ON THE BOUND OF MAKESPANS AND ITS APPLICATION IN M MACHINE SCHEDULING PROBLEM

ICHIRO NABESHIMA

University of Electro-Communications

(Received February 28, 1967)

Preface

For so-called m machine scheduling problem, recently there has been applied branch and bound method to min-makespan problem both in case where no passing is allowed [1] [2] [3] and in case where passing is allowed [4]. In those cases, to make more exact the value of the lower bound of makespans of sequences with definite presubsequence is very important in order to obtain an optimal solution using branch and bound algorithm (B.B. algorithm) by checking as smaller number of nodes as possible, under consideration on the quantity of calculations on the other hand.

In this paper, such more exact lower bound (revised lower bound) than already given in Refs. [1]~[4] is presented for each case by using Johnson's criterion for two machines [5] (§1) and, for the purpose of estimating better lower bound for the case where no passing is allowed, other devices for obtaining some different types of lower bound of makespans of sequences with definite backsubsequence and/or definite presubsequence will be shown with applications to B.B. algorithm (§2). Next, upper bound of makespans of sequences with definite presubsequence in case where no passing is allowed is presented with application to B.B. algorithm for max-makespan problem (§3). In each of these sections numerical examples will be shown in order to show the effectiveness of each bound. Finally, additional remarks will be shown, especially concerning the sensitivity of the "algorithm" for each problem (§4).
§1. Revised Lower Bound of Makespans of Sequences with Definite Presubsequence

In papers [1]~[4] already published up to now concerning branch and bound algorithm for optimal sequencing of \( n \) jobs through \( m \) machines along same machines order for each job, lower bound of makespans of sequences with definite presubsequence for each node has the terms that represent the sum of the processing time of each job belonging to set of unordered remained jobs at every machine where idle time of each machine by these processing doesn't been taken into account. But, estimation of this idle time can be taken into consideration by applying Johnson's criterion [5] for two machines case as shown in the following. Further let \( m \) machines be named by \( M_1, M_2, \ldots, M_m \) and be used in this order for any job and processing time of job \( i \) on \( M_k \) be \( m_{k,i} \) (\( i = 1 \sim n, k = 1 \sim m \)).

1.1. Case where no passing is allowed

1.1.1. Revised lower bound

First it's considered the case where no passing of job is allowed. Let \( J_r (r = 1 \sim n - 1) \) be a definite presubsequence of \( r \) jobs among \( n \) jobs that are processed on \( m \) machines and \( T_k(J_r) \) be the completion time of this sequence \( J_r \) on machine \( M_k (k = 1 \sim m, r = 1 \sim n - 1) \) and \( J_r \) be the set of all unordered remaining \( (n - r) \) jobs after the processing of \( J_r \).

Then, for each two machines \( M_k, M_{k+1} (k = 1 \sim m - 1) \) adjoining each other, let \( i_{r+1}, i_{r+2}, \ldots, i_n \) be the sequence of \((n-r)\) jobs in \( J_r \) which is determined by next Johnson’s criterion (1.1) for independent two machines \( M_k, M_{k+1} \): that is, for any two jobs \( i, j \) in \( J_r \), if it holds inequality

\[
\min [m_{k,i}, m_{k+1,i}] \leq \min [m_{k,j}, m_{k+1,j}],
\]

then job \( i \) must precede job \( j \) in order to minimize the makespan of the sequence of \((n-r)\) jobs in \( J_r \) on \( M_k, M_{k+1} \) alone.

Hence, this sequence \( i_{r+1}, i_{r+2}, \ldots, i_n \) must be processed on \( M_k, M_{k+1} \) along this order after the time \( T_k(J_r) \) on \( M_k \) and also \( T_{k+1}(J_r) \) on \( M_{k+1} \).
and let $T_{k+1}^* (J_r)$ be elapsed time of the processing of $i_{r+1}^k, i_{r+2}^k, \cdots i_n^k$ on $M_{k+1}$ after the time $T_{k+1}^* (J_r)$ [cf. Fig. 1]. As shown in Fig. 1, completion time $T_{k+1}$ of the sequence $i_{r+1}^k, i_{r+2}^k, \cdots i_n^k$ on $M_{k+1}$ is obviously a possible earliest completion time of any sequence of $(n-r)$ jobs in $J_r$ on $M_{k+1}$.

So that it must be obtained next revised lower bound $LB(J_r)$ (1.2) of the makespans of every sequences of $n$ jobs with definite presubsequence $J_r$:

$$LB(J_r) = \max \begin{cases} 
T_2(J_r) + T_2^*(J_r) + \min_{i \in J_r} \sum_{\substack{p=3 \atop p \neq 4, 2}}^m m_{p, i}, \\
T_3(J_r) + T_3^*(J_r) + \min_{i \in J_r} \sum_{\substack{p=4 \atop p \neq 4, 2}}^m m_{p, i}, \\
T_{m-1}(J_r) + T_{m-1}^*(J_r) + \min_{i \in J_r} m_{m, i}, \\
T_m(J_r) + T_m^*(J_r).
\end{cases}$$

(1.2)

Here, $LB(J_r)$ is an increasing function of $r$ for $J_1 \subset J_2 (r_1 < r_2)$ and $LB(J_{n-1})$ is equal to real total elapsed time (makespan) $TE(J_{n-1})$ of a sequence uniquely determined by presubsequence $J_{n-1}$.
Also, $T_k(J_r) (k=1 \sim m, r=1 \sim n-1)$ is calculated by next recurrent relation where $J_r \equiv J_{r-1} \, i_r$ ($i_r$ is the last job of $J_r$) and $T_0(J_r) \equiv 0, T_k(J_0) \equiv 0$

\[ T_k(J_r) = \max \left[ T_{k-1}(J_r), T_k(J_{r-1}) \right] + m_{k, i_r} \, (k=1 \sim m, r=1 \sim n-1). \]  

(1.3)

Especially for obtaining the value of each $T^k_{k+1}(\overline{J}_r) \equiv T^k_{k+1}(i_{r+1}^* \cdots i_{n-1}^*)$ ($k=1 \sim m-1, r=1 \sim n-1$), next recurrent relation (1.4) similar to (1.3) can be used:

\[ T^k_{k+1}(i_{r+1}^* \cdots i_{l+1}) = \max \left[ T^k_{k+1}(i_{r+1}^* \cdots i_{l+1}), T_k(J_r) \right] + \sum_{j=1}^{l} m_{k, i_{r+j}} + m_{k+1, i_{l+1}^*} \]  

(1.4)

(l=1 \sim n-r)

where $T^k_{k+1}(i_r^*) \equiv T_{k+1}(J_r)$ (l=1).

1.1.2. Comparison with lower bound in refs. [3] [4] and B. B. algorithm

Revised lower bound (1.2) is more exact than the lower bound already given [3] [4] as follows:

\[ \text{LB}(J_r) = \max \begin{cases} 
  T_1(J_r) + \sum_{i \in \overline{J}_r} m_{1, i} + \min \sum_{i \in \overline{J}_r} m_{p, i}, \\
  T_2(J_r) + \sum_{i \in \overline{J}_r} m_{2, i} + \min \sum_{i \in \overline{J}_r} m_{p, i}, \\
  T_3(J_r) + \sum_{i \in \overline{J}_r} m_{3, i} + \min \sum_{i \in \overline{J}_r} m_{p, i}, \\
  \vdots \\
  T_{m-1}(J_r) + \sum_{i \in \overline{J}_r} m_{m-1, i} + \min m_{m, i}, \\
  T_m(J_r) + \sum_{i \in \overline{J}_r} m_{m, i}.
\end{cases} \]  

(1.5)

That is to say, since $T^k_{k+1}(\overline{J}_r) \geq \sum_{i \in \overline{J}_r} m_{k+1, i}$, if it's neglected the first term in maximum bracket of (1.5), LB$(J_r)$ of (1.2) is larger than that of (1.5) and more efficient in the sense that $T^k_{k+1}(\overline{J}_r)$ characterizes a possible
minimum sum of the idle time of \( M_{k+1} \) by processing of any backsubsequence of unordered remained \((n-r)\) jobs in \( J_r \) on machine \( M_{k+1} \) \((k=1 \sim m-1)\) and the calculations of the value of \( T^*_{k+1}(J_r) \) which is the same as that of \( T_{r+1}(J_r) \) isn’t so complicated. In the next section it will be shown this facts by solving some examples using B.B. algorithm with lower bound (1.2) and (1.5) respectively, resulting that the number of nodes by (1.2) is smaller than that by (1.5).

As B.B. algorithm with revised lower bound (1.2) is the same as that with (1.5) [1]~[4], there shall be no language of it.

1.1.3. Numerical examples

In this section some numerical examples are solved by branch and bound algorithm with revised lower bound (1.2) and lower bound (1.5) respectively.

Then, efficiency of the revised lower bound becomes clear.

**Example 1.** \((m=3, n=6)\) [1]

Processing Time (hrs.)

<table>
<thead>
<tr>
<th>(i)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_1, i)</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>(m_2, i)</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>(m_3, i)</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

In order to decide a subsequence \( i_{r+1} \), \( i_{r+2} \), \ldots \( i_{n-k} \) of \((n-r)\) jobs in \( J_r \) on \( M_k, M_{k+1} \) for each node \((J_r)\), it's sufficient to decide an optimal sequence of \( n \) jobs on two machines \( M_k, M_{k+1} \) in advance.

