Multiple Machines

- Model Multiple Available resources
  - people
  - time slots
  - queues
  - networks of computers
- Now concerned with both allocation to a machine and ordering on that machine.
$P||C_{\text{max}}$

NP-complete from partition.

**Example**

<table>
<thead>
<tr>
<th>$j$</th>
<th>$p_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

- What is the makespan on 2 machines?
- 3 machines ?
- 4 machines ?
Approximation Algorithms

- Cannot come up with an optimal solution in polynomial time
- Will look at relative error: \( C_{\text{max}}(\text{our algorithm})/C_{\text{max}}(OPT) \)

Challenges:
- Our algorithm’s performance is different on different instances
- We can’t compute \( C_{\text{max}}(OPT) \)
Approximation Algorithms

• Cannot come up with an optimal solution in polynomial time
• Will look at relative error: \( \frac{C_{\text{max}}(\text{our algorithm})}{C_{\text{max}}(\text{OPT})} \)
• Challenges:
  – Our algorithm’s performance is different on different instances
  – We can’t compute \( C_{\text{max}}(\text{OPT}) \)

Solution:

• We will use a worst case measure on performance
• We will use a lower bound on \( C_{\text{max}}(\text{OPT}) \)
Approximation Algorithms

An algorithm $A$ is a $\rho$ approximation algorithm for a problem, if for all inputs

$$\frac{C_{\text{max}}(A)}{C_{\text{max}}(OPT)} \leq \rho$$

In addition, $A$ must run in polynomial time.

We can’t compute $C_{\text{max}}(OPT)$.

Recipe:

- Instead, we compute a lower bound $LB(OPT)$, such that
  - $LB(OPT)$ is easy to compute
  - $LB(OPT) \leq C_{\text{max}}(OPT)$.

- We then show that $C_{\text{max}}(A) \leq \rho LB(OPT)$.

Combining the previous two steps, we have:

$$C_{\text{max}}(A) \leq \rho LB(OPT) \leq \rho C_{\text{max}}(OPT)$$

which can be rewritten as

$$\frac{C_{\text{max}}(A)}{C_{\text{max}}(OPT)} \leq \rho$$
Notes:

- Must come up with a good lower bound
- Can replace $C_{\text{max}}$ with any objective.
Lower Bounds for $P||C_{max}$

- Average load
- Longest job
Lower Bounds for $P||C_{\text{max}}$

- **Average load** – $\lceil \sum p_j / m \rceil$
- **Longest job** – $p_{\text{max}} = \max_j \{p_j\}$
List Scheduling Algorithm

A Greedy Algorithm

1. Make a list of the jobs (in any order)
2. When a machine becomes available, schedule the next job on the list.
List Scheduling Algorithm

A Greedy Algorithm

1. Make a list of the jobs (in any order)
2. When a machine becomes available, schedule the next job on the list.
List Scheduling Algorithm

A Greedy Algorithm

1. Make a list of the jobs (in any order)
2. When a machine becomes available, schedule the next job on the list.

Analysis

• Let $t$ be the last time at which all machines are busy.
• $t \leq \frac{\sum j \ p_j}{m}$
• $C_{\text{max}} \leq t + p_{\text{max}} \leq \frac{\sum j \ p_j}{m} + p_{\text{max}}$.

Put this together with our lower bound:

$$C_{\text{max}} \leq t + p_{\text{max}} \leq \frac{\sum j \ p_j}{m} + p_{\text{max}} \leq 2LB \leq 2OPT$$
Improved Algorithm

- Schedule length is average load plus last job.
- When last job is small, the schedule is shorter.
- Force last job to be small – LPT (Longest Processing Time).

LPT is a $4/3$-approximation for $P\|C_{\text{max}}$.

Proof Outline
- If last job is small ($\leq 1/3OPT$) then $4/3$-approximation
- Otherwise, there are at most 2 jobs per machine and LPT is optimal.

Even better algorithms are possible: . A polynomial-time approximation scheme (PTAS) is an algorithm that, given fixed $\epsilon > 0$, returns at $(1 + \epsilon)$-approximation in polynomial time. The running time can have a bad dependence on $\epsilon$, such as $n^{O(1/\epsilon)}$.

$P\|C_{\text{max}}$ has a PTAS.
Precedence Constraints

- $P^\infty|\text{prec}|C_{\max}$ is known as project scheduling.
- $P|\text{prec}|C_{\max}$ has a 2-approximation.

What are good lower bounds for $P|\text{prec}|C_{\max}$?
Precedence Constraints

- $P_{\infty}|\text{prec}|C_{\text{max}}$ is known as project scheduling.
- $P|\text{prec}|C_{\text{max}}$ has a 2-approximation.

What are good lower bounds for $P|\text{prec}|C_{\text{max}}$?

