Heuristics for Flow Shop Scheduling with Weighted WIP Costs

Jaehwan Yang* and Hyunsoo Kim**
*School of Business, University of Seoul (jyang@uos.ac.kr)
**Major of Industrial Engineering, Division of Advanced Industrial Engineering, Kyonggi University (hskim@kyonggi.ac.kr)

Abstract

This paper considers a flow shop scheduling problem where a different WIP (work-in-process) state has different weight on the duration time. The objective is to minimize the sum of the weighted WIP.

For the two machine flow shop case, the recognition version is unary NP-Complete. The three simple and intuitive heuristics H0, H1, and H2 are presented for the problem. For each heuristic, we find an upper bound on relative error which is tight in limit. For heuristic H2, we show that H2 dominates the other two heuristics.

1. Introduction.

An important objective that has not received much attention in the scheduling literature is the *Work In Process* (WIP) cost associated with value that is added during the production process. The value of the product and the WIP cost increases as labor and material are added to a product. Consequently, it may be possible to reduce the total WIP costs if a factory can move WIP inventory to earlier stages of manufacturing process.

Minimizing WIP costs is an important criterion for many manufacturing facilities. Level of WIP stocks is often considered as one of the measures for production efficiency (Sipper and Shapira, 1989). While it is almost impossible to operate production lines without any WIP stock, most companies try to minimize WIP (Sipper and Shapira, 1989). Some companies intentionally keep WIP inventory at work centers for better utilization (Vollman et al., 1997) or hedging against late delivery penalty, but for most of companies, reducing unnecessary WIP inventory is an important goal to achieve. For the role of WIP in serial production lines, see Conway et al. (1988).

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 of a job remains the same throughout the manufacturing process.

Another problem which considers WIP costs as the objective is the cyclic sequencing problem with the minimum-wait objectives. The 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. In this case, the WIP cost of a job also remains the same during 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 a different weight on the duration time. The value is added while a raw material is processed through the flow shop. A major difference from the other flow shop scheduling problems which we describe earlier is that the WIP cost changes (usually increases) as production process continues.

For this paper, we consider the two machine flow shop case.

The problem we consider has some similarity with the two machine flow shop problem with the objective of minimizing total completion time. Actually, this well known problem is a special case of the problem we consider. The recognition version of the two machine flow shop problem with the objective of minimizing total completion time is known to be unary NP-Complete (Garey et al. ,1979). Several studies are done and most of them are focused on developing efficient algorithms (Aidri and Amit, 1984; Velde, 1992; Wang et al., 1996;Hoogeveen and Kawaguchi, 1999;Croce et al., 1996, 2002; Lee and Wu, 2001).

In the next two sections, we introduce some notation and describe the problem. Then, we present some preliminary results. Then, we introduce three heuristics H0, H1, and H2 using a simple scheduling rule, and for each heuristic, we find an upper bound on relative error which is tight in limit. For heuristic H2, we show that H2 dominates the other two heuristics.

2. Notation

The decision variables in our models are

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

σ = schedule of all jobs = (σ_1, σ_2)

Other notation that is used in this work includes

n = number of jobs

N = set of jobs = $\{1,2,\mathrm{K}\ ,n\}$

M_i = machine i for $i \in \{1,2\}$

p_{ij} = processing time of job j on machine i for $j \in N$ and $i \in \{1,2\}$

$C_j(\sigma_i)$ = completion time of job j on machine i in schedule σ for $j \in N$ and $i \in \{1,2\}$

$C_j(\sigma)$ = completion time of job j in schedule σ for $j \in N$

$S_j(\sigma_i)$ = start time of job j on machine i in schedule σ for $j \in N$ and $i \in \{1,2\}$

$T_{1j}(\sigma)$ = wait time of job j before it starts processing on machine 1 in schedule j for $j \in N$

$T_{1j}(\sigma)$ = wait time of job j before it starts processing on machine 2 after job j completes on machine 1 in schedule σ for $j \in N$

$WIP_j(\sigma)$ = *work in process cost* for job j in schedule σ for $j \in N$ and

$z*$ = value of optimal schedule.