In this example, for \( M_1, M_2 \) an optimal sequence is say 643152 and for \( M_2, M_3 \) it's 235641. Then, calculations of revised lower bound \( LB(J_r) \) of each node \((J_r)\) is made as shown in the following for three nodes. That is, for a node \((J_3)\equiv(3)\), since \( i_{8}^1 \cdot i_{7}^1 \cdot \ldots \cdot i_{1}^1 = 64152 \) and \( i_{9}^2 \cdot i_{8}^2 \cdot \ldots \cdot i_{6}^2 = 25641 \), each term in maximum bracket of \( LB(3) \) can be calculated as below:

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Completion time of a sequence $J_r i_{r+1} i_{r+2} \cdots i_n$ on $M_2$.

Order of jobs 3 6 4 1 5 2

$(M_1)$

$4 \quad 6 \quad 9 \quad 15 \quad 21 \quad 33 \quad T_2 + \min m_{i_1} i = 52 + 3 = 55$

$M_2$

10 24 35 42 50 52

Completion time of a sequence $J_r i_{r+1} i_{r+2} \cdots i_n$ on $M_3$.

Order of jobs 3 2 5 6 4 1

$(M_2)$

10 12 20 34 45 52 $\text{LB}(3) = \max \{55, 56\} = 56$

$M_3$

18 21 31 46 53 56

Next for a node $(J_r) = (35)$, since $i_3^1 i_4^1 i_5^1 i_6^1 = 6412$ and $i_3^0 i_4^2 i_5^2 i_6^2 = 2641$, $\text{LB}(35)$ can be calculated as below briefly:

3 5 6 4 1 2

$(M_1)$

4 10 12 15 21 33 $52 + 3 = 55$

$M_2$

10 18 32 43 50 52

3 5 2 6 4 1

$(M_2)$

10 18 20 34 45 52 $\text{LB}(35) = \max \{55, 56\} = 56$

$M_3$

18 28 31 46 53 56

Another example is shown for $\text{LB}(356)$ as below; here they hold $i_4^1 i_5^1 i_6^1 = 412$ and $i_4^3 i_5^2 i_6^2 = 241$

3 5 6 4 1 2

$(M_1)$

4 10 12 15 21 33

$M_2$

10 18 32 43 50 52 $52 + 3 = 55$

3 5 6 2 4 1

$(M_2)$

10 18 32 34 45 52

$M_3$

18 28 44 47 54 57 $\text{LB}(356) = \max \{55, 57\} = 57$

By similar calculations of each value of $\text{LB}(J_r)$, scheduling tree for example 1 becomes as in Fig. 2 where upper number at each node denotes a revised lower bound and lower number in parenthesis denotes a lower bound (1.5) already given [1].
Number of nodes by revised lower bound (1.2) becomes 20 which is smaller than 58 by lower bound [1.5] [1], and optimal sequences are 365241 and 356412, 356421 with 57 hrs. further obtained if node (3564) may be branched.

Another two examples have next processing time respectively:

**Example 2.** \((m=5, n=6)\) [3]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
i & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
m_{1,i} & 5 & 6 & 30 & 2 & 3 & 4 \\
m_{2,i} & 8 & 30 & 4 & 5 & 10 & 1 \\
m_{3,i} & 20 & 6 & 5 & 3 & 4 & 4 \\
m_{4,i} & 15 & 7 & 9 & 28 & 1 & 1 \\
m_{5,i} & 5 & 17 & 10 & 8 & 15 & 4 \\
\hline
\end{array}
\]

**Example 3.** \((m=3, n=6)\)

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
i & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
m_{1,i} & 5 & 3 & 12 & 2 & 9 & 11 \\
m_{2,i} & 9 & 8 & 10 & 6 & 3 & 1 \\
m_{3,i} & 6 & 2 & 4 & 12 & 7 & 3 \\
\hline
\end{array}
\]

Then, let \(N_{r} = \text{Number of nodes by revised lower bound (1.2)}\)

\(N_{o} = \text{Number of nodes by lower bound (1.5)}\)

\(N_{a} = \text{Number of nodes by lower bound (1.5) with additional two terms } h^{(1)}, h^{(2)} [3] \text{ for three machines case:}\)
Machine Scheduling Problem

\[ k^{(1)} = T_1(J_r) + \sum_{p=1}^{3} m_p, k_0 + \sum_{i} \min(m_1, i, m_3, i), \]

where \( \sum_{p=1}^{3} m_p, k_0 = \max \sum_{i} m_p, i \)

and \[ k^{(3)} = T_2(J_r) + \sum_{p=2}^{3} m_p, k_1 + \sum_{i} \min(m_2, i, m_3, i), \]

where \( \sum_{p=2}^{3} m_p, k_1 = \max \sum_{i} m_p, i. \)

Then, results are shown in Table 1.

Table 1: Number of Nodes in Scheduling Tree.

<table>
<thead>
<tr>
<th>Example</th>
<th>Nr</th>
<th>No</th>
<th>Na</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>58</td>
<td>/</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>74</td>
<td>/</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>37</td>
<td>37</td>
</tr>
</tbody>
</table>

1.1.4. Remarks

Some remarks concerning this section will be itemized as follows:

1. Generalization of the additional terms [3] \((m=3)\) to lower bound (1.5) is given as below for \( m \) machines case \((m \geq 3)\).

\[ k^{(q)} = T_q(J_r) + \sum_{p=q}^{m} m_p, k_{q-1} + \sum_{i} \min(m_q, i, m_m, i), \]

where

\[ \sum_{p=q}^{m} m_p, k_{q-1} = \max \sum_{i} m_p, i, \quad (q = 1 \sim m-1) \]

Here, only for \( \text{LB}(J_r) = \max \sum_{q=1}^{m-1} [k^{(q)}] \) it holds \( \text{LB}(J_{n-1}) \leq \text{TE}(J_{n-1}) \).

2. Another way to eliminate the number of nodes in scheduling tree is to use the next theorem which determines the definite order of two neighbouring jobs regardless of their position in sequence: that is,
Theorem. In $m$ machines case, for each neighbouring two jobs $i, j$, if next $\frac{1}{2}m(m-1)$ inequalities (1) and (2) hold;

$$\min [m_k, i, m_{k+1}, j] \leq \min [m_k, j, m_{k+1}, i] \quad (k=1 \sim m-1) \quad (1)$$

$$\min \left[ \sum_{k=p}^{q} m_k, i, \sum_{k=p+1}^{q+1} m_k, j \right] \leq \min \left[ \sum_{k=p}^{q} m_k, j, \sum_{k=p+1}^{q+1} m_k, i \right], \quad (2)$$

then job $i$ must always precede job $j$ regardless of their position.

Corollary. In three machines case, inequalities (1) and (2) in the theorem become next forms:

$$\min [m_1, i, m_2, j] \leq \min [m_1, j, m_2, i], \min [m_2, i, m_3, j] \leq \min [m_2, j, m_3, i] \quad (1)$$

$$\min [m_1, i+m_2, i, m_2, j] \leq \min [m_1, j+m_2, i, m_2, i+m_3, i]. \quad (2)$$

Determination of this definite order $ij$ is very simple because each inequality in (1) and (2) has transitive property. By using this theorem, if definite order $ij$ is determined, then any nodes that contain the order $ji$ or shall contain the order $ji$ afterwards can be omitted in branching a node. For example, scheduling tree of the example ($m=3, n=6$) in Ref. [3] (p. 184) which has 367 nodes by lower bound (1.5) and 65 nodes by (1.5) with additional two terms $h^{(1)}, h^{(2)}$, has 49 nodes by revised lower bound (1.2) applying this corollary and about 93 nodes only by revised lower bound.

1.2. Case where passing is allowed

For this case, former paper [4] has presented some branch and bound algorithms for optimal sequencing of $n$ jobs through $m(m \geq 4)$ machines where passing of job is allowed. There, formulation of the lower bound:

---

1) The proof of this theorem will be shown in paper to be published in future [6].
of each node hasn't taken into account the sum of idle time of each machine by processing of each job belonging to the set of unordered remained jobs. In the following it can be taken into consideration by using johnson’s criterion for two machines in order to make more exact that lower bound.

For the present case the order of \( n \) jobs may not be the same for each of \( m \) machines, but for obtaining optimal solution it can be assumed that the order of \( n \) jobs is the same for first two machines \( M_1, M_2 \) and for last two machines \( M_{m-1}, M_m \) respectively.

First, some terminologies must be defined as follows.

Let \( J_{r}^{12}, J_{r}^{k} (k=3\sim m-2), J_{r}^{m} \) be definite subsequence of \( r \) jobs among \( n \) jobs that are processed on machine \( M_1 \) and \( M_2, M_k (k=3\sim m-2), M_{m-1} \) and \( M_m \) respectively and let

\[
(J_r) = \begin{pmatrix}
J_{r}^{12} \\
J_{r}^{3} \\
J_{r}^{4} \\
\vdots \\
J_{r}^{m-2} \\
J_{r}^{m-1, m}
\end{pmatrix} \quad (r=1\sim n-1)
\]

(1.6)

denotes the set of all sequences of \( n \) jobs that have definite subsequence \( J_{r}^{12}, J_{r}^{k} (k=3\sim m-2), J_{r}^{m} \) as their first \( r \) jobs processed on \( M_1 \) and \( M_2, M_k (k=3\sim m-2), M_{m-1} \) and \( M_m \) respectively.

