Note Title 4/7/2010

CS 314 - Approximation : Load Balancing

Announcements

- HW 8 is out
  due next Wed. in class

Approximation

- Problems can be hard to solve
- Exponential algorithms take too long

So: sometimes we can get (in polynomial time) an answer that is "close" to optimal

<u>Load Balancing</u> (Sec. 11.1 in book, Sec. 1 in notes)

m machines $M_1, \ldots, M_m$

n jobs given in array, t_j = job j's run time

[Want a balanced assignment:

So if A_i = jobs assigned to M_i

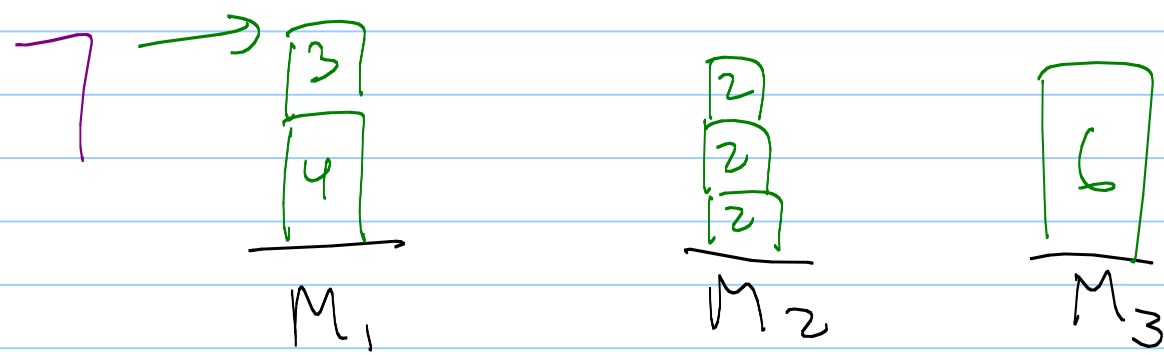
& $T_i = \sum_{j \in A_i} t_j$ is <u>load</u> on M_i,

Minimize $T = \max_i T_i$

(called <u>make span</u>)

Example: 3 machines, jobs: 2, 3, 4, 6, 2, 2

How small can makespan be?



I won't prove it, but this is NP-Hard.
(from partition)

What if we need to solve it anyway?

We'll use approxamation:

What is a natural greedy idea?

Pseudocode

Greedy-Balance:

$T_i \leftarrow 0$, $A_i \leftarrow \emptyset$ for all M_i

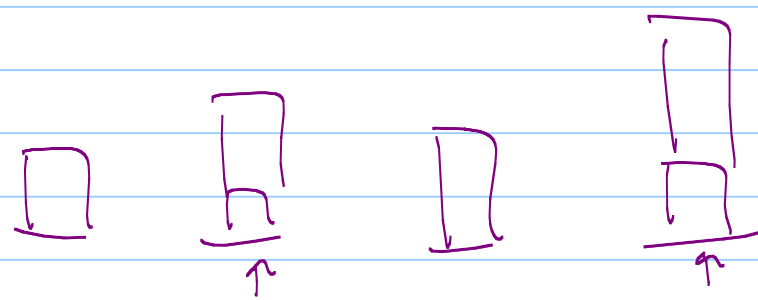
for $j \leftarrow 1$ to n

Let M_i be machine minimizing T_i

Assign job j to M_i

$A_i \leftarrow A_i \cup \{j\}$

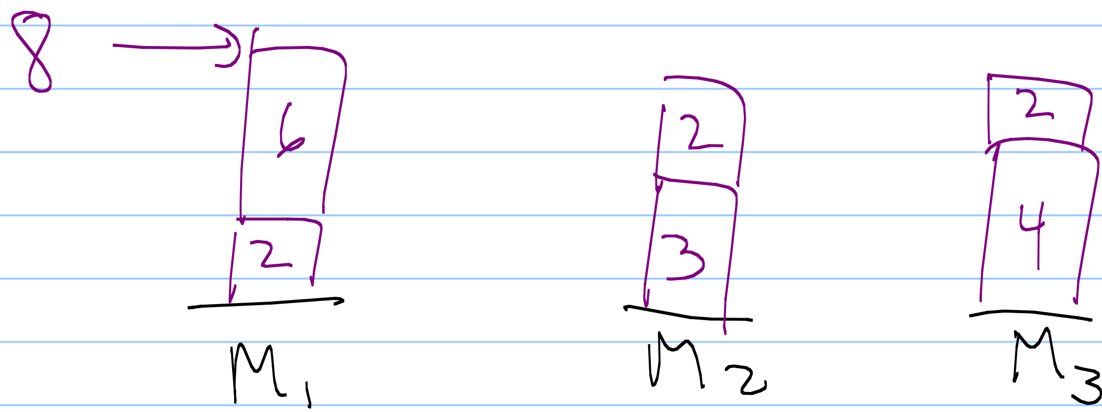
$T_i \leftarrow T_i + t_j$



Greedy Balance:

Example: 3 machines, jobs: 2, 3, 4, 6, 2, 2

How small can makespan be?



Would like to argue that we are close to optimal value T^*.

But we don't know T^*!

→ We can, however, get bounds on T^*.

our makespan $= T$

(in prev. example, $T^* = 7$)

For example:

$$\text{(1)} \quad T^* \geq \overbrace{\frac{1}{m} \cdot \sum_j t_j}^{\text{average}}$$

Why?

