ANALYSIS FOR MINIMIZING WEIGHTED MEAN FLOW-TIME IN FLOW-SHOP SCHEDULING

Shigeji Miyazaki
and
Noriyuki Nishiyama
University of Osaka Prefecture

(Received May 11, 1979; Revised February 7, 1980)

Abstract This paper deals with the problem of minimizing the weighted mean flow-time in \( n/m \) flow-shop scheduling where no passing is allowed. Analysis, through the adjacent pairwise interchange method, leads to a condition for determining the precedence relation between adjacent jobs. The condition consists of inequalities, the number of which equals the square of the number of machines. An algorithm based on these inequalities is proposed to obtain the optimal or near optimal solution. The numerical examples show that the algorithm can produce a solution which has an average approximation ratio of 91.4 percent over 160 problems. The three factors: the number of jobs, the number of machines and the range of weights do not affect the approximation ratio of the tested problems. The computational time required to obtain a solution through the proposed algorithm is proportional to (the number of jobs) \( \times (\text{the number of machines})^2 \). As a result, the CPU time needed to solve a seven job and six machine problem through TOSBAC 5600/120 is 0.25 sec.

1. Introduction

There have been many theoretical studies on flow-shop scheduling \([1 \sim 5, 8 \sim 15, \text{etc.}]\). The performance measures considered in these papers are mainly concentrated on maximal flow-time. In the previous paper \([8]\), we investigated the minimization of mean flow-time in \( n/m \) flow-shop scheduling by means of adjacent pairwise interchange method. The paper presented sufficient conditions to decide the precedence relations between adjacent pairwise jobs. On the basis of the conditions, a computational algorithm was proposed for an optimal or near optimal solution.
The model studied in the paper [8], however, takes no account of job importance. In many situations, the jobs do not have equal importance. For instance, the earlier due-dates are, and the higher inventory costs are, the jobs should be regarded as more important objects for scheduling. This paper introduces the weighting factor \( w_i \) to each job (the larger \( w_i \), the higher priority of job) and deals with the problem of minimization of weighted mean flow-time. A computational algorithm is presented for an optimal or near optimal solution on the basis of adjacent pairwise approach. The efficiency of the algorithm is verified by means of numerical experiment.

2. Flow-Shop Model

2.1 Definition and Notation of the Model

The discussed model can be stated as follows:

1) Let \( n \) be the number of jobs to be processed, and \( i \)th job in the arbitrary sequence \( S \) is denoted by \( J_i \) where \( i = 1, 2, \ldots, n \). All these jobs are available for processing at time zero.

2) The manufacturing system consists of \( m \) different machines which are numbered according to the order of production stage. Let \( M_j \) be the \( j \)th machine in the system where \( j = 1, 2, \ldots, m \). Every machine is continuously available. A machine can process only one job at a time.

3) Every job is completed through the same production stage that is \( M_1 \rightarrow M_2 \rightarrow \ldots \rightarrow M_m \).

4) Let \( p_{i,j} \) denote the processing time of \( J_i \) on \( M_j \). Setup times for operations are sequence-independent and are included in processing times. Handling times are assumed to be so limited that they can be neglected.

5) Let \( F_{j}(i) \) denote the partial flow-time of \( J_i \) counted from the starting time of first job \( J_1 \) on \( M_1 \) to the completion time of \( J_i \) on \( M_j \), referring to Fig. 1. In particular, \( F_m(i) \) is called as flow-time of \( J_i \).
6) The same job sequence occurs on each machine; in other words, no passing is allowed in the shop.

7) Each job is assigned weight \( w_i \) according to its importance.

![Diagram of machine sequences and time](image)

**Fig. 1.** Definition of \( F_j(i) \).

### 2.2 Performance Measure

The performance measure studied is weighted mean flow-time defined by:

\[
\bar{F}_w = \frac{1}{n} \sum_{i=1}^{n} w_i F(m(i)).
\]

This measure can be redefined as:

\[
\bar{F}_w = \frac{\sum_{i=1}^{n} w_i F(m(i))}{\sum_{i=1}^{n} w_i}.
\]
Each definition produces the same solution, since the denominators of (1.1) and (1.2) are sequence-independent. From (1.1) we have:

\[(1.3) \quad n\overline{F}_w = \sum_{i=1}^{n} w_i F_m(i),\]

where \(n\overline{F}_w\) expresses the total weighted flow-time. \(n\overline{F}_w\) shall be used in place of \(\overline{F}_w\) in the further analysis.

