Minimizing Rental Cost in 2-Machine Flow-shop Problem under Unavailability Constraint

Laxmi Narain

(Associate Professor, Department of Mathematics, Acharya Narendra Dev College, University of Delhi, Delhi, India)

Abstract

This paper considers n-job, 2-machine flow-shop problems under rental policy $RP_2$ with unavailability constraint on each machine. Under rental policy $RP_2$, all the machines are taken on rent at one time in the starting of processing the jobs and are returned as and when a machine is no longer required for processing the jobs. Under unavailability constraint, it is assumed that unavailable time is known in advance (deterministic) and if a job cannot finish before the unavailable period of a machine then job can continue after the machine is available again (resumable). To solve the problem optimally, an algorithm is developed using Branch-and-Bound technique and Johnson's algorithm. The Algorithm is illustrated through a numerical example.

Keywords: Flow-shop, Scheduling, Deterministic, Resumable, Rental Cost

1. INTRODUCTION

In real life, situation may arise when one has got the assignment but does not have one’s own machines and does not have money for the purchase of machines. Under such circumstances, may take machines on rent to complete the assignment. Minimization of total rental cost of machines will be the criterion in such situation. Generally, the following renting policies exist:

$RP_1$ : All the machines are taken on rent at one time in the starting of processing the jobs and are returned at one time when the last job completes processing on the last machine.
RP_2 : All the machines are taken on rent at one time in the starting of processing the jobs and are returned as and when they are no longer required.

RP_3 : All the machines are taken on rent as and when they are required and are returned as and when they are no longer required for processing.

Authors [3, 4, 8, 9, 11, 12, 13] study these rental policies with the assumptions that machines are available at all the times. However, this availability may not be true in real industry settings. For example, machines may not be available during the scheduling period due to breakdown (stochastic) or preventive maintenance (deterministic).

Authors [1, 2, 6, 10] studied flow-shop problem under unavailability constraint to optimize different objective functions. Adiri et al. [1] studied a single-machine problem with machine breakdown. They studied both the stochastic case and the deterministic case. For the deterministic case, the objective function was minimization of total completion time under the assumption that a job must be restarted if it did not finish before the breakdown. Lee [6] studied the two-machine flow-shop problem in deterministic environment under two cases; availability constraint on machine M_1 only, and on machine M_2 only. He provided pseudo-polynomial dynamic programming algorithm to solve the problem optimally. Narain [10] provided an algorithm to minimize total elapsed time for two-machine flow-shop problem under an unavailability constraint on each machine.

In this paper, we consider n-job, 2-machine flow-shop scheduling problems under rental policy RP_2 with an availability constraint on each machine. Under rental policy RP_2, both machines will be taken on rent in the starting of processing the jobs and each machine will be returned as and when it is no longer required. Under unavailability constraint it is assumed that the unavailable times are known in advance (deterministic) and if a job cannot finish before the unavailable period of a machine then job can continue after the machine is available again (resumable).

An algorithm is developed, using Branch-and-Bound technique and Johnson's algorithm, to solve the problem optimally. Algorithm is illustrated through a numerical example.

2. NOTATIONS

S : Sequence of jobs 1, 2, ..., n.
M_j : Machine j, j = 1, 2.
s_j, t_j : Machine M_j is unavailable from s_j to t_j, where 0 ≤ s_j ≤ t_j.
p_{ij} : Processing time for job i on machine M_j.
Minimizing Rental Cost in 2-Machine Flow-shop Problem...

\[ I_{ij} \]: Idle time of machine \( M_j \) for job \( i \).

\[ C_j \]: Rental cost of machine \( M_j \).

\( J_r \): Partial sequence of \( r \) scheduled jobs.

\( J'_r \): Set of remaining \((n-r)\) free jobs.

\( i_1 \): 1st job of partial schedule \( J_r \).