We may omit σ after T_{1j}, T_{2j}, C_j, and WIP_j when there exists no confusion. The standard classification scheme for scheduling problems (Graham et al., 1979) is $\alpha_1 \mid \alpha_2 \mid \alpha_3$, where α_1 describes the machine structure, α_2 gives the job characteristics or restrictive requirements, and α_3 defines the objective function to be minimized. We extend this scheme to provide for WIP costs by using WIP_j in the α_3 field. Following the standard scheduling classification schedule of Graham et al. (1979), we refer to the problem of minimizing the WIP cost in a two machine flow shop as $F2 \mid\mid \sum WIP_j$. All of the heuristic procedures we develop use the following well known rule to determine the order in which the jobs are processed.

SPT (Shortest Processing Time): When machine 1 becomes available, an unscheduled job with a shortest total processing time is selected first for processing.

3. Description of Problem

We assume that all jobs are available at time zero. 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)

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

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

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

4. Preliminary Results for $F2 \| \sum WIP_j$

We begin by reviewing the following results.

Theorem 1 (Yang, 2005) *The recognition version of problem* $F2 \| \sum WIP_j$ *is NP-Complete in the strong sense.*

Lemma 1 (Yang, 2005) *For problem* $F2 \| \sum WIP_j$, *there exists an optimal permutation schedule.*

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

Lemma 2 (Yang, 2005) *For problem* $F2 \| \sum WIP_j$, *some optimal schedule requires inserted idle time 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). Finally, we present the following lower bound which is established by Yang (2005).

Lemma 3 (Yang, 2005) *For problem* $F2 \| \sum WIP_j$,

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

5. Heuristic with No Wait Time

In this section, we introduce a heuristic and analyze the worst case behavior of the heuristic. The heuristic is originally based on the approximation algorithm for problem $F2 \| \sum C_j$ presented by Gonzalez and Sahni (1978), and Yang (2005) modifies for $F2 \| \sum WIP_j$. The heuristic works 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. 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:

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

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.

Heuristic H0 (Yang, 2005).

0. Reindex jobs so that $p_{1j} + p_{2j} \le p_{1,j+1} + p_{2,j+1}$ for $j = 1,2,\mathrm{K}\ ,n-1$.

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.

The following theorem by Yang (2005) provides an upper bound on the relative error which is tight in limit.

Theorem 2 (Yang, 2005) *For problem* $F2 \,||\, \sum WIP_j$, $z^{H0} / z^* \le 2\beta/(\alpha + \beta)$, *and this bound is tight in limit where* z^{H0} *is a solution value of H0 where* α *and* β *denote the minimum and maximum processing time of all operations.*

6. Heuristic with No Inserted Idle Time

In this section, we introduce a heuristic and analyze the worst case behavior of the heuristic. The heuristic is the approximation algorithm for problem $F2 \,||\, \sum C_j$ presented by Gonzalez and Sahni (1978). We apply the same heuristic to problem $F2 \,||\, \sum WIP_j$, which has a different objective function. Since the heuristic uses SPT and does not allow any inserted idle time, it is typical choice for operation managers who are interested in maximizing utilization of machines.

6.1 Description

This 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 sequentially schedules 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:

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

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. For the sake of completeness, we 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 $j = 1,2,\mathrm{K}\ ,n-1$.

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 $C_{1j} = C_{1j-1} + p_{1j}$ and $C_{2j} = \max\{C_{2,j-1}, C_{1j}\} + p_{2j}$ 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.

6.2 An Upper Bound on the Relative Error

In this section, we analyze heuristic H1 and find the worst case bound on relative error. We assume throughout this subsection that jobs are indexed so that $p_{1j} + p_{2j} \le p_{1,j+1} + p_{2,j+1}$ for $j = 1,2,\mathrm{K}\ ,n-1$. Then, we show that the bound is $[u/(u+v) + w_3 v/\{w_1(u+v)\}]/2$ where $u = \sum_{j=1}^{n-1}(n-j)p_{1j}$, $v = \sum_{j=1}^{n-1}(n-j)p_{2j}$, and the bound is tight in limit. Note that the bound has the minimum of 2 when $w_1 = w_3$.