Next let \( T_k(J_{r}^{12}) (k=1, 2), T_k(J_{r}^{k}) (k=3\sim m-2), T_k(J_{r}^{m-1, m}) (k=m-1, m) \) be the earliest completion time of the sequence \( J_{r}^{12}, J_{r}^{k} (k=3\sim m-2), J_{r}^{m-1, m} \) on \( M_k (k=1, 2), M_k (k=3\sim m-2), M, (k=m-1, m) \) respectively if necessary by considering the following sequences of some unordered remained jobs of the former sequence \( J_{r}^{k} \) and let \( l_{r, k} \) be the last job of \( J_{r}^{k} (k=1\sim m) \) \((l_{r, 1}=l_{r, 2}=l_{r, 12}, l_{r, m-1}=l_{r, m}=l_{r, m-1, m}, \) the same for \( J_{r}^{k} \)) and \( J_{r}^{12}, J_{r}^{k} (k=3\sim m-2), J_{r}^{m-1, m} \) be the set of all unordered remained jobs after the processing of presubsequence \( J_{r}^{12}, J_{r}^{k} (k=3\sim m-2), J_{r}^{m-1, m} \) respectively.
1.2.1. Value of the earliest completion time $T_k(J_r^k)$

Each value of $T_k(J_r^{12})$ ($k=1, 2$), $T_k(J_r^k)$ ($k=3 \sim m-2$) and $T_k(J_r^{m-1,m})$ ($k=m-1, m$) is determined as follows:

1. $T_k(J_r^{12})$ ($k=1, 2$)

\[
T_1(J_r^{12}) = \sum_{i \in J_r^{12}} m_{i, r},
\]

\[
T_2(J_r^{12}) = \max \left[ T_1(J_r^{12}), T_2(J_r^{12} - l_{r, 12}) \right] + m_{2, r, 12},
\]

($r=1 \sim n-1$)

where $J_r^{12} - l_{r, 12}$ denotes the sequence obtained from $J_r^{12}$ by excluding its last job $l_{r, 12}$.

2. $T_k(J_r^k)$ ($k=3 \sim m-1$).

As each $J_r^k$ is defined independently for each other, there may be jobs of $J_r^k$ not belonging to some of the former $J_r^{12}, J_r^3, \ldots, J_r^{k-1}$.

Hence for example, let $i_3^{12}$ be a job in $J_r^3 \cap J_r^{12}$ having a smallest position number in $J_r^3$ and $i_3^{12}, i_4^{12}$ be a job in $J_r^4 \cap J_r^3 \cap J_r^{12}, J_r^4 \cap J_r^3 \cap J_r^{12}$ respectively both having a smallest position number in $J_r^4$, then let $i_4^3$ be a job in $J_r^4 \cap J_r^3$ having a smallest position number in $J_r^4$; that is, a job equal to either $i_3^{12}$ or $i_4^{12}$. Generally, let $i_{k,q-1}^{b-1,p-1}$ be a job in $J_r^k \cap J_r^q \cap J_r^{p-2}$ for all $q=b-1 \sim k-1$ and for each $p$ ($3 \leq p \leq k$) having a smallest position number in $J_r^k$ where for example $i_{k,4}^{3,3} = i_{k,4}^{13}$ ($k=4, p=3$) and $i_{k-1}^1$ be a job in $J_r^k \cap J_r^{k-1}$ having a smallest position number in $J_r^k$; that is, a job equal to either of $i_{k,p-1}^{b-1,p-1}$.

Then, for the determination of the value of $T_3(J_r^3)$ which is by definition the earliest completion time of $J_r^3$ on $M_3$, subsequence of jobs in $J_r^3 \cap J_r^{12}$ must be processed on $M_1$ and $M_2$ after $J_r^{12}$. As $i_3^{12}$ has a smallest position number in $J_r^3 \cup J_r^{12}$, by considering the idle time of $M_3$ caused by the processing of the sequence $\{i_3^{12} \cdots l_{r, 3}\}$, $T_3(J_r^3)$ must be replaced by $T_3(J_r^3)$ as in the next form: [cf. Fig. 3]
where \( \{i_9^{12} \cdots l_r, s\} \) denotes the subsequence of \( J_r^s \) which begins from \( i_9^{12} \) and ends at \( l_r, s \), but when \( i_9^{12} \) doesn’t exist it means \( \{i_9^{12} \cdots l_r, s\} \equiv l_r, s \) and \( T_5^*(J_r^s - \{i_9^{12} \cdots l_r, s\}) \) which denotes the completion time on \( M_9 \) of a subsequence \( J_r^s - \{i_9^{12} \cdots l_r, s\} \) contained in \( J_r^s \cap J_r^s \) can be calculated by using the relation (a) which is the same as equation (1.3) in sec. 1.1.1:

\[
T_k(i_j, k) = \max \left[ \frac{T_k-1(i_j, k)}{T_k(i_{j-1}, k)} \right] + m_k, i_j, k.
\]  

where \( T_k(i_j, k) \) denotes the completion time on \( M_k \) of \( j \)th job \( i_j, k \) of the sequence on \( M_k \).

\[
\varepsilon(J_3) = \begin{pmatrix} J_3^2 \\ J_3^3 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 2 \\ 3 & 5 & 4 \end{pmatrix}, \quad \ell_3, 12 = 2, \quad i_3^{12} = 5, \quad \ell_3, 3 = 4 \quad \{i_3^{12} \cdots l_3, 3\} = \{5, 4\}
\]

**Fig. 3.** Example of \( T_4(J_r^s) \) and \( T_5(J_r^s) \) where \( n=5, r=3 \).

Next, it holds next form for \( T_4(J_r^s) \) by similar reasons as above:
where they must be defined that if $i_8 = i_{12}$ then it holds $i_8 = i_{12} = i_{13}$ and if $i_8 = i_{12}$ then it holds $m_{k_3} = i_{12} = 0$ ($k = 1, 2$) [cf. Fig. 4], and $T_s^*(J^* - \{i_8 \cdots l_{r, 4}\}$ having the same meaning as in (1.8) can be calculated by using (a).

\[ T_s(J^*) \geq T_s(J^*) = \max \left( \begin{array}{c} \max \left[ T_1(J^*) + m_1, i_8^{12}, \right] + m_2, i_8^{13}, \right] + m_3, i_8^{13} \\ T_s(J^*) \end{array} \right) + \sum_{i \in \{i_8 \cdots l_{r, 4}\}} m_{k_3}, i \\ (r = 1 \sim n - 1) \]

where maximum operations $T_k(p)$ ($p = 2 \sim k - 1$) are defined as follows to simplify the term in maximum bracket of (1.10) and calculated for increasing $p$ to obtain $T_{k, k-1}$,
Machine Scheduling Problem

\[ T_{k, 2} \equiv \max \left[ T_s(J^*_r) + m_1, i_{k-1, k-1}^{r-1, r-1} \right] \quad (p=2) \]

\[ T_{k, p} \equiv \max \left[ T_s(J^*_r) + m_{p-1}, i_{k-1, k-1}^{r-1, r-1} \right] \quad (p=3 \sim k-1) \]

and they must be defined that if \( i_{k-1}^{r-1} \equiv \cdot \equiv i_{k-q+1}^{r-1} \equiv 0 \) (3 \( \leq p \leq q-1 \)) and \( i_{k-1}^{r-1} \equiv \cdot \equiv i_{k-p+1}^{r-1} \equiv 0 \) (q + 1 \( \leq p \leq k \)) and moreover \( T_k^*(J^*_r - \{i_{k-1}^{r-1} \cdots l_r, k\}) \) denotes the completion time on \( M_k \) of a sequence \( J^*_r - \{i_{k-1}^{r-1} \cdots l_r, k\} \) which can be calculated by using (a) from temporarily determined completion time on \( M_{k-1} \) of each job of this sequence for calculation of \( T_{k-1}(J_{r-1}^*) \) by (1.10) or by (1.8) for \( k=4 \).

3. \( T_m(J_r^{n-1, m}) \). Lastly it holds

\[ T_m(J_r^{n-1, m}) \geq T_m(J_r^{n-1, m}) = \max \left[ T_{m-1}(J_r^{n-1, m}), T_m^*(J_r^{n-1, m} - l_r, m-1, m) \right] + m_m, l_r, m-1, m, \]

(1.11)

where \( T_m^*(J_r^{n-1, m} - l_r, m-1, m) \) is calculated by using (a) as above.

1.2.2. Revised lower bound

Next, Johnson's criterion for two machines is applied to each two machines \( M_k, M_{k+1} \) (k = 1 \( \sim m-1 \)) with the set of \((n-r)\) unordered remained jobs \( J^*_r, J^{r+1}_r \) respectively.