\( t(J_r, j) \): The time when the last job of the partial schedule \( J_r \) is completed on machine \( M_j \), where \( j = 1, 2 \).

\( t(S, j) \): The time when the last job of sequence \( S \) is completed on machine \( M_j \), where \( j = 1, 2 \).

\( Z_{\text{max}} \): Total elapsed time.

\( LB[J_r] \): Lower bound corresponding to partial schedule \( J_r \), irrespective of any schedule of \( J'_r \).

\( F2//Z_{\text{max}} \): Minimizing the total elapsed time in the 2-machine flow-shop problem.

\( F2/r-a(M_j)/Z_{\text{max}} \): Minimizing the total elapsed time in the 2-machine flow-shop problem with a resumable availability constraint on machine \( M_j \).

3. MATHEMATICAL FORMULATION

The traditional two-machine flow-shop problem, \( F_2/Z_{\text{max}} \), can be solved optimally by Johnson’s Algorithm. We first review it in the following.

**Johnson's Rule:** Job \( i \) precedes job \( j \) if \( \min (p_{i1}, p_{j2}) < \min (p_{i2}, p_{j1}) \). If \( \min (p_{i1}, p_{j2}) = \min (p_{i2}, p_{j1}) \), we can break the tie arbitrarily.

**Johnson's Algorithm:** Divide the \( n \)-job set into two disjoint subsets, \( S_1 \) and \( S_2 \), where \( S_1 = \{i; p_{i1} \leq p_{i2}\} \) and \( S_2 = \{i; p_{i1} > p_{i2}\} \). Order the jobs in \( S_1 \) in the non-decreasing order of \( p_{i1} \) and those jobs in \( S_2 \) in the non-increasing order of \( p_{i2} \). Sequence jobs in \( S_1 \) first, followed by \( S_2 \).

**Lemma 3.1:** Johnson's Algorithm is optimal for \( F2/r-a/Z_{\text{max}} \) if \( s_1 = s_2 = 0 \).

**Proof:** This can be proved by adding one job with \( p_{n+1,1} = 0, p_{n+1,2} = \max \{0, t_2 - t_1\} \), to the traditional \( F2/Z_{\text{max}} \) problem and shifting the problem to start at \( t_1 \). It is explained in Lee [6].

**Lemma 3.2:** There exist an optimal solution such that the set of jobs after job \( k \) should follow Johnson's Rule, where job \( k \) is the first job such that \( C_{kj} \geq \max \{t_1, t_2\} \).

**Proof:** The set of jobs after job \( k \) should follow Johnson's rule in an optimal solution by Lemma 3.1.
For sequence $S$, total rental cost of machines

$$\text{TRC}(S) = \sum_{j=1}^{2} \sum_{i=1}^{n} (p_{ij} + I_{ij}) \times C_{j}$$

$$= \sum_{i=1}^{n} [(p_{i1} + I_{i1}) \times C_{1} + (p_{i2} + I_{i2}) \times C_{2}]$$

$$= \sum_{i=1}^{n} p_{i1} \times C_{1} + \sum_{i=1}^{n} I_{i1} \times C_{1} + \sum_{i=1}^{n} p_{i2} \times C_{2} + \sum_{i=1}^{n} I_{i2} \times C_{2}$$

Under $R_{P2}$, $\sum_{i=1}^{n} I_{i1} = t_{1} - s_{1}$, which is constant.

Therefore, total rental cost of machines

$$\text{TRC}(S) = \sum_{i=1}^{n} p_{i1} \times C_{1} + \sum_{i=1}^{n} I_{i1} \times C_{1} + \sum_{i=1}^{n} p_{i2} \times C_{2} + \sum_{i=1}^{n} I_{i2} \times C_{2}$$

Here processing times $p_{ij}$ and rental costs $C_{j}$ are constants $\forall i$ and $j$. Therefore, total rental cost of machines is minimum when $\sum_{i=1}^{n} I_{i2} \times C_{2}$ is minimum.

