sum WIP_j$.

Remark 2 *If* $w_1 = w_3$, *then problem* $F2 \| \sum WIP_j$ *is identical to problem* $F2 \| \sum C_j$.

Proof. Note that $C_j = T_{1j} + p_{1j} + T_{2j} + p_{2j}$ where p_{1j} and p_{2j} are fixed for any job sequence. Let $\overline{w} = w_1 = w_3$. If we divide WIP_j by $\overline{w}$, then

$$\frac{WIP_j}{\overline{w}} = T_{1j} + \frac{w_2 p_{1j}}{\overline{w}} + T_{2j} + \frac{w_4 p_{2j}}{\overline{w}}.$$

Since $w_2 p_{1j} / \overline{w}$ and $w_4 p_{1j} / \overline{w}$ remain the same for any schedule, the objective of problem $F2 \| \sum WIP_j$ is now to minimize $\sum_{j=1}^{n} (T_{1j} + T_{2j})$. This is the same as minimizing $\sum C_j$. Hence, we have the result. □

The following theorem establishes the complexity of problem $F2 \| \sum WIP_j$.

Theorem 1 *The recognition version of problem* $F2 \| \sum WIP_j$ *is NP-Complete in the strong sense.*

Proof. Note that the recognition version of problem $F2\|\sum C_j$ is NP-Complete in the strong sense (Garey et al., 1979). From Remark 2, $F2\|\sum C_j$ is a special case of $F2\|\sum WIP_j$. □

The following theorem establishes the complexity of problem $F2\|\sum WIP_j$ when $p_{11} = p_{12} = \Lambda = p_{1n}$.

Theorem 2 *The recognition version of problem* $F2\|\sum WIP_j$ *with* $p_{11} = p_{12} = \Lambda = p_{1n}$ *is NP-Complete in the strong sense.*

Proof. Note that the recognition version of problem $F2\|\sum C_j$ with $p_{11} = p_{12} = \Lambda = p_{1n}$ is NP-Complete in the strong sense (Hoogeveen and Kawaguchi, 1999). From Remark 2, $F2\|\sum C_j$ is a special case of $F2\|\sum WIP_j$. □

The following lemma establishes the relationship between problem $F2\|\sum WIP_j$ and problem $F2\,|\,no-wait\,|\sum C_j$.

Lemma 1 *If* $(n-1)w_1 \le w_3$ *, then problem* $F2\|\sum WIP_j$ *is identical to problem* $F2\,|\,no-wait\,|\sum C_j$.

Proof. Let σ^* be an optimal schedule for problem $F2\|\sum WIP_j$. Note that in σ^* , the first job requires no inserted idle time on machine 1 since no job is scheduled ahead on machine 2.

We assume that in σ^* , jobs are scheduled in their index order. Suppose that job 2 has wait time before machine 2, say $\Delta t > 0$. Inserting idle time $\Delta t > 0$ on machine 1 for job 2 leads to at most Δt additional wait time for job 2 and each subsequent job before machine 1 compared to a schedule without an inserted idle time. Without inserted idle time, the schedule must have at least Δt idle time on machine 2.

Hence, delaying job 2 by Δt on machine 1 can create additional wait time of at most $(n-1)\Delta t$ before machine 1 in total instead of Δt wait time before machine 2 for job 2. Similarly, delaying job 3 by Δt on machine 1 can create additional wait time of at most $(n-2)\Delta t$ before machine 1 instead of Δt wait time on machine 2 and so on.

Therefore, it is sufficient to have $(n-1)w_1\Delta t \le w_3\Delta t$ for an optimal schedule to have no idle time on machine 2. □

The following corollary establishes complexity of problem $F2\|\sum WIP_j$ with $(n-1)w_1 \le w_3$.

Corollary 1 *The recognition version of problem* $F2\|\sum WIP_j$ *with* $(n-1)w_1 \le w_3$ *is NP-Complete in the strong sense.*

Proof. Note that the recognition version of problem $F2\,|\,no-wait\,|\sum C_j$ is NP-Complete in the strong sense (Röck, 1984). From Lemma 2, problem $F2\,|\,no-wait\,|\sum C_j$ is identical to problem $F2\|\sum WIP_j$ with $(n-1)w_1 \le w_3$. □

4. Preliminary Results

In this section, we establish that two important characteristics of the problem. We begin with the following lemma.