3. Analysis

In the sequence \(S\), let \(s\) be a subsequence consisting of the first \(q-1\) jobs, that is, \(J_1, J_2, \ldots, J_{q-1}\), and in succession to \(s\), \(J_q\) and \(J_{q+1}\) (these two jobs are called adjacent two jobs hereafter) are assumed to be processed in the order \(J_q, J_{q+1}\). Now consider the sequence \(S'\) in which \(J_q\) and \(J_{q+1}\) are pairwise interchanged and are processed in the order \(J_{q+1}, J_q\). The sequence is the same for the first \(q-1\) jobs and the last \((n-q-1)\) jobs under either \(S\) or \(S'\) as illustrated in Fig. 2.

\[
S = J_1, J_2, \ldots, J_{q-1}, J_q, J_{q+1}, \ldots, J_n
\]

\[
S' = J_1, J_2, \ldots, J_{q-1}, J_{q+1}, J_q, \ldots, J_n
\]

Fig. 2. The Relationship between Sequence \(S\) and \(S'\)

In order to distinguish the notation of partial flow-time under \(S\) from \(S'\), let \(F_j(q)\), \(F_j(q+1)\), and \(F_j(i)\) \((i=q+2, q+3, \ldots, n)\) denote the partial flow-time of \(J_q, J_{q+1}\), and \(J_i\) \((i=q+2, q+3, \ldots, n)\) under \(S\) in turn, and let \(F'_j(q+1), \overline{F}_j(q+1, q)\), and \(F'_j(i)\) \((i=q+2, q+3, \ldots, n)\) denote the partial flow-time of \(J_{q+1}, J_q\) and \(J_i\) \((i=q+2, q+3, \ldots, n)\) under \(S'\) in turn, moreover let \(\overline{F}_w'\) be the weighted mean.
flow-time under $S'$. Then the total weighted flow-times under $S$ and $S'$ are expressed by:

(3.1) \[ n\bar{F}_w = \sum_{i=1}^{q-1} W_{i,m}(i) + W_{q,m}(q) + W_{q+1,m}(q,q+1) \]
\[ + \sum_{i=q+2}^{n} W_{i,m}(i), \]

and

(3.2) \[ n\bar{F'}_w = \sum_{i=1}^{q-1} W_{i,m}(i) + W_{q+1,m}(q+1) + W_{q,m}(q+1,q) \]
\[ + \sum_{i=q+2}^{n} W_{i,m}(i). \]

Eliminating the common terms between (3.1) and (3.2) from the each equation, and denoting the remaining, $<n\bar{F}_w>$ and $<n\bar{F'}_w>$, respectively, we have:

(3.3) \[ <n\bar{F}_w> = W_{q,m}(q) + W_{q+1,m}(q,q+1) + \sum_{i=q+2}^{n} W_{i,m}(i), \]

and

(3.4) \[ <n\bar{F'}_w> = W_{q+1,m}(q+1) + W_{q,m}(q+1,q) + \sum_{i=q+2}^{n} W_{i,m}(i). \]

If

(3.5) \[ <n\bar{F}_w> < <n\bar{F'}_w>, \]

that is:

(3.6) \[ n\bar{F}_w < n\bar{F'}_w, \]

holds, $J_{q+1}$ cannot directly precede $J_{q}$ in the optimal sequence. Therefore, we shall investigate the sufficient conditions, which have transitive property of job ordering, for satisfying (3.5) independently of the two jobs' position, as shown in the following:

Comparing each term of (3.3) with the corresponding term of (3.4), we have:

(3.7) \[ W_{q,m}(q) \leq W_{q+1,m}(q+1), \]

(3.8) \[ W_{q+1,m}(q) \leq W_{q+1,m}(q+1), \]

and

(3.9) \[ \sum_{i=q+2}^{n} W_{i,m}(i) \leq \sum_{i=q+2}^{n} W_{i,m}(i), \]

which are to be sufficient conditions for (3.5).
There exist next recurrence relations on partial flow-time $F_j(i)$, referring to Fig. 1.

\[
(3.10) \quad F_j(i) = \max \{F_{j-1}(i), F_j(i-1)\} + P_{i,j},
\]

\((i = 1, 2, \ldots, n; j = 1, 2, \ldots, m),\)

where $F_j(0) = 0$, $F_j(n) = 0$.