### Evaluation of Lower Bound

The lower bound for any partial schedule is the $VALUE$ so obtained so that whatever be the order of the remaining jobs to follow that schedule the total rental cost of machines should never be the less than the $VALUE$. The lower bound for any partial schedule $J_{r}$ is obtained under the assumption that jobs of $J_{r}'$ does not wait for processing on particular machine and jobs after completing the processing on this machine are not waiting for processing on the remaining machine as if the machines are always available for processing. This reduces the problem of a single machine processing and the criterion for obtaining the optimal order of single machine can be obtained.

### 4. ALGORITHM

The branch-and-bound technique is explained in Lomnicki [7]. The lower bounds for any partial schedule $J_{r}$ are evaluated through the following steps:

**Step 1:** Compute

(i) $G_{1} =
\begin{cases}
\begin{align*}
t(J_{r}, 1) + \sum_{i \in J_{r}'} p_{i1} + \min_{i \in J_{r}'} p_{i2} + (t_{1} - s_{1}); & \text{if } t(f_{r}, 1) \leq s_{1} \\
t(J_{r}, 1) + \sum_{i \in J_{r}'} p_{i1} + \min_{i \in J_{r}'} p_{i2}; & \text{otherwise}
\end{align*}
\end{cases}$

(ii) $G_{2} =
\begin{cases}
\begin{align*}
t(J_{r}, 2) + \sum_{i \in J_{r}'} p_{i2} + (t_{2} - s_{2}); & \text{if } t(f_{r}, 2) \leq s_{2} \\
t(J_{r}, 2) + \sum_{i \in J_{r}'} p_{i2}; & \text{otherwise}
\end{align*}
\end{cases}$
Step 2: Compute
\[ \text{LB}[J_r] = \max \{G_1, G_2\} \times C_2 \]

Step 3: Repeat Step 1 and Step 2 till we get \( J_r \) such that \( \text{LB}[J_r] \) is minimum and \( t(J_r, 1) \geq \max (t_1, t_2) \)

Step 4: Get \( J_r'' \) by sequencing all the jobs in \( J_r' \) based on Johnson’s rule. Sequence jobs in \( J_r' \) first, followed by \( J_r'' \) and calculate total work duration \( g_1 \) (say).

Step 5: Get sequence \( S = J_r J_r'' \) by putting jobs in \( J_r' \) first, followed by \( J_r'' \).

Step 6: Compute \( \text{LB}[S] = t(S, 2) \times C_2 \)
If \( \text{LB}[S] \) is greater than any non-discarded vertex then Go to Step 1
Otherwise
\[ S \] is the optimal sequence and total rental cost
\[ = \text{LB}[S] + \sum_{i=1}^{n} p_{il} \times C_1 + (t_1 - s_1) \times C_1 \]

5. EXAMPLE
Consider a 4-job, 2-machine scheduling problem with processing times as given in Table: T1. It is also given that \( M_1 \) will not be available for 4 hours from 7 to 11 and \( M_2 \) will not be available for 2 hours from 8 to 10 i.e., \( s_1 = 7, t_1 = 11 \) and \( s_2 = 8, t_2 = 10 \). Rental cost of \( M_1 \) is Rs 100 per hour and of \( M_2 \) is Rs 200 per hour.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Table: T1