Let they defined that

\[ J_r^{(k+1)} = J^*_r \cap J^{r+1}_r, \quad J_r^{(z)} = J^*_r \cap J^{r+1}_r, \quad J_r^{(a+1)} = J^*_r \cap J^{r+1}_r, \]

where \( J_r^{(n-1)} \equiv \phi, J_r^{(0)} \equiv \phi (k = m-1) \).

Then, all jobs in \( J_r^{(z+1)} \) must be optimally ordered by Johnson's criterion (1.1) for two machines \( M_k, M_{k+1} \) as in sec. 1.1.1.

After the time \( T_k(J_r^{z}) \) on \( M_k \), first all jobs in \( J_r^{(z)} \) are processed along the same ordering as in \( J_r^{z+1} \), and after the time \( T_{k+1}(J_r^{z+1}) \) on \( M_{k+1} \), first all jobs in \( J_r^{(a+1)} \) are processed along the same ordering as in \( J_r^{z+1} \), then after the processing of all jobs in \( J_r^{(z)} \) on \( M_k \) and all jobs in \( J_r^{(a+1)} \) on
A sequence of $J_r^{(k+1)}$ defined by Johnson's criterion is processed on $M_k$, $M_{k+1}$.

Then, let $T_{k+1}^{(k+1)}(J_r^{(k+1)})$ be the elapsed time of the processing of all jobs in $J_r^{(k+1)}$, $J_r^{(k+1)}$ after $T_{k+1}^{(k+1)}(J_r^{(k+1)})$ on $M_{k+1}$. [cf. Fig. 5]

Let as shown in Fig. 5, completion time of these jobs on $M_{k+1}$ be $T_{k+1}^{(k+1)}$ then $T_{k+1}^{(k+1)}$ is a possible earliest completion time of any sequence of all jobs in $J_r^{(k+1)}$ on $M_{k+1}$.

Hence, it holds next revised lower bound for each node $(J_r)$ ($r=1 \sim n-1$):

$$\begin{align*}
LB(J_r) &= \max \left( T_{m-2}(J_r^{m-2}) + T_{m-2}^{(k+1)}(J_r^{m-2}) + \sum_{k=1}^{m-2} \min_{i \in J_r^{m-1,m}} m_{k,i} + \sum_{k=m-1}^{m} m_{k,i}, \\
T_{m-1}(J_r^{m-1,m}) + T_{m-1}^{(k+1)}(J_r^{m-1,m}) + \min_{i \in J_r^{m-1,m}} m_{m,i}, \\
T_m(J_r^{m,m}) + T_m^{(k+1)}(J_r^{m,m}) \right)
\end{align*}$$

(1.12)

$LB(J_r)$ is an increasing function of $r$ for each $J_{r_1} \subset J_{r_2}$ ($r_1 < r_2$) and
Machine Scheduling Problem

LB\(J_{n-1}\) is equal to real total elapsed time \(TE(J_{n-1})\) of a set of sequences of \(n\) jobs on each machines that is uniquely determined by \(J_{n-1}\).

1.2.3. Comparison with lower bound in Ref. [4] and B. B. algorithms

Revised lower bound (1.12) is more exact than the lower bound already given [4] as follows:

\[
\begin{align*}
\text{LB}(J_r) &= \max \left( 
T_1(J_r^{12}) + \sum_{i \in J_r^{12}} m_1, i + \sum_{k=2}^{m-2} \min_{i \in J_r^{k-1,m}} \sum_{k=m-1}^{m} m_k, i, \\
T_2(J_r^{12}) + \sum_{i \in J_r^{12}} m_2, i + \sum_{k=3}^{m-2} \min_{i \in J_r^{k-1,m}} \sum_{k=m-1}^{m} m_k, i, \\
T_3(J_r^3) + \sum_{i \in J_r^3} m_3, i + \sum_{k=4}^{m-2} \min_{i \in J_r^{k-1,m}} \sum_{k=m-1}^{m} m_k, i, \\
T_m(J_r^{m-1,m}) + \sum_{i \in J_r^{m-1,m}} m_{m-1}, i + \min_{i \in J_r^{m-1,m}} \sum_{k=m-1}^{m} m_m, i, \\
T_m(J_r^{m-1,m}) + \sum_{i \in J_r^{m-1,m}} m_{m}, i,
\right) \\
(r=1\sim n-1)
\end{align*}
\]

That is to say, since \(T_{k+1}^k(J_r^{k+1}) \geq \sum_{i \in J_r^{k+1}} m_{k+1}, i\), if it's neglected the first term in maximum bracket of (1.13), \(\text{LB}(J_r)\) of (1.12) is larger than that of (1.13) and more efficient in the sense that \(T_{k+1}^k(J_r^{k+1})\) characterize a possible minimum sum of the idle time of \(M_{k+1}\) caused by processing of any backsequence of unordered remained \((n-r)\) jobs in \(J_r^{k+1}\) on machine \(M_{k+1}(k=1\sim n-1)\) and the calculations of the value of \(T_{k+1}^k(J_r^{k+1})\) which is the same as that of \(T_{k+1}^k(J_r^{k+1})\) say isn't so complicated.

In the next section, it will be shown this facts by solving an example using B. B. algorithm with lower bound (1.12) and (1.13) respectively.

Branch and bound algorithms.

Five B. B. algorithms has presented in former paper [4]. Principal algorithms are as follows:

\[\text{Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.}\]
Algorithm 1. The procedure is to start from a node \((J_0)\) representing all possible sequences of the amounts \((n!)^{n-2}\), then to divide this node into \(n^{n-2}\) subclasses (nodes \((J_i)\)) according to whether the first job of \(J_i^0\) and \(J_i^k (k=3-m-2)\) and \(J_i^{n-1,m}\) is 1, 2, ..., \(n\). And for each of them, by using (1.7)~(1.12), (a), (b), \(LB(J_i)\) is calculated and one of the nodes \((J_1)\) having minimum \(LB(J_i)\) is divided into \((n-1)^{n-2}\) subclasses (nodes \((J_2)\)) having the same first job as this node \((J_1)\) on each machine, according to whether the second job of \(J_i^1\) and \(J_i^k (k=3-m-2)\) and \(J_i^{n-1,m}\) is 1, 2, ..., \(n\) except the same number as this branched node \((J_1)\). And then \(LB(J_2)\) is calculated for each of them as above and one of the nodes \((J_2)\) having minimum \(LB(J_i)\) is divided into \((n-2)^{n-2}\) subclasses (nodes \((J_3)\)) having the same first two jobs as this node \((J_2)\) on each machine, according to whether the third job of \(J_i^2\) and \(J_i^k (k=3-m-2)\) and \(J_i^{n-1,m}\) is 1, 2, ..., \(n\) except the same numbers as this branched node \((J_2)\).

Proceeding by the same way to the nodes \((J_{n-1})\) of the amounts \(2^{n-2},LB(J_{n-1})\) of each of them is calculated as above. Let \(MLB(J_{n-1})\) denotes the minimum of these \(LB(J_{n-1})\), then already formed nodes \((J_r)\) \((r=1~n-1)\) of order one for which it holds inequality \(LB(J_r) \geq MLB(J_{n-1})\) are discarded.

In this situation if there are no nodes \((J_r) (r=1~n-1)\) such that \(LB(J_r) < MLB(J_{n-1})\), then a set of sequences of \(n\) jobs on each machine that is uniquely determined by \(J_{n-1}\) which gives \(MLB(J_r)\) is an optimal solution, being \(LB(J_{n-1})=TE(J_{n-1})\). Otherwise, the same procedure as above is applied to the remaining nodes of order one by branching a node having largest \(r\) among the minimum of their \(LB(J_r)\).

By proceeding by this way, finally it can be found a node \((J_{n-1})\) having the minimum of \(LB(J_r)\) among the remaining nodes \((J_r) (r=1~n-1)\) of order one and a set of sequences uniquely determined by this node is an optimal solution.

Algorithm 2. (algorithm 3. in Ref. [4])

The procedure that is different from algorithm 1 is to replace the
$T_k^*(J_r^k) = \{i_k^{-1} \cdots l_r, k\}$ by the term $T_k(J_r^k - \{i_k^{-1} \cdots l_r, k\})$ in revised lower bound $LB(J_r) (k=3 \sim m, \tau = 1 \sim n - 1)$ where each value of $T_k(J_r^k - \{i_k^{-1} \cdots l_r, k\})$ is known from the result of the calculations of the lower bound of a former node connected with this node $(J_r)$, having $J_r^k = \{i_k^{-1} \cdots l_r, k\}$ as presubsequence on $M_k$, and to calculate $TE(J_{n-1})$ by using (a) for a node $(J_{n-1})$ having minimum $LB(J_{n-1})$ obtained by the same way as in algorithm 1 and to discard all nodes $(J_r) (r = 1 \sim n - 1)$ of order one having $LB(J_r) \geq TE(J_{n-1})$ in the case when it holds $TE(J_{n-1}) \leq LB(J_{n-1})$ for each node of the remained nodes $(J_{n-1})$ on the same branch as a node having minimum $LB(J_{n-1})$, or otherwise to calculate $TE(J_{n-1})$ of some nodes $(J_{n-1})$ having smaller $LB(J_{n-1})$ than the firstly calculated $TE(J_{n-1})$ and to determine the least $TE(J_{n-1})$ among them in order to compare with $LB(J_r)$ for each of all remained nodes $(J_r) (r = 1 \sim n - 1)$ of order one, and to follow the same steps until a node $(J_{n-1})$ determining an optimal solution is found as in algorithm 1.