Working out the recurrence relations (3.10), we have:

\[
(3.11) \quad F_j(i) = \max_{r=1}^{m} \{F_{j-r+1}(i-1) + \Sigma_{t=1}^{r} P_{i,j-t+1}\}.
\]

Substituted into (3.7), (3.11) gives

\[
(3.12) \quad W \leq \max_{q=1}^{m} \{F_{m-r+1}(q-1) + \Sigma_{r=1}^{m} P_{q,m-r+1}\}.
\]

The comparison between the respectively corresponding terms of (3.12) gives the following sufficient conditions of (3.7):

\[
(3.13) \quad W < W_{q+1},
\]

and

\[
(3.14) \quad W \Sigma_{t=1}^{r} P_{q,m-t+1} \leq W_{q+1} \Sigma_{t=1}^{r} P_{q+1,m-t+1}, \quad (r = 1, 2, \ldots, m).
\]

Now the partial flow-time $F_j(i,i+1)$ is given as similar to (3.10),

\[
(3.15) \quad F_j(i,i+1) = \max \{F_{j-1}(i,i+1), F_j(i)\} + P_{i+1,j},
\]

\((i = 1, 2, \ldots, n-1; j = 1, 2, \ldots, m),\)

where $F_j(0,i+1) = 0$, $F_j(n,i+1) = 0$.

Working out the recurrence relation (3.15), we have:

\[
(3.16) \quad F_j(i,i+1) = \max_{r=1}^{m} \max_{t=1}^{r} \{F_{j-r+1}(i-1) + \Sigma_{k=1}^{r} P_{i+1,k} + \Sigma_{k=1}^{t} P_{i,j-k+1}\}.
\]

Substituted into (3.8), (3.16) gives

\[
(3.17) \quad W \leq \max_{q=1}^{m} \max_{t=1}^{r} \{F_{m-r+1}(q-1) + \Sigma_{j=m-r+1}^{m} P_{q,j} + \Sigma_{j=m-t+1}^{m} P_{q,j}\}.
\]
The comparison between the respectively corresponding terms of (3.17) gives:

\[(3.18) \quad W_q \geq W_{q+1},\]

\[(3.19) \quad W_q \sum_{j=m-t+1}^{m} P_{q,j} \geq W_{q+1} \sum_{j=m-t+1}^{m} P_{q+1,j},\]

\[(t = 1, 2, \ldots, m),\]

and

\[(3.20) \quad \left(\sum_{j=m-r+1}^{m} P_{j} / W_q\right) \leq \left(\sum_{j=m-r+1}^{m} P_{q+1,j} / W_{q+1}\right),\]

\[(r = 1, 2, \ldots, m; t = 1, 2, \ldots, r).\]

If

\[(3.21) \quad F_m(i) \leq F'_m(i), (i = q+2, q+3, \ldots, n)\]

hold, (3.9) should be satisfied. Moreover, Yueh [15] shows that

\[(3.22) \quad \min \left(P_{q,u}, P_{q+1,v}\right) \leq \min \left(P_{q+1,u}, P_{q,v}\right), (1 \leq u < v \leq m)\]

is the sufficient condition of (3.21) that is (3.9).

The discussion above has led the sufficient conditions of (3.7), (3.8) and (3.9) individually. Since all of these sufficient conditions have transitive property, the temporary sequence can be induced from each sufficient condition. In the case that all of these temporary sequences are equal to each other, the sequence is the optimal solution for this problem. According to the following algorithm an optimal solution can be produced in this case. In the usual cases in which all of the temporary sequences do not coincide with one another, a suboptimal solution can be obtained through the same algorithm.

Considering that (3.5) is composed of the sum of (3.7), (3.8), and (3.9), we make, in the algorithm, a solution by the procedure that calculates the sum of the ordinal numbers according to the temporary sequences. Between two inequalities (3.13) and (3.18) the expressions of the both sides are identical but only the sign of inequality is opposite. Such is the case between inequalities (3.14) and (3.19) too. The sum of the ordinal numbers derived from four inequalities: (3.13), (3.14), (3.18), and (3.19) becomes always equal to
each other job. Therefore, we can eliminate these four inequalities in the following algorithm from the beginning.

4. Algorithm

The algorithm will be explained by solving an example problem listed in Table 1.