The following theorem proves the bound. For the following theorem, let $u = \sum_{j=1}^{n-1}(n-j)p_{1j}$ and $v = \sum_{j=1}^{n-1}(n-j)p_{2j}$.

Theorem 2 *For problem* $F2\,||\sum WIP_j$, $z^{H1}/z^* \le [u/(u+v) + w_3 v/\{w_1(u+v)\}]/2$, *and this bound is tight in limit where* z^{H1} *is a solution value of H1.*

Proof. Let σ^{H1} be a schedule generated by heuristics H1. Also, we let the value of the lower bound from Lemma 3 be z^L. Then,

$$z^L \ge \frac{1}{2}w_1\{\sum_{j=1}^{n}(n-j)(p_{1j}+p_{2j}) - w_1\sum_{j=1}^{n}p_{1j}\} + w_2\sum_{j=1}^{n}p_{1j} + w_4\sum_{j=1}^{n}p_{2j}. \quad (2)$$

By construction of H1,

$$WIP_1(\sigma^{H1}) = w_2 p_{11} + w_4 p_{21} \quad (3)$$

and

$$WIP_j(\sigma^{H1}) = w_1\sum_{k=1}^{j-1}p_{1k} + w_2 p_{1k} + w_3\sum_{k=1}^{j-1}p_{2k} + w_4 p_{2j} \quad (4)$$

for $j = 2,3,\mathrm{K}\ ,n$. By combining (2) and (3), we have

$$\sum_{j=1}^{n}WIP_j(\sigma^{H1}) = w_1\sum_{j=1}^{n-1}(n-j)p_{1j} + w_2\sum_{j=1}^{n}p_{1j} + w_3\sum_{j=1}^{n-1}(n-j)p_{2j} + w_4\sum_{j=1}^{n}p_{2j}. \quad (5)$$

From (2) and (5),

$$\frac{z^{H1}}{z^*} \le \frac{z^{H1}}{z^L} \le \frac{w_1 u + w_3 v + w_2\sum_{j=1}^{n}p_{1j} + w_4\sum_{j=1}^{n}p_{2j}}{\frac{w_1(u+v) - w_1\sum_{j=1}^{n}p_{1j}}{2} + w_2\sum_{j=1}^{n}p_{1j} + w_4\sum_{j=1}^{n}p_{2j}}. \quad (6)$$

Now, we need to consider two cases. First, suppose that $u \ge v$. From (6),

$$\frac{z^{H1}}{z^L} \le \frac{w_1 u + \left(\frac{u}{u+v}\right)w_2\sum_{j=1}^{n}p_{1j} + w_4\sum_{j=1}^{n}p_{2j}}{\frac{w_1(u+v) + w_2\sum_{j=1}^{n}p_{1j}}{2} + w_4\sum_{j=1}^{n}p_{2j}} + \frac{w_3 v + \left(\frac{v}{u+v}\right)w_2\sum_{j=1}^{n}p_{1j}}{\frac{w_1(u+v) + w_2\sum_{j=1}^{n}p_{1j}}{2}} \le \frac{2u}{u+v} + \frac{2w_3 v}{w_1(u+v)}. \quad (7)$$

Similarly, for the case where $u < v$,

$$\frac{z^{H1}}{z^L} \le \frac{w_1 u + \left(\frac{u}{u+v}\right)w_2\sum_{j=1}^{n}p_{1j}}{\frac{w_1(u+v) + w_2\sum_{j=1}^{n}p_{1j}}{2}} + \frac{w_3 v + \left(\frac{v}{u+v}\right)w_2\sum_{j=1}^{n}p_{1j} + w_4\sum_{j=1}^{n}p_{2j}}{\frac{w_1(u+v) + w_2\sum_{j=1}^{n}p_{1j}}{2} + w_4\sum_{j=1}^{n}p_{2j}} \le \frac{2u}{u+v} + \frac{2w_3 v}{w_1(u+v)}. \quad (8)$$

From (7) and (8), we prove the worst case bound.