Lemma 2 *For problem* $F2\|\sum WIP_j$ *, there exists an optimal permutation schedule.*

Proof. We use pair wise interchange argument. Suppose that in a unique optimal schedule, jobs are processed in their index order. For $j,k \in N$, suppose that jobs j and k for $j < k$ are processed consecutively on machine 1 and job k precedes job j on machine 2.

If $p_{2k} > p_{2j}$, then switching processing order on machine 2 decreases solution value, a contradiction. If $p_{2k} \le p_{2j}$, then we need to consider two cases. Suppose $p_{1k} > p_{1j}$. Then, switching processing order on machine 1 does not increase solution value, a contradiction. Alternatively, suppose that $p_{1k} \le p_{1j}$. By switching processing order on machine 1, C_{1j} is decreased by p_{1k} and C_{1k} is increased by p_{1j} , a

contradiction. □

As a result of Lemma 2, we only consider a permutation schedule.

Lemma 3 *For problem $F2\|\sum WIP_j$, some optimal schedule requires inserted idle time on machine 1.*

Proof. Consider the instance where $n=2$, $p_{11}=p_{12}=1$, and $p_{21}=p_{22}=2$. Note that any job sequence is optimal because the two jobs are identical. Without loss of generality, we assume that (1,2) is an optimal schedule. Note that job 1 can start as early as at time 0 and completes at time 3. Hence, $C_1=1+2=3$. Job 2 can start at time 1 on machine 1 but then, job 2 must wait until job 1 completes on machine 2 at time 3. However, since $w_3 \geq w_1$, it is optimal to start job 2 on machine 2 at time 2. Hence, for job 2, one unit of inserted idle time is required on machine 1. □

As a result of Lemma 3, when we describe an optimal schedule, we may need to specify a job order and start time (or completion time) of each job. Having inserted idle time is a crucial difference between problems $F2\|\sum C_j$ and $F2\|\sum WIP_j$. As a remark, for problem $F2\|\sum C_j$, there exists at least one optimal permutation schedule without any idle time on machine 1 (Conway et al., 1967).

5. Solutions for Special Cases

In this section, we establish several special cases for problem $F2\|\sum WIP_j$. The following remark establishes a special case for the problem.

Remark 3 *For problem $F2\|\sum WIP_j$, suppose that we schedule jobs in SPT order of processing time on machine 1 without inserted idle time on machine 1. If $T_{2j}=0$ for all $j \in N$, then this schedule is optimal.*

Proof. Since we schedule jobs in SPT order of processing time on machine 1, $\sum_{j=1}^{n} T_{1j}$ is minimized. Since $T_{2j}=0$ for all $j \in N$, from (1), we have the result. □

Corollary 2 *The SPT rule without inserted idle time on machine 1 is optimal for $F2\,|\,p_{ij}=p_j\,|\sum WIP_j$.*

Proof. Since $p_{ij}=p_j$, SPT rule without inserted idle time on machine 1 guarantees that there exists no idle time before a job is put into machine 2. From Remark 3, SPT rule is optimal for $F2\,|\,p_{ij}=p_j\,|\sum WIP_j$. □

The following remark establishes the third special case for the problem.

Remark 4 *For problem $F2\|\sum WIP_j$, suppose that $p_{1j} \geq p_{2j}$ for all $j \in N$. Then, SPT order of processing time on machine 1 without inserted idle time on machine 1 is optimal.*

Proof. Since we schedule jobs in SPT order of processing time on machine 1 without inserted idle time on machine 1, $\sum_{j=1}^{n} T_{1j}$ is minimized. Also, $T_{2j}=0$ for all $j \in N$ because $p_{1j} \geq p_{2j}$ for all $j \in N$. □

Note that the condition in Remark 4 is usually true in real world since final assembly generally takes shorter time than fabrication process.

Remark 5 *If $w_1=0$, then any job sequence is optimal for problem $F2\|\sum WIP_j$.*

Proof. Since $w_1=0$, for any $j \in N$, job j can be delayed on machine 1 so that $T_{2j}=0$. Hence, any job sequence is optimal for problem $F2\|\sum WIP_j$. □

The next theorem establishes that if processing times on machine 2 are the same for all jobs, then SPT order of processing time on machine 1 without any wait time before machine 2 is optimal.

