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.

$P||C_{\max}$

NP-complete from partition.

Example

j	p_j
1	10
2	8
3	6
4	4
5	2
6	1

- **What is the makespan on 2 machines?**
- **3 machines ?**
- **4 machines ?**

Approxmiation Algorithms

- **Cannot come up with an optimal solution in polynomial time**
- **Will look at relative error :** $C_{\max}(\textbf{our algorithm})/C_{\max}(OPT)$
- **Challenges:**
  - **Our algorithm's performance is different on different instances**
  - **We can't compute** $C_{\max}(OPT)$

Approxmiation Algorithms

- **Cannot come up with an optimal solution in polynomial time**
- **Will look at relative error :** $C_{\max}(\textbf{our algorithm})/C_{\max}(OPT)$
- **Challenges:**
  - **Our algorithm's performance is different on different instances**
  - **We can't compute** $C_{\max}(OPT)$

Solution:

- **We will use a worst case measure on performance**
- **We will use a lower bound on** $C_{\max}(OPT)$

Approximation Algorithms

An algorithm **A** is a ρ approximation algorithm for a problem, if for all inputs

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

.

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

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

Recipe:

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

Combining the previous two steps, we have:

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

which can be rewritten as

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

Notes:

- **Must come up with a good lower bound**
- **Can replace $C_{\max}$ with any objective.**

Lower Bounds for $P||C_{\max}$

- **Average load**
- **Longest job**

Lower Bounds for $P||C_{\max}$

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

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.
- $t \leq \Sigma_j \, p_j/m$
- $C_{\max} \leq t + p_{\max} \leq \Sigma_j \, p_j/m + p_{\max}$.

Put this together with our lower bound:

$$C_{\max} \leq t + p_{\max} \leq \sum_j p_j/m + p_{\max} \leq 2LB \leq 2OPT$$

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 $P||C_{\max}$.

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 ϵ, such as $n^{O(1/\epsilon)}$.

$P||C_{\max}$ 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

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

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

- **Average load**
- $p_{\max}$
- **any path in the precedence graph**
- **the critical path is the longest path in the precedence graph.**

Unit Processing Times

$P|p_j = 1, \text{prec}|C_{\max}$ **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**
  - $P|p_j = 1, tree|C_{\max}$ **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** $P|p_j = 1, outtree|C_{\max}$
- **Generalization to arbitrary processing times is possible**

Fixed Number of Processors

- $P2|p_j = 1, \text{prec}|C_{\max}$ **is solvable in polynomial time**
- $P3|p_j = 1, \text{prec}|C_{\max}$ **is a big open question.**

Preemptions: $P|\mathrm{pmtn}|C_{\max}$

- **McNaughton's wrap-around rule is optimal.**

Example

j	p_j
A	**7**
B	**10**
C	**1**
D	**4**
E	**9**

LP for $P|\text{pmtn}|C_{\max}$

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

Constraints

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

$$
\begin{aligned}
& \min C_{\max} && (1)\\
& s.t. && (2)\\
& \Sigma_{i=1}^{m} x_{ij} = p_j \quad j = 1 \ldots n && (3)\\
& \Sigma_{j=1}^{n} x_{ij} \le C_{\max} \quad i = 1 \ldots m && (4)\\
& \Sigma_{i=1}^{m} x_{ij} \le C_{\max} \quad j = 1 \ldots n && (5)\\
& && (6)
\end{aligned}
$$

Note that LP only assigns pieces of jobs to machines. Need to also assign jobs to times.

Machines with speeds – $Q|\text{pmtn}|C_{\max}$

- 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

j	p_j
A	20
B	16
C	2
D	1

What are the lower bounds

Lower bounds for $Q|\mathrm{pmtn}|C_{\max}$

- **What is the analog of** $p_{\max}$ **?**
- **What is the analog of average load ?**
- **Are there others ?**

Lower bounds for $Q|\mathrm{pmtn}|C_{\max}$

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

General Lower Bound

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

Lower Bound

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

What is the lower bound for our example?

Can we achieve this lower bound?

LRPT-FM

Longest Remaining Processing Time on Fastest Machines

Example 1

j	p_j
A	**20**
B	**16**
C	**2**
D	**1**

$$v = (4, 2, 1)$$

Example 2

j	p_j
A	**20**
B	**16**
C	**12**
D	**1**

Notes:

- **LRPT-FM is optimal in continuous time**
- **LRPT-FM is near otimal in discrete time, for small time steps.**