## Lateness Models

**Contents**

1. Lawler’s algorithm which gives an optimal schedule with the minimum cost $h_{\text{max}}$
   when the jobs are subject to precedence relationship
   \[ | \text{prec} | h_{\text{max}} \]

2. A branch-and-bound algorithm for the scheduling problems with the objective to minimise lateness
   \[ | \text{r}_j | \text{L}_{\text{max}} \]

**Lawler’s Algorithm**

- Backward algorithm which gives an optimal schedule for
  \[ | \text{prec} | h_{\text{max}} \]
  $h_{\text{max}} = \max \left( h_1(C_1), \ldots, h_n(C_n) \right)$
  $h_i$ are nondecreasing cost functions

**Notation**

- makespan $C_{\text{max}} = \sum p_j$ completion of the last job
- $J$ set of jobs already scheduled
- they have to be processed during the time interval $[C_{\text{max}} - \sum p_j, C_{\text{max}}]$
- $J'$ complement of set $J$, set of jobs still to be scheduled
- $J \subseteq J'$ set of jobs that can be scheduled immediately before set $J$ (schedulable jobs)

**Example** (no precedence relationships between jobs)

<table>
<thead>
<tr>
<th>jobs</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i$</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>$h_i(C_i)$</td>
<td>$1 + C_i$</td>
<td>$1.2 C_i$</td>
<td>10</td>
</tr>
</tbody>
</table>

$J = \emptyset$, $J' = \{1, 2, 3\}$ jobs still to be scheduled

$C_{\text{max}} = 10$

- $h_1(10) = 11$
- $h_2(10) = 12$
- $h_3(10) = 10$

Job 3 is scheduled last and has to be processed in $[5, 10]$.

**Lawler’s Algorithm for $1 \mid \text{r}_j \mid \text{L}_{\text{max}}$**

**Step 1.**

$J = \emptyset$

$J' = \{1, \ldots, n\}$ \quad $k = n$

**Step 2.**

Let $j^*$ be such that

\[
\min \left( h_1 \left( \sum_{j \in J'} p_j \right), \ldots, h_n \left( \sum_{j \in J'} p_j \right) \right)
\]

Place $j^*$ in $J$ in the $k$-th order position

Delete $j^*$ from $J'$

**Step 3.**

If $J' = \emptyset$ then Stop

else \quad $k = k - 1$

go to Step 2

**Lawler’s Algorithm for $1 \mid \text{prec} \mid h_{\text{max}}$**

**Step 1.**

$J = \emptyset$

$J' = \{1, \ldots, n\}$ \quad $k = n$

**Step 2.**

Let $j^*$ be such that

\[
\min \left( h_1 \left( \sum_{j \in J'} p_j \right), \ldots, h_n \left( \sum_{j \in J'} p_j \right) \right)
\]

Place $j^*$ in $J$ in the $k$-th order position

Delete $j^*$ from $J'$

Modify $J'$ to represent the set of jobs which can be scheduled immediately before set $J$.

**Step 3.**

If $J' = \emptyset$ then Stop

else \quad $k = k - 1$

go to Step 2

$J = \{3\}$ \quad $J' = \{1, 2\}$ jobs still to be scheduled

$C_{\text{max}} = 5$

- $h_1(5) = 6$
- $h_2(5) = 6$

Either job 1 or job 2 may be processed before job 3.

Two schedules are optimal: 1, 2, 3 and 2, 1, 3

J = 3 \quad J' = 1, 2 \quad jobs still to be scheduled

C_{\text{max}} = 5

- h_1(5) = 6
- h_2(5) = 6

Either job 1 or job 2 may be processed before job 3.

or

Two schedules are optimal: 1, 2, 3 and 2, 1, 3

Automated Scheduling, School of Computer Science and IT, University of Nottingham
Example. What will happen in the previous example if the precedence \( 1 \rightarrow 2 \) has to be taken into account?

- \( J = \emptyset \)
- \( J = \{1, 2, 3\} \) still to be scheduled
- \( h_1(10) = 12 \)
- \( h_2(10) = 10 \)
- \( J = \{3\} \)
- \( J = \{1, 2\} \) still to be scheduled
- \( h_1(2) = 3 \)
- Optimal schedule: 1, 2, 3,

\( h_{max} = 10 \)

\[ J' = \{2\} \]

\[ J' = \{2\} \]

\[ h_2(10) = 10 \]

\[ h_3(10) = 10 \]

\[ J = \{3\} \]

\[ J = \{1\} \]

\[ h_1(2) = 3 \]

Optimal schedule: 1, 2, 3,

\( h_{max} = 10 \)

Branch-and-bound algorithm

- Search space can grow very large as the number of variables in the problem increases!
- Branch-and-bound is a heuristic that works on the idea of successive partitioning of the search space.

We need some means for obtaining a lower bound on the cost for any particular solution (the task is to minimise the cost).

\[ f_{bound} \leq f(x), x \in S \]

\[ f_{bound} \leq \beta \leq f(x), x \in S \]

there is no need to explore \( S \)

Branch-and-bound algorithm for 1 | \( r_j \) | \( L_{max} \)

Solution space contains \( n! \) schedules (\( n \) is number of jobs).

Total enumeration is not viable!
Branching rule:
k-1 level, j_1, ..., j_{k-1} are scheduled, j_k need to be considered if no job still to be scheduled can not be processed before the release time of j_k that is: 
\[ r_{j_k} + \min_{r_{j_l} \in \text{set of jobs not yet scheduled}} (C_{j_l} + p_{j_l}) \] 

J set of jobs not yet scheduled 
t is time when j_{k-1} is completed

Lower bound:
- Preemptive earliest due date (EDD) rule is optimal for 1 | r_j, \text{prec} | L_{\text{max}}

A preemptive schedule will have a maximum lateness not greater than a non-preemptive schedule.
- If a preemptive EDD rule gives a nonpreemptive schedule then all nodes with a larger lower bound can be disregarded.

Example.

<table>
<thead>
<tr>
<th>jobs</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_j</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>r_j</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d_j</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

- Non-preemptive schedules

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>3</th>
<th>7</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 = 3 )</td>
<td>( L_2 = 6 )</td>
<td>( L_{\text{max}} = 6 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>( L_1 = 5 )</td>
<td>( L_2 = 1 )</td>
<td>( L_{\text{max}} = 5 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Preemptive schedule obtained using EDD

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>3</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 = 3 )</td>
<td>( L_2 = 3 )</td>
<td>( L_{\text{max}} = 3 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \therefore \text{the lowest } L_{\text{max}}! \]

Example

<table>
<thead>
<tr>
<th>jobs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_j</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>r_j</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>d_j</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

\( L_{\text{B}} = 5 \)

job 1 could be processed before job 4

\( L_{\text{B}} = 6 \)

job 2 could be processed before job 3

Summary

\[ 1 | \text{prec} | h_{\text{max}}, h_{\text{max}} = \max(h_j(C_j), \ldots h_j(C_n)) \], Lawler’s algorithm

\[ 1 || L_{\text{max}} \], EDD rule

\[ 1 | r_j | L_{\text{max}} \] is NP hard, branch-and-bound is used

\[ 1 | r_j, \text{prec} | L_{\text{max}} \], similar branch-and-bound

\[ 1 | r_j, \text{prec} | L_{\text{max}} \], preemptive EDD rule