Step 1. Decide the \( m(m+1)/2 \) kinds of temporary sequences which can be led from (3.20) as follows: calculate the value \( \frac{\sum_{j=m-r+1}^{m} p_j}{w_j} \) of all jobs for each combination of \( r(=1,2,\ldots,m) \) and \( t(=1,2,\ldots,r) \), as tabulated in Table 2. Make the temporary sequences in accordance with the non-decreasing order of each row value in Table 2. Assign an integer to each job according to its order, as shown in Table 3. In case more than two jobs have the same value in a row, assign the same integers to them.

Step 2. Make the temporary sequence in which all jobs satisfy (3.22) for each combination of \( u \) and \( v \), using Johnson's Algorithm [5]. Assign an integer to each job as similar to Step 1. The results of this is indicated in Table 4. This step produces \( m(m-1)/2 \) kinds of temporary sequences.

Step 3. Calculate the sum of integers assigned to each job in the Step 1 and 2 as Table 5. Arrange each job in the nondecreasing order of the total integers. Break a tie by placing jobs with lower original numbers first.

The solution for this example becomes \( J_1-J_4-J_2-J_3 \).
Table 1. Four Job Three Machine Problem.

<table>
<thead>
<tr>
<th>Job</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
<th>( J_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Weight</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2. The Value of \( \left( \sum_{j=m-r+1}^{m-t} \frac{p_{r,j}}{w_j} \right) / \omega_j \).

<table>
<thead>
<tr>
<th>Job</th>
<th>( r )</th>
<th>( t )</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
<th>( J_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6/5</td>
<td>6/5</td>
<td>4/3</td>
<td>7/7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2/6</td>
<td>13/5</td>
<td>10/3</td>
<td>10/7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6/3</td>
<td>8/5</td>
<td>6/3</td>
<td>3/7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7/6</td>
<td>18/5</td>
<td>13/3</td>
<td>16/7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5/6</td>
<td>13/5</td>
<td>9/3</td>
<td>9/7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2/6</td>
<td>5/5</td>
<td>3/3</td>
<td>6/7</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Ordinal Numbers by Step 1.

<table>
<thead>
<tr>
<th>( r )</th>
<th>( t )</th>
<th>Job</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
<th>( J_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2/6</td>
<td>5/5</td>
<td>4/3</td>
<td>7/7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3/4</td>
<td>2</td>
<td>2</td>
<td>3/4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3/4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3/4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3/4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Ordinal Numbers by Step 2.

<table>
<thead>
<tr>
<th>( u )</th>
<th>( v )</th>
<th>Job</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
<th>( J_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3/4</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3/4</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Sum of Ordinal Numbers.

<table>
<thead>
<tr>
<th>Job</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
<th>( J_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum of ordinal numbers</td>
<td>13</td>
<td>25</td>
<td>30</td>
<td>20</td>
</tr>
</tbody>
</table>

5. Efficiency of the Algorithm

5.1 Approximation Ratio

The definition of the approximation ratio, to evaluate the quality of the solution, used in this paper, is different from that often used in previous papers [3, 9, etc.]. Previously, the approximation ratio was simply defined by:
(5.1)  \( \eta_1 = 100 \times \left( \frac{o}{a} \right), \quad (%) \)

where \( o \) and \( a \) are the values of performance measures of the optimal and obtained solutions, respectively. This ratio, however, does not take into consideration the existing range of possible solutions. Consequently, it has the following shortcomings: Suppose that there exist two flow-shop scheduling problems I and II of which possible solutions are distributed as shown in Fig. 3. If the obtained solutions \( a_I \) and \( a_{II} \) for each problem have a equal value of performance measure, the approximation ratio defined by \( \eta_1 \) indicates the same percentage. The quality of \( a_{II} \), however, is practically higher than \( a_I \), as the existing range of possible solutions for problem II is wider than that for problem I. Moreover, it is a shortcoming of \( \eta_1 \) that it indicates a percentage greater than zero even if the obtained solution coincides with the worst possible solution.

Fig. 3. Distribution of Possible Solutions for Problem I and II.
We defined the approximation ratio as:

\[ \eta_2 = 100 \times \frac{b-a}{b-o} \] (\%)

where \( b \) is the value of worst possible solution [8]. This ratio contains the optimal and the worst value of performance measure so as to reach 0\% in case the obtained solution coincides with the worst one, and 100\% in case the obtained solution coincides with the optimal one.