Practically it may be more efficient to use algorithm 2 than the other if balance between the quantity of calculations of the value of lower bound $LB(J_r)$ for each node $(J_r)$ and the number of nodes of scheduling tree is taken into consideration.

### 1.2.4. Numerical example

In this section, algorithm 2 with revised lower bound (1.12) and lower bound (1.13) already given in Ref. [4] are applied to four machines case respectively. For this case, earliest completion time $T_k(J_r^{12}) (k=1, 2)$, $T_k(J_r^{34}) (k=3, 4)$ of a subsequence $J_r^{12}, J_r^{34}$ on machines $M_1$ and $M_2, M_3$ and $M_4$ respectively become next forms:

\[
\begin{align*}
T_k(J_r^{12}) &= \sum_{i_r \in r} m_{1r}, \\
T_k(J_r^{34}) &= \max \left[ T_k(J_r^{12}), \right. \\
&\left. + m_{2r} - l_{12} \right]
\end{align*}
\]

(1.14)
\[ T_b(J_r^{s4}) \geq T_b(J_r^{s4}) = \max \left[ \frac{T_b(J_r^{12})}{T_b(J_r^{s4} - \{i_{s4}^*\})} + \frac{m_{s}, i_{s4}^*}{m_{s}, i_{s4}} \right] \]

\[ T_d(J_r^{s4}) \geq T_d(J_r^{s4}) = \max \left[ \frac{T_d(J_r^{s4})}{T_d(J_r^{s4} - l_{s4})} + m_{s4}, i_{r, s4} \right] \] (1.15)

And revised lower bound \( LB(J_r) \) of a node \( (J_r) \) of a tree is as follows:

\[
LB(J_r) = \begin{cases} 
T_b(J_r^{12}) + T_b(J_r^{12}) + \min_{i_{s4}^*, i_{s4}^*} \sum_{r=3}^{n} m_{p}, i_{s4} \\
(r=1 \sim n-1) 
\end{cases}
\]

This \( LB(J_r) \) is an increasing function of \( r \) for \( J_r \subseteq J_{r2} \) \((r_1 < r_2)\) and it holds \( LB(J_{n-1}) \leq TE(J_{n-1}) \).

Following the algorithm 2, next example can be easily solved for each of two lower bounds.

**Example** \( (m=4, n=4) \). [4].

<table>
<thead>
<tr>
<th>Processing Time (hrs.)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_{b}, i )</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>( m_{b}, i )</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>( m_{b}, i )</td>
<td>4</td>
<td>4</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>( m_{b}, i )</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Scheduling tree by the procedure of the algorithm 2 with revised lower bound becomes the next form [Fig. 6] having 42 nodes smaller than 46 nodes by lower bound already given [4]. Upper number and lower number in parenthesis labeled at each node denotes revised lower bound and old lower bound respectively.

In this example each \( LB(J_r) \) \((n-1=3)\) has just coincided with \( TE(J_n) \) calculated by using (a).

Examples of the calculations of \( LB(J_r) \) are shown below for nodes \( LB(3), LB(4) \).
**m Machine Scheduling Problem**

<table>
<thead>
<tr>
<th>LB(i)</th>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M₆</td>
<td>7</td>
<td>14</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>M₇</td>
<td>28</td>
<td>35</td>
<td>39</td>
<td>39+1=40</td>
</tr>
<tr>
<td></td>
<td>M₈</td>
<td>28</td>
<td>31</td>
<td>35</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>M₉</td>
<td>32</td>
<td>40</td>
<td>47</td>
<td>48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job (3)</th>
<th>Ms</th>
<th>7</th>
<th>14</th>
<th>15</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₈</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>39</td>
<td>39+1=40</td>
</tr>
<tr>
<td>M₉</td>
<td>28</td>
<td>31</td>
<td>35</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>M₄</td>
<td>32</td>
<td>40</td>
<td>47</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LB(Jt)</th>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M₅</td>
<td>7</td>
<td>14</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>M₆</td>
<td>8</td>
<td>13</td>
<td>19</td>
<td>20+5=25</td>
</tr>
<tr>
<td></td>
<td>M₇</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M₈</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>M₉</td>
<td>11</td>
<td>18</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>M₄</td>
<td>18</td>
<td>26</td>
<td>36</td>
<td>37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job (4)</th>
<th>Ms</th>
<th>7</th>
<th>14</th>
<th>15</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₅</td>
<td>4</td>
<td>8</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M₆</td>
<td>7</td>
<td>14</td>
<td>15</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>M₇</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>M₈</td>
<td>11</td>
<td>18</td>
<td>32</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>M₉</td>
<td>18</td>
<td>26</td>
<td>36</td>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 6. Scheduling Tree of the Example.**

A node \( (J₃) = (1,2) \) having \( \text{LB}(J₃) = \text{TE}(J₃) = 37 \) gives an optimal solution.
with a sequence (1342) on $M_1$ and $M_2$ and a sequence (1432) on $M_3$ and $M_4$ and minimal total elapsed time is 37 hrs.

§ 2. Other types of lower bound in case where no passing is allowed

In this section, only the case where no passing is allowed will be considered. And, for the purpose of estimating a better lower bound of makespans, two types of lower bound under definite backsubsequence and under definite pre—and back-subsequences respectively are constructed.

2.1. Sequences with definite backsubsequence

Let $\text{LB}(R_r)$ be the lower bound of makespans of sequences with definite backsubsequence $R_r$ of $r$ jobs ($r=1 \sim n-1$) and $\overline{R_r}$ be the set of unordered remained $(n-r)$ jobs.

As in sec. 1, let $i_1^ki_2^k \cdots i_{n-r}^k$ be a sequence of all jobs in $\overline{R_r}$, which is determined by Johnson's criterion (1.1) for two machines $M_k, M_{k+1}$ ($k=1 \sim m-1$), then after the time $T^0_k \equiv \min_{i \in \overline{R_r}} \sum_{p=1}^{k-1} m_{p,i}$ ($T^0_k \equiv 0$), on each $M_k$ ($k=1 \sim m-1$), the above sequence $i_1^k i_2^k \cdots i_{n-r}^k$ is processed on $M_k$ and $M_{k+1}$ such that idle time of $M_k$ doesn't exist.

Next, let $T^*_k+1(\overline{R_r})$ be the elapsed time of the processing of this sequence on $M_{k+1}$ after the time $T^0_k$. [cf. Fig. 7]

Let $E^*_k+1$ be the completion time on $M_{k+1}$ of a last job $i_{n-r}^k$ of this sequence as shown in Fig. 7. Next, after the time $T^*_k-r \equiv \min_{i \in \overline{R_r}} \sum_{p=1}^{k-1} m_{p,i} + \sum_{q=1}^{n-r} m_{k,q}$ on $M_k$, definite backsubsequence $R_r$ must be processed on $M_k, M_{k+1}, \cdots, M_m$; that is, each job of $R_r$ must be processed on $M_k$ continuously without idle time of $M_k$ and then each job of $R_r$ must be processed on the following machines $M_{k+1}, M_{k+2}, \cdots, M_m$ by using (a) such that the first job of $R_r$ must be processed on $M_{k+1}$ after the time $E^*_k+1$, on each $M_q$ ($q=k+2 \sim m$) after the time $E^*_k+1 + \sum_{\substack{p=k+2 \in \overline{R_r}}} \min_{i} m_{p,i}$.
Machine Scheduling Problem

Let $E_m^k(R_r)$ be the elapsed time on $M_m$ from the time $F_{M,m}^{k-1} = E_m^{k-1} + \sum_{\rho=k+2}^{m} \min_{u \in R_r} m_{p, u}$ to the completion time $G_m^k(R_r)$ of the last job of $R_r$.

Then, completion time on $M_m$ of the last job of any sequence of $n$ jobs with definite backsubsequence $R_r$ isn’t earlier than the time $G_m^k(R_r)$.

Hence, next lower bound $LB(R_r)$ of the makespans of sequences with definite backsubsequence $R_r$ is obtained:

$$
LB(R_r) = \max \left\{ \begin{array}{ll}
T_2^1(\bar{R}_r) + \sum_{\rho=3}^{m} \min_{u \in \bar{R}_r} m_{p, u} + E_m^1(R_r), \\
\min_{u \in \bar{R}_r} m_{1, u} + T_3^2(\bar{R}_r) + \sum_{\rho=4}^{m} \min_{u \in \bar{R}_r} m_{p, u} + E_m^2(R_r), \\
\min_{u \in \bar{R}_r} \sum_{\rho=1}^{m-5} m_{p, u} + T_5^4(\bar{R}_r) + \min_{u \in \bar{R}_r} m_{m, u} + E_m^4(R_r), \\
\min_{u \in \bar{R}_r} \sum_{\rho=1}^{m-2} m_{p, u} + T_4^5(\bar{R}_r) + E_m^5(R_r), \\
\end{array} \right. 
$$

$$
LB(R_{n-1}) = T_2^1(\bar{i}_1) + \sum_{\rho=3}^{m} \min_{u \in \bar{R}_{n-1}} m_{p, u} + E_m^1(R_{n-1}) \quad [\text{First term in max. bracket}] 
\equiv T_2^1(\bar{i}_1) + \sum_{\rho=3}^{m} m_{p, \bar{i}_1} + E_m^1(R_{n-1}),
$$

where $\bar{R}_{n-1} \equiv \bar{i}_1$.