Theorem 3 *For problem $F2\|\sum WIP_j$, suppose that $p_{21}=p_{22}=\Lambda=p_{2n}$. Then, SPT order of processing time on machine 1 without any wait time before*

machine 2 is optimal.

Proof. Let $\bar{p}_2 = p_{2j}$ for $j = 1,2,\mathrm{K}\,,n$. We need to consider the following three cases. We assume that jobs are indexed in nondecreasing order of p_{1j} for $j = 1,2,\mathrm{K}\,,n$.

Case 1. $\bar{p}_2 \le p_{12}$.

Note that $WIP_1 = w_2 p_{11} + w_4 p_{21}$. Since $\bar{p}_2 \le p_{12}$, SPT order of processing time on machine 1 does not generate any wait time before machine 2. From Remark3, the result holds.

Case 2. $\bar{p}_2 > p_{1n}$.

Note that $WIP_1 = w_2 p_{11} + w_4 p_{21}$. Since $w_3 \ge w_1$, it is optimal that we schedule jobs $2,3,\mathrm{K}\,,n$ such that the jobs do not wait before machine 2. So, we delay each job by $\bar{p}_2 - p_{1j}$ for $j = 2,3,\mathrm{K}\,,n$ on machine 1. Note that this does not create any additional wait time compared to a schedule with no inserted idle time on machine 1. Then, for $j = 2,3,\mathrm{K}\,,n$,

$$WIP_j = w_1\{p_{11} + (j-2)\bar{p}_2 + \bar{p}_2 - p_{1j}\} + w_2 p_{1j} + w_4 p_{2j}. \quad (2)$$

If we add up for $j = 1,2,\mathrm{K}\,,n$, then

$$\begin{aligned}&\sum_{j=1}^{n} WIP_j \\ &= w_1(n-1)p_{11} + w_1\sum_{j=2}^{n}(j-1)\bar{p}_2 - w_1\sum_{j=2}^{n} p_{1j} + \sum_{j=1}^{n}(w_2 p_{1j} + w_4 p_{2j}) \\ &= w_1\left\{np_{11} + \frac{n(n-1)\bar{p}_2}{2} - \sum_{j=1}^{n} p_{1j}\right\} + \sum_{j=1}^{n}(w_2 p_{1j} + w_4 p_{2j}).\end{aligned}$$

Notice that $\sum_{j=1}^{n} WIP_j$ is minimized as long as job 1 is scheduled first where $\arg\min_{1\le j\le n} p_{1j} = 1$. Hence, we have the result.

Case 3. $p_{12} < \bar{p}_2 \le p_{1n}$.

Let σ be a schedule generated by SPT order of processing time on machine 1 without any wait time before machine 2. Since $p_{12} < \bar{p}_2 \le p_{1n}$, there exists $k \in N$ such that $p_{1k} \le \bar{p}_2 \le p_{1,k+1}$ for $k \in \{3,4,\mathrm{K}\,,n-1\}$. Note that $WIP_1(\sigma) = w_2 p_{11} + w_4 p_{21}$. For $2 \le j \le k$, we use the result from Case 2. From (2),

$$WIP_j(\sigma) = w_1\{p_{11} + (j-2)\bar{p}_2 + \bar{p}_2 - p_{1j}\} + w_2 p_{1j} + w_4 p_{2j} \quad (3)$$

for $j = 2,3,\mathrm{K}\,,k$. For $j = k+1$, we have

$$WIP_j(\sigma) = w_1\{p_{11} + (k-1)\bar{p}_2\} + w_2 p_{1j} + w_4 p_{2j}. \quad (4)$$

Similarly, for $j \ge k+2$, we have

$$WIP_j(\sigma) = w_1\{p_{11} + (k-1)\bar{p}_2\} + w_1\sum_{u=k+1}^{j-1} p_{1u} w_2 p_{1j} + w_4 p_{2j}.$$

We use job pair wise interchange argument. Suppose there exists a unique optimal schedule where for some $j,k \in N$, job j precedes job k and $p_{1j} > p_{1k}$. Let jobs q and r be the first pair of such jobs where q and r are processed consecutively and $p_{1q} > p_{1r}$.