- Average load
- $p_{\text{max}}$
- any path in the precedence graph
- the critical path is the longest path in the precedence graph.
Unit Processing Times

\[ P|p_j = 1, \text{prec}|C_{\text{max}} \text{ is NP-hard}. \]

Heuristics

- Critical Path (CP) rule
  - The job at the head of the longest string of jobs in the constraint graph has the highest priority
  - \( P|p_j = 1, \text{tree}|C_{\text{max}} \) is solved by CP.

- Largest Number of Successors First (LNS)
  - The job with the largest total number of successors in the constraint graph has highest priority.
  - For in-trees and chains, LNS is identical to CP
  - LNS is also optimal for \( P|p_j = 1, \text{outtree}|C_{\text{max}} \)

- Generalization to arbitrary processing times is possible

Fixed Number of Processors

- \( P2|p_j = 1, \text{prec}|C_{\text{max}} \) is solvable in polynomial time
- \( P3|p_j = 1, \text{prec}|C_{\text{max}} \) is a big open question.
Preemptions: $P|\text{pmtn}|C_{\text{max}}$

- McNaughton’s wrap-around rule is optimal.

Example

<table>
<thead>
<tr>
<th>$j$</th>
<th>$p_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>9</td>
</tr>
</tbody>
</table>
LP for $P|\text{pmtn}|C_{\text{max}}$

Variables: $x_{ij}$ is the time that job $j$ runs on machine $i$. $C_{\text{max}}$ is also a variable.

Constraints

- Each job runs for $p_j$ units of time
- Each machine runs for at most $C_{\text{max}}$ time.
- $C_{\text{max}}$ is more than any processing time.

\[
\begin{align*}
\text{min} & \quad C_{\text{max}} \\
\text{s.t.} & \quad \sum_{i=1}^{m} x_{ij} = p_j \quad j = 1 \ldots n \quad (3) \\
& \quad \sum_{j=1}^{n} x_{ij} \leq C_{\text{max}} \quad i = 1 \ldots m \quad (4) \\
& \quad \sum_{i=1}^{m} x_{ij} \leq C_{\text{max}} \quad j = 1 \ldots n \quad (5) \\
\end{align*}
\]

Note that LP only assigns pieces of jobs to machines. Need to also assign jobs to times.
Machines with speeds – \( Q|\text{pmtn}|C_{\text{max}} \)

- Machines \( M_1, \ldots, M_m \) with speeds \( v_1, \ldots, v_m \).
- Assume wlog that \( v_1 \geq v_2 \geq v_m \).
- Assume wlog that \( p_1 \geq p_2 \geq p_n \).
- If a job runs for one unit of time on machine \( M_i \), it uses up \( v_i \) units of processing.
- If job \( j \) runs on machine \( M_i \), then it takes \( p_j/v_i \) time units to complete.

**Example**

<table>
<thead>
<tr>
<th>( j )</th>
<th>( p_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

What are the lower bounds
Lower bounds for $Q|\text{pmtn}|C_{\text{max}}$

- What is the analog of $p_{\text{max}}$?
- What is the analog of average load?
- Are there others?
Lower bounds for $Q|\text{pmtn}|C_{\text{max}}$

- What is the analog of $p_{\text{max}}$? – $p_1/v_1$
- What is the analog of average load? – $\sum p_j/\sum v_i$
- Are there others? – Yes

General Lower Bound

$$C_{\text{max}} \geq \max \left( \frac{p_1}{v_1}, \frac{p_1 + p_2}{v_1 + v_2}, \ldots, \frac{\sum_{j=1}^{m-1} p_j}{\sum_{i=1}^n v_i}, \frac{\sum_{j=1}^m p_j}{\sum_{i=1}^n v_i} \right)$$
Lower Bound

\[ C_{\text{max}} \geq \max \left( \frac{p_1}{v_1}, \frac{p_1 + p_2}{v_1 + v_2}, \ldots, \frac{\sum_{j=1}^{m-1} p_j}{\sum_{i=1}^{m-1} v_i}, \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{n} v_i} \right) \]

What is the lower bound for our example?

Can we achieve this lower bound?
LPRT-FM

Longest Remaining Processing Time on Fastest Machines

Example 1
\[
\begin{array}{ll}
  j & p_j \\
  A & 20 \\
  B & 16 \\
  C & 2 \\
  D & 1 \\
\end{array}
\]
\[
v = (4, 2, 1)
\]

Example 2
\[
\begin{array}{ll}
  j & p_j \\
  A & 20 \\
  B & 16 \\
  C & 2 \\
  D & 1 \\
\end{array}
\]

Notes:
- LRPT-FM is optimal in continuous time
- LRPT-FM is near optimal in discrete time, for small time steps.