Now, we show that the bound is tight in limit. Consider the following instance. There are $2m$ jobs with processing times $p_{1j} = 1$ and $p_{2j} = 0$ for $j = 1,2,\mathrm{K}\ ,n$ and $p_{1j} = 0$ and $p_{2j} = 1$ 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, the solution value is $z^{H1} = w_1(3m^2 - m)/2 + w_3(m^2 - m)/2 + w_2 m + w_4 m$.

An optimal schedule $\sigma^* = (m+1, m+2, \text{K}, 2m, 1, 2, \text{K}, m)$ and the solution value is $z^* = w_1(m^2 - m) + w_2 m + w_4 m$.

The relative error goes to $3/2 + w_3/(2w_1)$ for $m \to \infty$.

Note that $u = \sum_{j=1}^{n-1}(n-j)p_{1j} = (3m^2 - m)/2$ and

$v = \sum_{j=1}^{n-1}(n-j)p_{2j} = (m^2 - m)/2$. As $m \to \infty$,

$2u/(u+v)$ goes to 3/2 and $2v/(u+v)$ goes to 1/2.

Hence, the relative error bound goes to $3/2 + w_3/(2w_1)$ for $m \to \infty$. □

7. Heuristic with Inserted Idle Time and Wait Time

In this section, we introduce the third heuristic which dominates heuristics H0 and H1. Throughout this section, let $\gamma = \lfloor w_3 / w_1 \rfloor$. Note that $\gamma \geq 1$.

7.1 Description

The heuristic proceeds by reindexing the jobs in order of nondecreasing $p_{1j} + p_{2j}$ for $j = 1, 2, \text{K}, n$, settling ties arbitrarily, and sequentially schedules jobs in that order in M_1 and M_2 such that unnecessary idle time is avoided. Then, for jobs in positions from $n-\gamma+1$ to n, the heuristic delays the jobs on machine 1 so that they do not create any wait time before machine 2. Throughout this section, job $[j]$ denotes the job that occupies the j th position in σ; $C_{i[j]}$ and $S_{i[j]}$ are defined accordingly. Suppose that $S_{2,[n-\gamma+k]} - C_{1,[n-\gamma+k]} = \Delta t > 0$. Then, delaying job $[n-\gamma+k]$ by Δt decreases wait time of the job before machine 2 by Δt for $k \in \{1, 2, \text{K}, \gamma\}$. But, it also increases wait time before machine 1 for each of job $[n-\gamma+k]$ and subsequent jobs by at most Δt, and the total increase of wait time is no greater than $\gamma\Delta t$. Since $w_3 \geq \gamma w_1$, delaying job $[n-\gamma+k]$ for $k \in \{1, 2, \text{K}, \gamma\}$ does not increase solution value and may improve the solution value.

Now, the heuristic checks whether job $[n-\gamma]$ can be delayed on machine 1 without delaying the following jobs. Formally, the heuristic checks whether $C_{1,[n-\gamma]} < S_{2,[n-\gamma]}$. If it does, then the heuristic also checks whether job $[n-\gamma]$ can be delayed without delaying the very next job. This is possible if $C_{1,[n-\gamma]} < S_{1,[n-\gamma+1]}$. If it does, then delay job $[n-\gamma]$ by $\min\{S_{2,[n-\gamma]} - C_{1[n-\gamma]}, S_{1,[n-\gamma+1]} - C_{1[n-\gamma]}\}$. Note that if job $[n-\gamma]$ can be delayed, then the solution value must be decreased due to $w_3 \geq w_1$.

If job $[n-\gamma]$ is delayed, then the heuristic needs to check whether job $[n-\gamma-1]$ can be also delayed without delaying the next job, which is job $[n-\gamma]$. The heuristic tries to repeat this process for jobs $[n-\gamma-1], [n-\gamma-2], \text{K}, 2$, $[n-\gamma-2]$ until there exists no such delay any more.

For convenience, we call this process *free delay*.