If $q \le k$ and $r \ge k+1$, then the proofs are similar to those in Cases 1 and 2, respectively. Suppose that $r = k$ and $q = k+1$. Let σ' be the new schedule. Then,

$$WIP_q(\sigma') = w_1\{p_{11} + (k-2)\bar{p}_2\} + w_2 p_{1q} + w_4 p_{2q}$$ and

$$\begin{aligned}&WIP_r(\sigma') \\ &= w_1\{p_{11} + (k-1)\bar{p}_2 + (p_{1q} - \bar{p}_2) + (\bar{p}_2 - p_{1r})\} + w_2 p_{1r} + w_4 p_{2r} \\ &= w_1\{p_{11} + (k-1)\bar{p}_2 + p_{1q} - p_{1r}\} + w_2 p_{1r} + w_4 p_{2r}\end{aligned}$$

Note that

$$w_1(\bar{p}_2 - p_{1k}) + w_2(p_{1k} + p_{1,k+1}) \le w_1(p_{1q} - p_{1r}) + w_2(p_{1q} + p_{1r})$$

since $p_{1q} \ge \bar{p}_2$. Hence, from (3) and (4),

$$\begin{aligned}&WIP_q(\sigma') + WIP_r(\sigma') \\ &= w_1\{2p_{11} + (2k-3)\bar{p}_2 + p_{1q} - p_{1r}\} + w_2(p_{1q} + p_{1r}) + w_4(p_{2q} + p_{2r}) \\ &\ge w_1\{2p_{11} + (2k-3)\bar{p}_2 + \bar{p}_2 - p_{1k}\} + w_2(p_{1k} + p_{1,k+1}) + w_4(p_{2k} + p_{2,k+1}) \\ &= WIP_q(\sigma') + WIP_r(\sigma').\end{aligned}$$

For completion time, $C_r(\sigma'_1) = p_{11} + (k-1)\bar{p}_2 + p_{1q}$ and $C_r(\sigma'_2) = p_{11} + (k-1)\bar{p}_2 + p_{1,k+1}$. Also,

$C_{k+1}(\sigma_1) = p_{11} + (k-1)\bar{p}_2 + p_{1,k+1}$ and
$C_{k+1}(\sigma_2) = p_{11} + (k-1)\bar{p}_2 + p_{1,k+1} + p_{2,k+1}$. Hence,
$C_r(\sigma'_1) = C_{k+1}(\sigma_1)$ and $C_r(\sigma'_2) = C_{k+1}(\sigma_2)$.
Therefore, we have the result. □

6. A Heuristic

In this section, we introduce a heuristic and analyze the worst case behavior of the heuristic. The heuristic is based on the approximation algorithm for problem $F2\|\sum C_j$ presented by Gonzalez and Sahni (1978).

The approximation algorithm by Gonzalez and Sahni (1978) proceeds by reindexing the jobs in order of nondecreasing $p_{1j} + p_{2j}$ for $j = 1,2,\mathrm{K}\,,n$, settling ties arbitrarily, and by subsequently scheduling jobs in that order in M_1 and M_2 such that unnecessary idle time is avoided. This leads to the set of completion times:

$$C_{11} = p_{11}, \qquad C_{21} = p_{11} + p_{21}, \ and$$
$$C_{1j} = C_{1,j-2} + p_{1j}, \ C_{2j} = \max\{C_{2,j-1}, C_{1j}\} + p_{2j},$$
$$for \ \ j = 2,3,\mathrm{K}\,,n.$$

Note that this approximation algorithm runs in $O(n\log n)$ time and the resulting schedule is a permutation schedule with no idle time on M_1 between the execution of the jobs.