(2.1)

Another lower bound can be obtained by using only the first term in maximum bracket of $LB(R_r)$ ($r=1 \sim n-2$); that is,

$$
LB(R_r) = T_2^1(\bar{r}_R) + \sum_{\rho=3}^{m} \min_{u \in \bar{R}_r} m_{p, u} + E_m^1(R_r) \quad (r=1 \sim n-1)
$$

(2.2)
Always it holds $(2.2) \leq (2.1)$. Each of the above two lower bounds $(2.1), (2.2)$ is an increasing function of $r$ where $R_{r_1} \subset R_{r_2} (t_1 \subset t_2)$ and $LB(R_{n-1})$ is equal to the real total elapsed time $TE(R_{n-1})$ of a sequence which is uniquely determined by $R_{n-1}$.

Moreover, another less exact lower bound in comparison with lower bound $(2.1)$ can be obtained as below by similar constructions as that of lower bound $LB(J_r)$ (1.5) in sec. 1.1.2.

As in the case of lower bound $(2.1)$, after the time $T_{i_1}^{r-r} \equiv \min_{i \in R_r} \sum_{p=1}^{k-1} m_{p,i}$ on each $M_k (k=1 \sim m)$, definite backsubsequence $R_r$ must be processed on $M_k, M_{k+1}, \cdots, M_m$ by the same ways as to lower bound $(2.1)$ [cf. Fig. 8], such that the first job of $R_r$ must be processed on $M_k$ after the time $T_{i_1}^{r-r}$ continuously and on $M_q (q=k+1 \sim m)$ after the time $T_{i_1}^{r-r} + \sum_{p=k+1}^{q} \min_{i \in R_r} m_{p,i}$. Let the time $F_{i_1}^{r-r}$ be defined as below: $F_{i_1}^{r-r} = \sum_{i \in R_r} \min_{i \in R_r} m_{p,i}$. Let the time $F_{i_1}^{r-r}$ be defined as below: $F_{i_1}^{r-r} = \sum_{i \in R_r} \min_{i \in R_r} m_{p,i}$.
Machine Scheduling Problem

\( T_{e-r}^{k} + \sum_{p=k+1}^{m} \min m_{p,i}, \text{ then let } E_{m}^{k}(R_{r}) \) be the elapsed time on \( M_{m} \) from the time \( E_{e-m}^{k} \) to the completion time \( G_{m}^{k}(R_{r}) \) of the last job of \( R_{r} \) where it's holds that the time \( G_{m}^{k}(R_{r}) \) isn't larger than the time \( G_{m}^{k}(R_{r}) \) in Fig. 7.

Hence next simple lower bound can be obtained:

\[
\begin{align*}
\text{LB}(R_{r}) &= \max \\
&= \begin{cases} \\
\sum_{i \in R_{r}} m_{1,i} + \sum_{p=2}^{m} \min m_{p,i} + E_{m}^{1}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=2}^{m} \min m_{p,i} + E_{m}^{2}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=3}^{m} \min m_{p,i} + E_{m}^{3}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=4}^{m} \min m_{p,i} + E_{m}^{4}(R_{r}), \\
\end{cases} \\
&= \begin{cases} \\
\min_{i \in R_{r}} \sum_{p=1}^{m-2} \min m_{p,i} + E_{m}^{m-1}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=1}^{m-1} \min m_{p,i} + E_{m}^{m-2}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=1}^{m-1} \min m_{p,i} + E_{m}^{m-3}(R_{r}), \\
\min_{i \in R_{r}} \sum_{p=1}^{m-1} \min m_{p,i} + E_{m}^{m-4}(R_{r}), \\
\end{cases} \\
\end{align*}
\]

\((r = 1 \sim n-1)\)
Also next more simple lower bound which is the first term in maximum bracket of the lower bound (2.3) can be obtained:

\[ \text{LB}(R_r) = \sum_{i \in R_r} m_{i,1} + \sum_{p=2}^{m} \min_{i \in R_r} m_{p,i} + E_{m,1}(R_r). \quad (r=1 \sim n-1) \]

(2.4)

Each of LB(R_r) (2.3), (2.4) is an increasing function of r where \( R_1 \subset R_2 (r_1 < r_2) \) and LB(R_{n-1}) is equal to the real total elapsed time TE(R_{n-1}) of a sequence of n jobs uniquely determined by R_{n-1}.

**Remark:** The other lower bound LB(R_r) can be obtained if the first term in maximum bracket of the lower bound (2.1) or (2.3) is contained in its maximum bracket.

2.1.2. **B. B. algorithm**

B. B. algorithm can be constructed along similar procedures as the algorithm in sec. 1.1.2 by determining one job upward from last job of the backsequence at each step.

2.1.3. **Numerical example**

Next example \((m=3, n=6)\) will be solved by B. B. algorithm with lower bound LB(R_k) (2.1).

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Time (hrs.)</td>
</tr>
<tr>
<td>( i )</td>
</tr>
<tr>
<td>( m_{1,i} )</td>
</tr>
<tr>
<td>( m_{2,i} )</td>
</tr>
<tr>
<td>( m_{3,i} )</td>
</tr>
</tbody>
</table>

In three machines case, lower bound LB(R_r) (2.1) becomes next forms:

*Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.*
**m Machine Scheduling Problem**

\[
LB(R_r) = \max_{(r=1 \sim 4)} \left[ T_2^*(R_r) + \min_{i \in R_r} m_{3,i} + E_3^*(R_r), \right]
\]

\[
LB(R_5) = T_2^*(i_1) + m_{3,i_1} + E_3^*(R_5)
\]

\[
= \sum_{p=1}^{3} m_{p,i_1} + E_3^*(R_5). \quad (R_5 = i_1)
\]

In this example, a sequence determined by Johnson's criterion (1.1) is 146532 for \(M_1\) and \(M_2\) and 215346 for \(M_2\) and \(M_3\). Then, examples of the calculations of the value of \(LB(R_r)\) are as follows:

First, for \(LB(3)\), \(i_1 = 1\), \(i_2 = 4\), \(i_3 = 6\), \(i_4 = 5\), \(i_5 = 2\) = 14652 and \(i_1 = 3\), \(i_2 = 5\), \(i_3 = 4\), \(i_4 = 6\), \(i_5 = 1\) = 352647.

Next, for \(LB(53)\), \(i_1 = 1\), \(i_2 = 4\), \(i_3 = 6\), \(i_4 = 5\), \(i_5 = 2\) = 1462 and \(i_1 = 3\), \(i_2 = 5\), \(i_3 = 4\), \(i_4 = 6\), \(i_5 = 1\) = 352647.

By the same ways, scheduling tree can be constructed as in Fig. 9 where number rabeled at each node denotes the lower bound (2.1) of each node.

Optimal sequence is 142653 and 214653, 124653 (57 hrs.) further obtained if a node (4653) may be branched. Total number of the nodes of this tree is a possible least number 20.

**Remarks**

1. If lower bound (2.3) is used, then inspite of its simpler calculations of the value of \(LB(R_r)\), number of nodes in tree will much more increase.
2. The above optimal sequences have definite backsubsequence 653 in common, hence, for this example lower bound \(LB(R_r)\) is more efficient.
than lower bound $LB(J_r)$ in sec. 1.1.1. But lower bound $LB(J_r)$ is more efficient than lower bound $LB(R_r)$ for example 1 in sec. 1.1.3 on the other hand.

2.2. Sequence with definite pre-and back-subsequences

As mentioned in the remark (2) in former section, either one among the two lower bound $LB(J_r)$, $LB(R_r)$ is more efficient according to the type of the example. So that, construction of the lower bound $LB(J_r, R_S)$ of the makespans of sequences with definite presubsequence $J_r$ and definite backsubsequence $R_r$ may have some meaning.

2.2.1. Lower bound $LB(J_r, R_S)$

The above lower bound $LB(J_r, R_S)(r+S=1 \sim k-1)$ is constructed as follows:

Let $S_{n-r-S}$ be the set of unordered remained $(n-r-S)$ jobs. First for the case where $rS=0$, let $T_i(J_r)$ be the completion time of presubsequence $J_r$ on $M_k (k=1 \sim m)$ and a sequence $i_t^k i_2^k \cdots i_{n-r-s}^k$ be a sequence
of all jobs in $S_{n-r-s}$ which is determined by Johnson's criterion (1.1) for two machines $M_k, M_{k+1} (k=1 \sim m-1)$.

Then, after the time $T_{k}(J_r)$ on each $M_k (k=1 \sim m-1)$, the above sequence $i_{1}^{k} \cdot i_{2}^{k} \cdots i_{n-r-s}^{k}$ is processed on $M_k, M_{k+1}$ such that idle time of $M_k$ doesn't exist and let $T_{k+1}(S_{n-r-s})$ be the elapsed time of the processing of this sequence on $M_{k+1}$ after the time $T_{k+1}(J_r)$ and the completion time of this sequence on $M_{k+1}$ be $E_{k+1}^{n-r-s}$ [cf. Fig. 10].