For \( J_r = (1) \),

Step 1: \( t(J_r, 1) = 7, \quad t(J_r, 2) = 7 + 9 + 2 = 18 \)

\[ G_1 = \begin{cases} \text{t(J, 1)} + \sum_{i \in J_r} p_{i1} + \min_{i \in J_r} p_{i2} + (t_1 - s_1) & ; \text{if } t(J_r, 1) \leq s_1 \\ \text{t(J, 1)} + \sum_{i \in J_r} p_{i1} + \min_{i \in J_r} p_{i2} & ; \text{otherwise} \end{cases} \]

\[ G_1 = 7 + (8+5+6) + \min \{7, 4, 7\} + (11 - 7) = 26 + 4 + 4 = 34 \]
Laxmi Narain

\[ \begin{align*}
G_2 &= \begin{cases} 
t(\mathbf{J}, 2) + \sum_{i \in \mathbf{J}'} p_{i2} + (t_2 - s_2) &; \text{if } t(\mathbf{J}, 2) \leq s_2 \\
t(\mathbf{J}, 2) + \sum_{i \in \mathbf{J}'} p_{i2} &; \text{otherwise}
\end{cases} \\
&= 18 + (7 + 4 + 7) \\
&= 18 + 18 = 36
\end{align*} \]

\textbf{Step 2: } \text{LB}[\mathbf{J}_r] = \max \{ G_1, G_2 \} \times C_2 \\
&= \max \{ 34, 36 \} \times 200 \\
&= 36 \times 200 = 7200

Applying the above steps for \( J_r = (2), (3), \) and \( (4), \) the relevant lower bounds evaluations are given in Table: \text{T2}.

<table>
<thead>
<tr>
<th>( J_r )</th>
<th>( t(J_r, 1) )</th>
<th>( \text{LB}[J_r] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7200</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>7400</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7400</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7000</td>
</tr>
</tbody>
</table>

Table: \text{T2}

Minimum value of lower bound is 7000 for \( J_r = (4). \) Therefore, \( J_r = (4) \) is the branching node.

Now take the partial schedule \( J_r = (41), (42), \) and \( 43), \) the relevant lower bounds evaluations are given in Table: \text{T3}.

<table>
<thead>
<tr>
<th>( J_r )</th>
<th>( t(J_r, 1) )</th>
<th>( \text{LB}[J_r] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>17</td>
<td>7400</td>
</tr>
<tr>
<td>42</td>
<td>18</td>
<td>7600</td>
</tr>
<tr>
<td>43</td>
<td>15</td>
<td>7400</td>
</tr>
</tbody>
</table>

Table: \text{T3}

Minimum value of lower bound is 7200 for \( J_r = (1). \) Therefore, \( J_r = (1) \) is the branching node.

Now for partial schedule \( J_r = (12), (13), \) and \( (14), \) the relevant lower bounds evaluations are given in Table: \text{T4}.

<table>
<thead>
<tr>
<th>( J_r )</th>
<th>( t(J_r, 1) )</th>
<th>( \text{LB}[J_r] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>19</td>
<td>7400</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>7400</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>7200</td>
</tr>
</tbody>
</table>

Table: \text{T4}

Minimum value of lower bound is 7200 for \( J_r = (14). \) Therefore, \( J_r = (14) \) is the branching node.
Step 3: For \( J_r = (14), t(J_r, 1) = 17 > \max \{11, 10\} \). Therefore, go to Step 4.

Step 4: \( J_r = (14), J'_r = \{2, 3\}, J''_r = (23) \).

Step 5: \( S = (1324) \)

Step 6: \( \text{LB}[S] = 7200 \) is minimum of all other non-discarded vertices. Therefore, total rental cost = 7200 + (26 + 4) x 100 = 7200 + 3000 = 10200

The complete scheduling tree is shown as in Figure: F1.

![Scheduling Tree](image)

**Figure:** F1 Scheduling Tree

Hence, 1-4-2-3 is the optimal sequence and total rental cost is Rs 10,200. Applying Johnson algorithm, we obtain the sequence 4-1-2-3 which has total rental cost as Rs 10,400. Hence, with un-availability constraint, Johnson algorithm may not give optimal sequence in n-job, 2-machine flow-shop problem.
6. FUTURE SCOPE
This paper studies 2-machine flow-shop problems with an availability constrain on each machine. This can be extended with more than one availability constraint on each machine. This can also be extended to general m-machine flow-shop problems.

REFERENCES