6.1 Description

In order to apply the heuristic to problem $F2\|\sum WIP_j$, we modify the algorithm as follows. Since $w_1 \le w_3$, for each job, we eliminate wait time before machine 2 as much as possible. A new heuristic proceeds by reindexing the jobs in order of nondecreasing $p_{1j} + p_{2j}$ for $j = 1,2,\mathrm{K}\,,n$, settling ties arbitrarily, and sequentially schedules jobs in that order in M_1 and M_2 as the heuristic by Gonzalez and Sahni (1978). However, while minimizing completion time, jobs are delayed on machine 1 so that it does not create any wait time before machine 2. This leads to the set of completion times:

$$C_{11} = p_{11}, \qquad C_{21} = p_{11} + p_{21}, \ and$$
$$C_{1j} = \max\{C_{1,j-1} + p_{1j}, C_{2,j-1}\}, \ C_{2j} = C_{1j} + p_{2j},$$
$$for \ \ j = 2,3,\mathrm{K}\,,n.$$

The new heuristic is different from the heuristic by Gonzalez and Sahni (1978) because it inserts idle time on machine 1 to eliminate wait time before machine 2. With this assumption, our problem becomes similar to flow-shop problem with no-wait time before machine 2. As a remark, the recognition version of problem $F2\,|\,no-wait\,|\sum C_j$ is NP-Complete in the strong sense (Röck, 1984). For a survey on this problem, see Hall and Sriskandarahjah (1996) and Kanet and Sridahran (2000). We now formally describe the heuristic.

Heuristic H1

0. Reindex jobs so that $p_{1j} + p_{2j} \le p_{1,j+1} + p_{2,j+1}$ for $j = 1,2,\mathrm{K}\,,n$.
1. Schedule job 1 first so that $C_{11} = p_{11}$ and $C_{21} = p_{11} + p_{21}$.
   Schedule jobs $2,3,\mathrm{K}\,,n$ in their index order on M_1 and M_2 such that $\max\{C_{1,j-1} + p_{1j}, C_{2,j-1}\}$ and $C_{2j} = C_{1j} + p_{2j}$ for $j = 2,3,\mathrm{K}\,,n$.
   When there exist ties, break them arbitrarily.
2. From a completed schedule, calculate WIP_j for $j = 1,2,\mathrm{K}\,,n$.

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

In Step 0, reindexing the jobs requires $O(n\log n)$ time. Since all the other operations require $O(n)$ time, the time requirement of H1 is $O(n\log n)$ time.

6.2 An Upper Bound on the Relative Error

In this section, we analyze heuristic H1 and show that the worst case bound on relative error is $2\beta/(\alpha+\beta)$ where α and β denote the minimum and maximum processing time of all operations, and the bound is tight in limit.

In order to analyze the worst case behavior of the heuristic, we follow the approach by Hoogeveen and Kawaguch (1999). They analyze the heuristic by Gonzalez and Sahni (1978) and establish the same bound for problem $F2\|\sum C_j$. Our analysis is more complicated than theirs due to a different weight on each type of WIP inventory.

We begin by establishing a lower bound for an optimal schedule. Throughout this section, let σ^* be an optimal schedule. Also, we reindex jobs so that $p_{1j}+p_{2j} \le p_{1,j+1}+p_{2,j+1}$ for $j=1,2,\mathrm{K},n$.

Lemma 4 *For problem* $F2\|\sum WIP_j$,

$$\sum_{j=1}^{n} C_j(\sigma^*) \ge \frac{1}{2}\Bigg[\sum_{j=1}^{n}\{(n-j)(p_{1j}+p_{2j})\} + nw_1 \min\{p_{1j}\} + 2w_4\sum_{j=1}^{n} p_{2j} + (2w_2-w_1)\sum_{j=1}^{n} p_{1j}\Bigg].$$

Proof. Consider any schedule σ. Let $J_{[j]}$ denote the job that occupies the j th position in σ; $C_{[j]}$, $p_{1[j]}$, and $p_{2[j]}$ are defined accordingly. Note that $WIP_{[j]} = w_2 p_{1[j]} + w_4 p_{2[j]}$.