Next, after the time $T_{k+1}^{n-r-s} = T_{k}(J_r) + \sum_{p=1}^{n-r-s} m_{k,i_p}$ on $M_k$, definite back-subsequence $R_s$ must be processed on $M_k, M_{k+1}, \ldots, M_m$: that is, each job of $R_s$ must be processed on $M_k$ continuously without idle time of $M_k$ and then each job of $R_s$ must be processed on the following machines $M_{k+1}, M_{k+2}, \ldots, M_m$ successively such that the first job of $R_s$ is processed on $M_{k+1}$ after the time $E_{k+1}^{n-r-s}$ and on $M_{k+2}$ after the time $F_{k+2}^{n-r-s}$ and on each $M_q (q=k+3 \sim m)$ after the time $F_{k,q}^{n-r-s}$ be the elapsed time from the time $E_{k,q}^{n-r-s}$ on $M_{k+1}$ to the time $F_{k,q}^{n-r-s}$ on $M_m$. 

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
and $E_n^k(R_s)$ be the elapsed time on $M_m$ after the time $F_{k,m}^{n-r}$ to the completion time $G_n^k(R_s)$ of the last job of $R_s$.

So, completion time on $M_m$ of the last job of any sequence of $n$ jobs with definite presubsequence $J_r$ and definite backsubsequence $R_s$ isn’t earlier than the time $G_m^k(R_s)$.

If $r=0$, then it’s defined that $LB(J_0, R_s) = LB(R_s)$: lower bound in sec. 2.1.1.

And if $s=0$, then it’s defined that $LB(J_r, R_0) = LB(J_r)$: revised lower bound in sec. 1.1.1.

Hence, next lower bound $LB(J_r, R_s)$ can be obtained:

$$
LB(J_r, R_s) = \max \begin{cases} 
T_g(J_r) + T_g^{n-r}(S_{n-r-1}) + E_m^2(S_{n-r-1}) + E_m^1(R_s), \\
T_g(J_r) + T_g^o(S_{n-r-1}) + E_m^2(S_{n-r-1}) + E_m^1(R_s), \\
T_m(J_r) + T_m^{n-1}(S_{n-r-1}) + E_m^{n-1}(R_s), \\
T_m(J_r) + T_m^{n-1}(S_{n-r-1}) + E_m^{n-1}(R_s), \\
\end{cases}
$$

$$(r+s = 1 \sim n-2, \; rs \neq 0)$$

Another lower bound $LB(J_r, R_s)$ can be obtained by using only the first term in maximum bracket of (2.5):

$$
LB(J_r, R_s) = T_g(J_r) + T_g^{i_1}(i_1) + \sum_{p=3}^{m} m_{p, i_1} + E_m^1(R_1): \text{first term in max. bracket,} \; (r+s = n-1, \; rs \neq 0) \; \text{where} \; S_1 \equiv i_1.
$$

$LB(J_r, R_0) = LB(J_r)$ (1.2) $(s=0)$

$LB(J_0, R_s) = LB(R_s)$ (2.1) $(r=0)$

(2.5)

Another lower bound $LB(J_r, R_s)$ can be obtained by using only the first term in maximum bracket of (2.6):

$$
LB(J_r, R_s) = T_g(J_r) + T_g^{n-r}(S_{n-r-1}) + E_m^2(S_{n-r-1}) + E_m^1(R_s),
$$

$$(r+s = 1 \sim n-1, \; rs \neq 0)$$

$LB(J_r, R_0) = LB(J_r)$ (1.2), $LB(J_0, R_s) = LB(R_s)$ (2.1).

(2.6)
Moreover, another less exact lower bound \( LB(J_r, R_r) \) in comparison with lower bound (2.5) can be obtained as below by similar constructions as that of lower bound \( LB(J_r) \) (1.5) and \( LB(R_r) \) (2.3): [cf. Fig. 11]

As in the case of lower bound (2.5), after the time \( T_k^{n-1} = T_k(J_r) + \sum_{i \in S_{n-r-k}} m_{k,i} \) on each \( M_k (k = 1 \sim m) \), definite backsubsequence \( R_r \) must be processed on \( M_k, M_{k+1}, \cdots, M_m \), that is, each job of \( R_r \) must be processed on \( M_k \) continuously without idle time of \( M_k \) and then each job of \( R_r \) must be processed on the following machines \( M_{k+1}, M_{k+2}, \cdots, M_m \) successively such that the first job of \( R_r \) is processed on \( M_{k+1} \) after the time \( F_{k,k+1}^{n-1} = \max[T_k^{n-1}, T_{k+1}(J_r)] + \min_{i \in S_{n-r-k}} m_{k+1,i} \) and on each \( M_q (q = k+2 \sim m) \) after the time \( F_{k,q}^{n-1} = \max[F_{k,q-1}^{n-1}, T_q(J_r)] + \min_{i \in S_{n-r-q}} m_{q,i} \).

Then, let \( E_m^k(S_{n-r-k}) \) be the elapsed time after the time \( T_k^{n-1} \) on \( M_k \) to the time \( F_{k,m}^{n-1} \) on \( M_m \) and \( E_m^k(R_r) \) be the elapsed time on \( M_m \) after the time \( F_{k,m}^{n-1} \) to the completion time \( G_m^k(R_r) \) of the last job of \( R_r \).

Hence, next lower bound can be obtained:
Another lower bound can be obtained by using only the first term in maximum bracket of (2.7): that is,

\[
\text{LB}(J_r, R_s) = T_1(J_r) + \sum_{i \in S_{n-r-s}} m_1, i + E_{m,i}(S_{n-r-s}) + E_{m,i}(R_s),
\]

\[
( r+s=1\sim n-2, \quad \text{rs} \neq 0 )
\]

\[
\text{LB}(J_r, R_s) = T_1(J_r) + m_1, i_1 + E_{m,1}(i_1) + E_{m,1}(R_s). \quad ( r+s=n-1, \quad \text{rs} \neq 0, \quad S_1 = i_1)
\]

\[
\text{LB}(J_r, R_0) = \text{LB}(J_r) \quad (s=0), \quad \text{LB}(J_0, R_s) = \text{LB}(R_s) \quad (r=0).
\]

(2.7)

Each of the above lower bounds \( \text{LB}(J_r, R_s) \) (2.5)-(2.8) is an increasing function of \( r \) and \( s \) where \( J_{r_1} \subset J_{r_2} \quad (r_1 < r_2) \) and \( R_{s_1} \subset R_{s_2} \quad (s_1 < s_2) \) and \( \text{LB}(J_r, R_s) \quad (r+s=n-1) \) is equal to the real total elapsed time \( \text{TE}(J_r, R_s) \) of a sequence uniquely determined by this subsequence \( (J_r, R_s) \quad (r+s=n-1) \).

2.2.2. B.B. algorithm

The procedure is almost the same as that of the B.B. algorithms in the above sections, except that: [I]. nodes \( (J_1), (J_1, R_1), (J_2, R_1), (J_3, R_2), \ldots, (J_r, R_i) \quad (r+s=n-1) \) or [II]. nodes \( (R_1), (J_1, R_1), (J_1, R_2), (J_2, R_3), \ldots, (J_r, R_i) \quad (r+s=n-1) \) are constructed successively.
2.2.3. Numerical examples compared with other lower bounds

Example 1 \((m=3, n=6)\) [Example 1 in sec. 1.1.3.].

<table>
<thead>
<tr>
<th>Processing Time (hrs.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
</tr>
<tr>
<td>(m_{1,i})</td>
</tr>
<tr>
<td>(m_{2,i})</td>
</tr>
<tr>
<td>(m_{3,i})</td>
</tr>
</tbody>
</table>

Scheduling tree of example 1 is shown in Fig. 12.

![Scheduling Tree of Example 1](image_url)

Fig. 12. Scheduling Tree of Example 1.

Number of nodes is a possible least number 20 which is the same as the result in example 1 for revised lower bound \(LB(J_*)\) and smaller than result by \(LB(J^c)\) (1.5) in sec. 1.1.3 and shall be smaller than the result by \(LB(R_*)\) in sec. 2.1.1.

Example 2 \((m=3, n=6)\) [Example 3 in sec. 1.1.3]

<table>
<thead>
<tr>
<th>Processing Time (hrs.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
</tr>
<tr>
<td>(m_{1,i})</td>
</tr>
<tr>
<td>(m_{2,i})</td>
</tr>
<tr>
<td>(m_{3,i})</td>
</tr>
</tbody>
</table>

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Scheduling tree of example 2 is shown in Fig. 13.

Number of nodes is a possible least number 20 which is smaller than the results by revised lower bound LB(J,.) (1.2) and lower bound LB(J,.) (1.5) respectively as shown in sec. 1.1.3.

Moreover, if all optimal sequences have to be found, then number of nodes is 28 for LB(J,, R,) and at least 42 even for revised lower bound LB(J,) (1.2).