Performing free delay means searching for free delays and delaying associated jobs. Note that free delay is created when job $[j]$ for $j \in N$ is delayed on machine 1 and jobs before job $[j]$ can be delayed on machine 1 without delaying following jobs so that it reduces the solution value. Notice that an optimal schedule does not have any free delay.

After performing initial free delay, the heuristic considers job $[2]$. If $C_{1[2]} < S_{2[2]}$, then the heuristic considers job $[2]$ for delaying by $S_{2[2]} - C_{1[2]}$. Alternatively, if $C_{1[2]} \geq S_{2[2]}$, then the heuristic continues with the next job, job $[3]$. Since there does not exist any free delay, delaying job $[2]$ delays jobs $[3], [4], \text{K}, [n]$. Calculate changes in the solution value and see whether the solution value improves. Delay job $[2]$ only if we can improve the solution value. Otherwise, go back to the status right before the heuristic tries to delay job $[2]$.

The heuristic repeats this process for job $[3]$. Note that for job $[3]$, the heuristic needs to include possible decrease in

solution value due to free delay which is caused by delaying job $[3]$. The heuristic repeats this process for all remaining jobs until job $[n-\gamma]$. We now formally describe the heuristic.

Heuristic H2.

0. Reindex jobs so that $p_{1j} + p_{2j} \le p_{1,j+1} + p_{2,j+1}$ for $j = 1,2,\text{K}\ ,n-1$.

Set $\gamma = \lfloor w_3 / w_1 \rfloor$

1. Schedule job 1 first so that $C_{11} = p_{11}$ and $C_{21} = p_{11} + p_{21}$.

2. If $\gamma \ge n-1$, then set $\gamma = n-1$ and go to Step 4.

3. Schedule jobs $2,3,\text{K}\ ,n$ in their index order on M_1 and M_2 such that $C_{1j} = C_{1,j-2} + p_{1j}$ and $C_{2j} = \max\{C_{2,j-1}, C_{1j}\} + p_{2j}$.

When there exist ties, break them arbitrarily.

4. Schedule jobs $n-\gamma+1, n-\gamma+2,\text{K}\ ,n$ in their index order on M_1 and M_2 such that $C_{1j} = \max\{C_{1,j-1} + p_{1j}, C_{2,j-1}\}$ and $C_{2j} = C_{1j} + p_{2j}$.

When there exist ties, break them arbitrarily.

5. Perform free delay for jobs $[2],[3],\text{K}\ ,[n-\gamma+1]$.

6. Set $k = 2$ and $\sigma =$ current schedule.

7. If $S_{2[k]} - C_{1[k]} \le 0$, then go to Step 12.

8. Delay job $[k]$ on machine 1 by $S_{2[k]} - C_{1[k]}$.

Delay jobs $k+1, k+2,\text{K}\ ,n$ on machine 1 by $S_{2[k]} - C_{1[k]}$.

Set $C_{2[\lambda]} = \max\{C_{1[\lambda]}, C_{2[\lambda-1]}\} + p_{2,\lambda+1}$ for $\lambda = k+1, k+2,\text{K}\ ,n$.

9. Perform free delay for jobs $2,3,\text{K}\ ,k-1$ if necessary.

10. Set σ' to be this new schedule.

11. If $\sum_{j=1}^{n} WIP_j(\sigma) > \sum_{j=1}^{n} WIP_j(\sigma')$, then set $\sigma = \sigma'$.

12. Set $k = k+1$.

If $k = n-\gamma+1$, then go to Step 13. Otherwise, go to Step 7.

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

In Step 0, reindexing the jobs requires $O(n \log n)$ time. All the other operations require $O(n)$ time. Steps 7 to 12 can be repeated up to $O(n)$ time, the time requirement of H2 is $O(n^2)$ time.

7.2 An Upper Bound on the Relative Error

Since H2 starts with the result of Heuristic H0 and only improves the schedule, H2 dominates H0. Hence, we have the following remark.

Remark 2 *For problem* $F2 \,||\, \sum WIP_j$ *, heuristic H2 dominates heuristic H1.*

Next, the following theorem shows that H2 dominates H0.