We derive the lower bound stated in the lemma by combing two lower bounds. The first of these comes from

$$WIP_{[j]} \ge w_1 p_{1[j]} + w_2\sum_{k=1}^{j-1} p_{2[k]} + (w_2 p_{1[j]} - w_1 p_{1[j]} + w_4 p_{1[j]}) + w_4 p_{2[j]}$$

for $j=2,3,\mathrm{K},n$. Note that since $w_1 \le w_2$, $w_1 p_{1[j]}$ is deducted and $w_2 p_{1[j]}$ is added to tighten the bound. From (5) and (6),

$$\begin{aligned}&\sum_{j=1}^{n} WIP_{[j]}\\ &\ge w_2 p_{1[j]} + w_4 p_{2[j]} + (n-1)w_1 p_{1[j]} + w_1\sum_{j=1}^{n-1}(n-j)p_{2[j]} + (w_2-w_1)\sum_{j=2}^{n} p_{1[j]} + w_4\sum_{j=2}^{n} p_{2[j]} \quad \text{The}\\ &\ge nw_1 p_{1[j]} + w_1\sum_{j=1}^{n-1}(n-j)p_{2[j]} + (w_2-w_1)\sum_{j=1}^{n} p_{1[j]} + w_4\sum_{j=1}^{n} p_{2[j]}. \qquad (7)\end{aligned}$$

second lower bound comes from

$$WIP_{[j]} \ge w_1\sum_{k=1}^{j-1} p_{1[k]} + w_2 p_{1[j]} + w_4 p_{2[j]} \qquad (8)$$

for $j=2,3,\mathrm{K},n$. From (5) and (8),

$$\sum_{j=1}^{n} WIP_{[j]} \ge w_2 p_{1[j]} + w_4 p_{2[j]} + w_1\sum_{j=1}^{n-1}(n-j)p_{2[j]} + w_2\sum_{j=1}^{n} p_{1[j]} + w_4\sum_{j=1}^{n} p_{2[j]}. \qquad (9)$$

Recall that jobs are indexed in their nondecreasing order of $p_{1j}+p_{2j}$ for $j=1,2,\mathrm{K},n$. Hence, $\sum_{j=1}^{n-1}(n-j)(p_{1j}+p_{2j}) \ge \sum_{j=1}^{n-1}(n-j)(p_{1[j]}+p_{2[j]})$.

By adding up (7) and (9), we have

$$\begin{aligned}&2\sum_{j=1}^{n} WIP_{[j]}\\ &\ge nw_1 p_{1[1]} + w_1\sum_{j=1}^{n-1}(n-j)(p_{1[j]}+p_{2[j]}) + (2w_2-w_1)\sum_{j=1}^{n} p_{1[j]} + 2w_4\sum_{j=1}^{n} p_{2[j]}\\ &\ge nw_1 \min_{1\le j\le n}\{p_{1j}\} + w_1\sum_{j=1}^{n-1}(n-j)(p_{1j}+p_{2j}) + (2w_2-w_1)\sum_{j=1}^{n} p_{1j} + 2w_4\sum_{j=1}^{n} p_{2j}. \qquad (10)\end{aligned}$$

□

The following lemma restates a simple rule on the minimum and maximum numbers for the sake of completeness of the proof.

Lemma 5 *We have that*

$$|p_{2k} - p_{1,k+1}| \le (\beta-\alpha)(p_{1,k+1}+p_{2k})/(\beta+\alpha)$$

for $k=1,2,\mathrm{K},n-1$.

Proof. First, consider the case where $|p_{2k}-p_{1,k+1}| = p_{2k}-p_{1,k+1}$. Suppose that $p_{2k}-p_{1,k+1} > (\beta-\alpha)(p_{1,k+1}+p_{2k})/(\beta+\alpha)$. The inequality can be rewritten as $2\alpha p_{2k} > 2\beta p_{1,k+1}$, and the right side of the inequality is $2\beta p_{1,k+1} > 2\beta\alpha$. Since $p_{2k} \le \beta$, we have a contradiction. We can apply the similar argument for the other case. □

Theorem 4 *For problem* $F2\|\sum WIP_j$, $z^{H1}/z^* \le 2\beta/(\alpha+\beta)$, *and this bound is tight in limit where* z^{H1} *is a solution value of H1.*

Proof. Note that a schedule by H1 does not

contain any wait time before machine 2. First, we have

$$WIP_1 = w_2 p_{11} + w_4 p_{21}.$$

For $j = 2,3,\mathrm{K}\ ,n$,

$$WIP_j \le w_1\left[p_{11} + \sum_{k=1}^{j-1} \max\{p_{2k}, p_{1,k+1}\} \right] - w_1 p_{1j} + w_2 p_{1j} + w_4 p_{2j}.$$