5.2 Computational Experience

In order to verify the efficiency of the algorithm, the example problems composed of 4 ~ 7 jobs and 4 ~ 6 machines are solved through the proposed algorithm and the solutions are evaluated by approximation ratio \( \eta_2 \). Table 6 shows the results of this evaluation together with \( \eta_1 \) for references. The processing times in the example problems are distributed uniformly between 1 and 99, and the weights assigned to jobs are distributed uniformly between 1 and 10 or 1 and 40. The optimal and worst solutions to calculate \( \eta_2 \) were obtained from complete enumeration method.

Table 6. Results of the Experiment

<table>
<thead>
<tr>
<th>Problems</th>
<th>Approximation ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \eta_1 )</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>( n ) ( m )</td>
<td>( p_i )</td>
</tr>
<tr>
<td>4 4 1 ~ 99</td>
<td>1 ~ 10</td>
</tr>
<tr>
<td></td>
<td>1 ~ 40</td>
</tr>
<tr>
<td>5 5 1 ~ 99</td>
<td>1 ~ 10</td>
</tr>
<tr>
<td></td>
<td>1 ~ 40</td>
</tr>
<tr>
<td>6 6 1 ~ 99</td>
<td>1 ~ 10</td>
</tr>
<tr>
<td></td>
<td>1 ~ 40</td>
</tr>
<tr>
<td>7 6 1 ~ 99</td>
<td>1 ~ 10</td>
</tr>
<tr>
<td></td>
<td>1 ~ 40</td>
</tr>
<tr>
<td>Grand average</td>
<td>96.2</td>
</tr>
</tbody>
</table>
The results of Table 6 indicate that the average approximation ratio $\eta_2$ is 91.4% and that none of the three factors, the number of jobs, the number of machines, and the range of weights affect the average approximation ratio for the tested problems. The minimal approximation ratio becomes larger as the number of jobs and the number of machines increase.

Table 7. Mean Computational Times
(CPU Time, sec.)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Proposed method</th>
<th>Complete enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ $m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 4</td>
<td>0.09</td>
<td>0.20</td>
</tr>
<tr>
<td>5 5</td>
<td>0.14</td>
<td>0.80</td>
</tr>
<tr>
<td>6 6</td>
<td>0.21</td>
<td>2.50</td>
</tr>
<tr>
<td>7 6</td>
<td>0.25</td>
<td>16.0</td>
</tr>
</tbody>
</table>

Table 7 shows the mean computational times of TOSBAC-5600/120 required to obtain a solution through the proposed algorithm and complete enumeration, respectively. Little time variation occurs in solving the problem which has the same job number and machine number, through both methods. The structure of the algorithm should lead the computational times to be proportional to $(\text{the number of jobs}) \times (\text{the number of machines})^2$.

The number of machines, which affects the computational time quadratically, has an upper limitation practically, for it coincides with the number of operations needed to complete a job in a flow-shop. Data from an actual machining shop indicate that over ninety-five percent of jobs are produced through the operation stages less than 11 [7]. Although the number of jobs becomes considerably large in practical shop, the computational time of the algorithm goes up just linearly with the number of jobs. The discussion above shows that the algorithm is much effective than general purpose optimizing techniques such as B. & B. method [14] or D.P. [6] from the viewpoint of computational times.
The required memory capacity should never become a major limitation in executing the algorithm, as it needs only 68K for solving a 1000 job and 25 machine problem.

6. Conclusion

In this paper, we dealt with the minimization of weighted mean flow-time problem in \(n/m\) flow-shop scheduling. A computational algorithm was proposed through an adjacent pairwise approach. In order to evaluate the quality of the solution, we used the new approximation ratio \(\eta_2\) derived from the discussion of the previous approximation ratio \(\eta_1\). The algorithm produces the solution which has 91.4\% of average approximation ratio \(\eta_2\). The computational time for the algorithm is proportional to \((\text{the number of jobs}) \times (\text{the number of machines})^2\). As a result, the CPU time needed to solve a seven job and six machine problem through TOSBAC 5600/120 is below 0.3 sec. Memory capacity should never be the major restriction on solving practical problems.

References


Sigeji MIYAZAKI: Department of Industrial Engineering, College of Engineering, University of Osaka Prefecture, Mozu-Umemachi, Sakai, Osaka, 591 Japan.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.