Theorem 3 *For problem* $F2 \,||\, \sum WIP_j$ *, heuristic H2 dominates heuristic H0.*

Proof. Let σ^{H1} and σ^{H2} be schedules generated by heuristics H0 and H2, respectively. By construction of H0 and H2, $C_j(\sigma_1^{H0}) \ge C_j(\sigma_1^{H2})$ and $C_j(\sigma_2^{H0}) \ge C_j(\sigma_2^{H2})$ for all $j \in N$. Recall that job sequence is the same for σ^{H0} and σ^{H2} so that jobs are sequenced in an increasing order of their processing times. Notice that for σ^{H0} and σ^{H2}, there does not exist any free delay. Consider a job such that $C_j(\sigma_1^{H0}) > C_j(\sigma_1^{H2})$ for $j \in N$. Let k be the first such a job. Then, it must be that $S_k(\sigma_2^{H2}) > C_k(\sigma_1^{H2})$. We delay job k in σ^{H2} by $S_k(\sigma_2^{H2}) - C_k(\sigma_1^{H2})$ on machine 1 and similarly, we delay jobs $k+1, k+2,\text{K}\ ,n$ by $S_k(\sigma_2^{H2}) - C_k(\sigma_1^{H2})$ on machine 1 and delay the same jobs accordingly on machine 2. Then, $C_k(\sigma_1^{H0}) - C_k(\sigma_1^{H2})$.

By construction of H2, this delay increases the solution value. Also, note that since σ^{H2} for jobs $1,2,\text{K}\ ,k$ is the same as σ^{H0}, there is no free delay for jobs $2,3,\text{K}\ ,k-1$.

We repeat this process for all remaining jobs such that $C_j(\sigma_1^{H0}) > C_j(\sigma_1^{H2})$ for $j \in N$. During the entire process, the solution value increases or remains the same. Therefore, H2 dominates H0. □

The following corollary establishes an upper bound on the relative error which is tight in limit.

Corollary 1 *For problem* $F2 \| \sum WIP_j$, $z^{H2} / z^* \le 2\beta/(\alpha+\beta)$, *and this bound is tight in limit where* z^{H2} *is a solution value of H1.*

Proof. The result follows from Theorems 1 and 3. □

7. Summary and Discussions

We have explored a flow-shop scheduling problem where a different WIP (work-in-process) state has different weight on the duration time. The objective is to minimize the sum of the weighted WIP. For the two machine flow shop case, the three simple and intuitive heuristics H0, H1, and H2 are presented for the problem. For each heuristic, we find an upper bound on relative error which is tight in limit. For heuristic H2, we show that H2 dominates the other two heuristics.

For future research, we want to develop more heuristics and find some special cases where heuristics can find optimal schedules for the problem. 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

Adiri, I and N. Amit (1984), Openshop and Flowshop Scheduling to Minimize Sum of Completion Times, *Computers and Operations Research*, 11(3), 275-284.

Conway, R. W., Maxwell, W. L., and Miller, L. W. (1967), Theory of Scheduling, Addision-Wesley, Reading, MA.

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.

Lee, W.-C. and C.-C. Wu (2001), Minimizing the Total Flow Time and Tardiness in a Two-machine Flow Shop, *International Journal of Systems Science*, 32(3), 365-373.

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, *Operations Research*, 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.

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.

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.

Sipper, D and R. Shapira (1989), JIT Vs. WIP - a Trade-off

Analysis, *International Journal of Production Research*, 27, 903-914.

Velde, S. L. V. D. (1992), Minimizing the Sum of the Job Completion Times in the Two-machine Flow Shop by Lagrangian Relaxation, *Annals of Operations Research*, 26, 257-268.

Vollman, T. E., W. L. Berry, and D. C. Whybark (1997), Manufactuting Planning and Control System, McGraw-Hill, New York, NY.

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

Yang, J. (2005), Flow-shop Scheduling with Weighted Work-In-Process, *Proceedings of the IEMS2005*, 199-206, Chungbuk National University, 2005