Using the equality $2\max\{p_{2k}, p_{1,k+1}\} = p_{2k} + p_{1,k+1} + | p_{2k} - p_{1,k+1} |$, we have

$$\begin{aligned} & 2WIP_j \\ &\le 2w_1 p_{11} + w_1 \sum_{k=1}^{j-1} p_{2k} + w_1 \sum_{k=1}^{j-1} p_{1,k+1} + \\ & \quad w_1 \sum_{k=1}^{j-1} | p_{2k} - p_{1,k+1} | -2w_1 p_{1j} + 2w_2 p_{1j} + 2w_4 p_{2j} \\ &= w_1 p_{11} + w_1 \sum_{k=1}^{j-1} p_{2k} + w_1 \sum_{k=1}^{j-1} p_{1k} + \\ & \quad w_1 \sum_{k=1}^{j-1} | p_{2k} - p_{1,k+1} | -w_1 p_{1j} + 2w_2 p_{1j} + 2w_4 p_{2j}. \end{aligned} \quad (12)$$

From (11) and (12), it follows that

$$\begin{aligned} & 2\sum_{j=1}^{n} WIP_j \\ &\le (n-1) w_1 p_{11} + w_1 \sum_{k=1}^{j-1} (n-j)(p_{1j} + p_{2j}) + \\ & \quad w_1 \sum_{j=1}^{n-1} \sum_{k=1}^{j} | p_{2k} - p_{1,k+1} | -w_1 \sum_{j=2}^{n} p_{1j} + 2w_2 \sum_{j=1}^{n} p_{1j} + 2w_4 \sum_{j=1}^{n} p_{2j} \\ &\le n w_1 p_{11} + w_1 Y + w_1 \sum_{j=1}^{n-1} \sum_{k=1}^{j} | p_{2k} - p_{1,k+1} | - \\ & \quad w_1 \sum_{j=2}^{n} p_{1j} + 2w_2 \sum_{j=1}^{n} p_{1j} + 2w_4 \sum_{j=1}^{n} p_{2j} \end{aligned}$$

where $Y = w_1 \sum_{k=1}^{j-1} (n-j)(p_{1j} + p_{2j})$ for notational convenience. Using Lemma 5, we have

$$\begin{aligned} & 2\sum_{j=1}^{n} WIP_j \\ &\le n w_1 p_{11} + w_1 Y + \\ & \quad \left\{ w_1 (Y + \sum_{j=2}^{n} p_{1j}) - n w_1 (p_{11} + p_{21}) \right\} (\beta - \alpha)/(\beta + a) \\ & \quad + (2w_2 - w_1) \sum_{j=1}^{n} p_{1j} + 2w_4 \sum_{j=1}^{n} p_{2j} \qquad \text{Note} \\ &\le \frac{2w_1 \beta Y}{\alpha + \beta} + 2w_4 \sum_{j=1}^{n} p_{2j} + \\ & \quad \frac{\{2w_2(\alpha+\beta) - 2w_1\alpha\} \sum_{j=1}^{n} p_{1j}}{\alpha + \beta} + \frac{2n w_1 \alpha p_{11}}{\alpha + \beta}. \end{aligned} \quad (13)$$

that

$$\begin{aligned} & 2w_2(\alpha+\beta) - 2w_1\alpha \\ &= 4w_2\beta - 2w_2\beta + 2w_2\alpha - 2w_1\alpha \\ &= 4w_2\beta - 2w_2\beta + 2w_1\beta - 2w_1\beta + 2(w_2 - w_1)\alpha \\ &= 4w_2\beta + 2(w_2 - w_1)\alpha - 2(w_2 - w_1)\beta - 2w_1\beta \\ &\le 4w_2\beta - 2w_1\beta. \end{aligned}$$

From (13) and (14),

$$\begin{aligned} & 2\sum_{j=1}^{n} WIP_j \\ &\le \frac{2w_1\beta Y}{\alpha+\beta} + 2w_4 \sum_{j=1}^{n} p_{2j} + \frac{\{2(2w_2 - w_1\}\beta \sum_{j=1}^{n} p_{1j}}{\alpha+\beta} \\ & \quad + \frac{2n w_1 \alpha p_{11}}{\alpha+\beta} \\ &\le \frac{2\beta}{\alpha+\beta} \left\{ w_1 Y + 2w_4 \sum_{j=1}^{n} p_{2j} + 2(2w_2 - w_1\} \sum_{j=1}^{n} p_{1j} \right. \\ & \quad \left. + n w_1 \min_{1 \le j \le n} \{p_{1j}\} \right\}. \end{aligned}$$

Then, the worst case bound follows from Lemma 4.