Remarks: Another B. B. algorithms with LB(J,, R,) may be considered as below: [III]. First it's constructed n(n-1) nodes (J,, R,) then if lowest bound is LB(i, i2) and LB(i, i2) is smaller than LB(i, j) for almost all i(i=1~n) and j≠i2, then next it's constructed nodes (i1, R2) where last job of R2 is i2 and vise versa. [IV]. Policy of branching is to branch from a node of order one with lowest bound.

§ 3. Upper Bound in Case Where no Passing is Allowed

In this section upper bound of makespans of sequences with definite presubsequence is constructed by using Johnson's criterion for two machines and applied to B. B. algorithm for max-makespan problem in case where no passing is allowed.
3.1. Upper bound of makespans of sequences with definite presubsequence

As before, let \( J_r \) be definite presubsequence of \( r \) jobs and \( \overline{J}_r \) be the set of unordered remained \((n-r)\) jobs and \( T_*(J_r) \) be completion time of \( J_r \) on \( M_k \) \((k=1 \sim m)\). Also, let \( i_1^r i_2^r \cdots i_{n-r}^r \) be a sequence of all jobs in \( \overline{J}_r \) determined by Johnson's criterion (1.1) for each two machines \( M_k, M_{k+1} \) \((k=1 \sim m-1)\), then a sequence \( o_k^r \equiv i_1^r i_2^r \cdots i_d^k \) is a sequence which is optimal for max-makespan problem for each two machines \( M_k, M_{k+1} \) \((k=1 \sim m-1)\).

So, in order to construct the upper bound of makespans of sequences of \( n \) jobs with definite presubsequence \( J_r \), completion time \( T^r_{k+1} \) of each sequence \( i_1^r \cdots i_d^k \) on \( M_{k+1} \) \((k=1 \sim m-1)\) must be defined as shown in the following: [cf. Fig. 14]

\[
T^r_{k+1} = \text{completion time of a sequence } o_k^1 \text{ on } M_2 \text{ in case when } o_k^1 \text{ is processed on } M_1 \text{, } M_2 \text{ after the time } T_1(J_r) \text{ on } M_1, T_2(J_r) \text{ on } M_2 \text{ respectively.}
\]

\[
T^r_{k+1} = \text{completion time of a sequence } o_k^2 \text{ on } M_3 \text{ in case when } o_k^2 \text{ is processed on } M_2 \text{, } M_3 \text{ after the time } T_3(J_r) \text{ on } M_2, T_3(J_r) \text{ on } M_3 \text{ respectively.}
\]
generally,

\( T_q^{n-r} = \) completion time of a sequence \( \omega_1^{q-1} \) on \( M_q \) in case when \( \omega_1^{q-1} \) is processed on \( M_{q-1}, M_q \) after the time \( T_q^{n-r} - \sum_{i \in J_r} m_{q-1,i} \) on \( M_{q-1}, T_q(J_r) \) on \( M_q \) respectively \((q=2\sim m)\), where \( T_q^{n-r} = T_1(J_r) + \sum_{i \in J_r} \). 

Then, for the processing of any sequence of \((n-r)\) jobs in \( J_r \), its completion time on \( M_2 \) isn’t after the time \( T_2^{n-r} \) and even if its starting time on \( M_2 \) be the time \( T_2^{n-r} - \sum m_{2,i} \), its completion time on \( M_3 \) isn’t after the time \( T_3^{n-r} \) and by the same reasons its completion time on \( M_n \) isn’t after the time \( T_n^{n-r} \). So that the time \( T_m^{n-r} \) is an upper bound \( UB(J_r) \) of makespans of sequences of \( n \) jobs with \( J_r \) as their definite presubsequence: that is,

\[
UB(J_r) = T_m^{n-r} \quad (r=1 \sim n-1).
\] (3.1)

Obviously, \( UB(J_r) \) is a decreasing function of \( r \) where \( J_{r_1} \subset J_{r_2} \) \((r_1 < r_2)\) and \( UB(J_{n-1}) \) is equal to real total elapsed time \( TE(J_{n-1}) \) of a sequence uniquely determined by \( J_{n-1} \).

**Remark.** \( UB(R) \) and \( UB(J_r, R) \) can be constructed by similar devices as in former and present sections.

### 3.2. B. B. Algorithm for max-makespan problem

Upper bound \( UB(J_r) \) defined in former section can be applied to the following cases where no passing is allowed: that is,

1. By using \( UB(J_r) \) together with \( LB(J_r) \), interval of variability of makespans of sequences with definite presubsequence \( J_r \) can be estimated.

2. By using B. B. algorithm with \( UB(J_r) \), optimal sequence can be obtained for max-makespan problem and then largest makespan can be recognized.

Next, B. B. algorithm for max-makespan problem is presented.

**B. B. algorithm for max-makespan problem**

The procedure is almost the same as that of B. B. algorithm for min-makespan problem in sec. 1.1 except that maximum \( UB(J_r) \) must
be branched at each stage and return to a former node \((J_r)\) \((1 \leq r < n-1)\)
must be done when it holds \(UB(J_r) > UB(J_{n-1})\).

3.3. Numerical example for max-makespan problem

**Example** \((m=3, n=6)\) [Example 1 in sec. 1.1.3]

<table>
<thead>
<tr>
<th>Processing Time (hrs.)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_{1,i})</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>(m_{2,i})</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>(m_{3,i})</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

In this example, by reversing the order determined by Johnson's
criterion for two machines, they hold \(\omega_2^1=251346\) and \(\omega_3^8=146532\). An
eexample of the calculations of \(UB(J_r)\) will be shown for \(UB(1)\) as below:

\[
\begin{align*}
\text{Job} & \quad 1 & \quad 2 & \quad 5 & \quad 3 & \quad 4 & \quad 6 \\
M_1 & \quad 6 & \quad 18 & \quad 24 & \quad 28 & \quad 31 & \quad 33 \\
M_2 & \quad 13 & \quad 20 & \quad 32 & \quad 38 & \quad 49 & \quad 63 \\
\text{Job} & \quad 1 & \quad 4 & \quad 6 & \quad 5 & \quad 3 & \quad 2 \\
M_2 & \quad 22 & \quad 33 & \quad 47 & \quad 55 & \quad 61 & \quad 63 \\
M_3 & \quad 16 & \quad 40 & \quad 59 & \quad 69 & \quad 77 & \quad 80 \\
\end{align*}
\]

\[T_{i,j}^r = \sum_{i,j} m_{2,i} = 63 - 41 = 22. \quad UB(1) = 80.\]

Scheduling tree of example is shown in Fig. 15.

Number of nodes in scheduling tree is 39 and all optimal sequences
are 214635, 214653 with 80 hrs..

**Remarks:** inverse sequence of the optimal sequence for max-makespan
problem isn't always optimal sequence for min-makespan problem and
vise versa. For example, an inverse 536412 of an optimal sequence 214635
for this example has makespan 59 hrs. which is larger than min-makespan
57 hrs. as shown in example 1 in sec. 1.1.3, and an inverse 142653 of an
optimal sequence 356241 for min-makespan problem (example 1 in sec. 1.1.3) isn’t an optimal sequence for max-makespan problem (example in this section) because its makespan is 70 hrs.

Fig. 15. Scheduling Tree of the Example.

§ 4. Additional Remarks

Each of the B.B. algorithm with lower bound \( \text{LB}(J_r) \) in sec. 1.1, 1.2, \( \text{LB}(R_r) \) in sec. 2.1, \( \text{LB}(J_r, R_r) \) in sec. 2.2, and upper bound \( \text{UB}(J_r) \) in sec. 3.1 respectively can be programmed for computer, to these this paper doesn’t refer. But they will be presented next remarks about the efficiency and sensitivity concerning the B.B. algorithm with each of the above bounds.

4.1. Efficiency of each lower bound defined in the former sections

Revised lower bound \( \text{LB}(J_r) \) in secs. 1.1 and 1.2 is clearly more efficient than lower bound already given in Refs. [1]~[4], especially for small machine’s number. According to an example, each one among revised lower bound \( \text{LB}(J_r) \) and \( \text{LB}(R_r) \) and \( \text{LB}(J_r, R_r) \) in case where no passing is allowed is more efficient than the others as shown by the
example in each section.

But average number of nodes for many examples and number of nodes in case when all optimal sequences must be obtained may be smaller for LB(J, R) than for the others. On the other hand, complexity of the calculations of the value of lower bound of each node may slightly increase for LB(J, R). So that, if balance between the number of nodes in scheduling tree in various cases and the quantity of the calculations of the value of lower bound at each node is concerned, revised lower bound LB(J, R) may be more efficient than the others.

Also, if B.B. algorithms with lower bound LB(J, R) and upper bound UB(J, R) are applied together, then range of makespans of sequences with definite presubsequence J can be estimated and this may be useful for determining approximate solutions or ultimately optimal solutions for min-(max-)makespan problem.

4.2. Sensitivity of the "algorithm" for min-or max-makespan problem

There will happen the cases where some jobs are omitted from or new jobs participate in a present lot of jobs. In these situations, it must be again applied B.B. algorithm to a new lot of jobs from the beginning of its procedure in order to repeat whole steps. Also, even when it must be found an optimal sequence of the subset of n jobs, the situation is the same. That is, in so far as the sensitivity of the "algorithm" is concerned, B.B. algorithm isn't so effective in spite of its speed of computations for each problem.

REFERENCES