Now, we show that the bound is tight in limit. Consider the following instance. There are $2m$ jobs with processing times $p_{1j} = \beta$, $p_{2j} = \alpha$ for $j = 1,2,\mathrm{K}\ ,n$ and $p_{1j} = \alpha$ and $p_{2j} = \beta$ for $j = m+1, m+2, \mathrm{K}\ ,2m$. Since $p_{1j} + p_{2j}$ is equal for all jobs, any job sequence can be a result from the heuristic. Suppose that $\sigma^{H1} = (1,2,\mathrm{K}\ ,2m)$. Then, solution value is $z^{H1} = w_1(2m^2 - m)\beta + w_2 m(\alpha+\beta) + w_4 m(\alpha+\beta)$. An optimal schedule is $\sigma^* = (m+1,1,m+2,2,\mathrm{K}\ ,2m,m)$ and solution value is

$$z^* = w_1(m^2 - m)\beta + w_1 m^2 \alpha + w_2 m(\alpha+\beta) \quad + w_4 m(\alpha+\beta).$$

The relative error bound goes to $2\beta/(\alpha+\beta)$ for

$m \to \infty$. □

7. Summary and Discussions

We have explored a new flow-shop scheduling problem where a different WIP (work-in-process) state has different weight on the duration time. Specifically, the two machine flow-shop problem is considered. Even if the problem seems simple, it is difficult to solve because problem $F2 \| \sum WIP_j$ is a general case of $F2 \| \sum C_j$ where problem $F2 \| \sum C_j$ is already NP-Hard in the strong sense. We establish some preliminary results and several special cases. We also develop a heuristic and provide an upper-bound on relative error which is tight in limit.

For future research, we want to develop more heuristics and study their performance. We also want to explore more general cases of the problem such as different weights on WIP costs for different jobs. This makes the problem harder, but it is more realistic.

References

Conway R., W. Maxwekk, J.O. McClain, and L.J. Thomas (1988), The role of work-in-process inventory in serial production lines, Operations Research, 36, 229-241.

Croce F.D., V. Narayan, and R. Tadei (1996), The two-machine total completion time flow shop problem, European Journal of Operational Research, 90, 227-237.

Croce F.D., M. Ghirardi, and R. Tadei (2002), An improved branch-and-bound algorithm for the two machine total completion time flow shop problem, European Journal of Operational Research, 139, 293-301.

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

Gonzalez, T. and S. Sahni (1978) Flowshop and job shop schedules: Complexity and approximation, Oper. Res., 26, 36-52

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

Hall, N.G. and C. Sriskandarahjah (1996), A survey of machine scheduling problems with blocking and no-wait in process, Operations Research, 44, 510-525.

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

Kamoun, H C. and Srikandarajah (1993). The Complexity of Scheduling Jobs in Repetitive Manufacturing Systems, European Journal of Operational Research, 70, 350-364.

Kanet, J.J. and V. Sridaharan. (2000), Scheduling with inserted idle time: problem taxonomy and literature review, Operations Research, 48, 99-110.

Matsuo, H. (1990), Cyclic Sequencing Problems in the Two-Machine Permutation Flow Shop: Complexity, Worst-Case, and Average-Case Analysis, Naval Research Logistics, 37, 674-694

Röck, H. (1984) Some New Results in No-wait Flowshop Scheduling, Zeitschrift Für Opns. Res., 28, 1-16.

Wang, C, C. Chu, and J.-M. Proth (1996), Efficient heuristic and optimal approaches for $n/2/F/\sum C_i$ scheduling problems, Int. J. Production Economics, 44, 225-237.