This is what value would be if things could be spread perfectly evenly over all machines.

if T^* were less, then $m * T^* < \sum t_i$

Another:

2. $T^* \geq \max_j t_j \geq t_j$

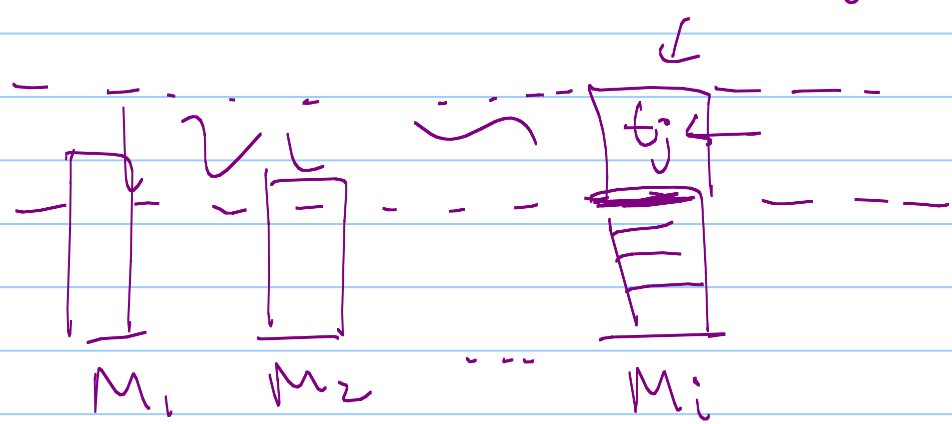
Why?

Some machine has to run the longest job.

Thm: Greedy-Balance gives a schedule with makespan $T \le 2 \cdot T^*$.

pf: Consider the machine M_i with the largest load, so $\underbrace{T_i = \max_k T_k}$.

Consider the last job that M_i runs, job j.



$T_i - t_j$ was lowest "load" on a machine

$\Rightarrow$ every other machine had $\ge T_i - t_j$

pf cont: If we add load on all machines,

$$\sum_k T_k \geq m(T_i - t_j)$$

rewrite: $T_i - t_j \leq \underbrace{\frac{1}{m}\sum_k T_k}_{} = \underbrace{\frac{1}{m}\sum_\ell t_\ell}_{\text{looks familiar!}}$

we know $T^* \geq \frac{1}{m} \cdot \sum_\ell t_\ell$

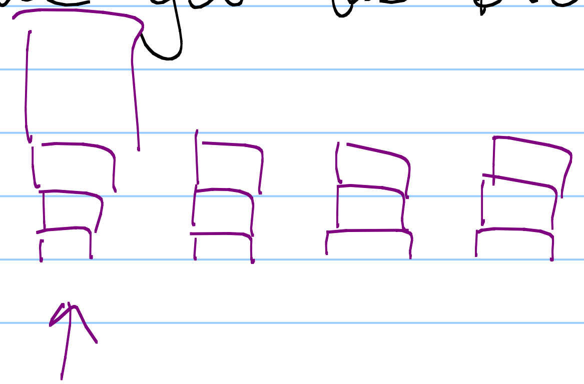
$\Rightarrow \underbrace{T_i - t_j}_{} \leq \frac{1}{m}\sum_\ell t_\ell \leq T^*$

so M_i had $\leq T^*$ on it before it got its last job.

+ (2) gives us $t_j \leq T^*$.

so $T_i = (T_i - t_j) + t_j \leq T^* + T^*$ $\boxed{\text{m}}$

Can we get this bad?



Bad case: large job at the end

If the jobs are sorted largest to smallest, we can do better!

Sorted-Balance:

$T_i \leftarrow 0$, $A_i \leftarrow \emptyset$ for all M_i

Sort jobs so that $t_1 \geq t_2 \geq \dots \geq t_n$

for $j \leftarrow 1$ to n

Let M_i be machine minimizing T_i

Assign job j to M_i

$A_i \leftarrow A_i \cup \{j\}$

$T_i \leftarrow T_i + t_j$

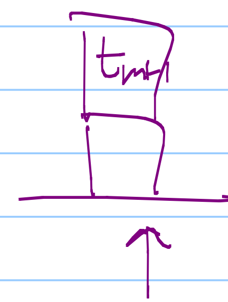
We'll use a slightly better bound on T^*

Lemma: If there are more than m jobs, then $T^* \geq 2t_{m+1}$.

3

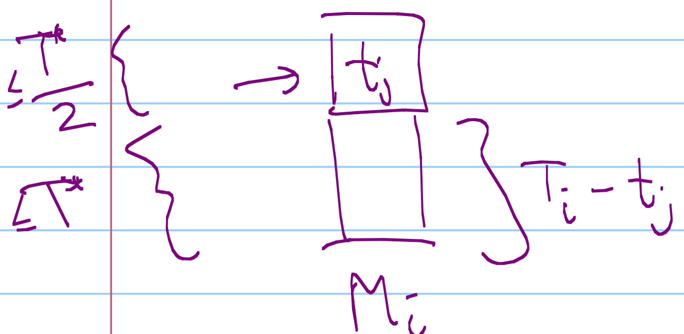
pf: First m jobs go on empty machines. Job $m+1$ gets put with a job whose processing time is $\geq t_{m+1}$

So T^* has to take at least $2 \cdot t_{m+1}$ time.



Thm: Algorithm Sorted-Balance gives $T \le \frac{3}{2} \cdot T^*$.

pf: Consider worst machine M_i, & let job j be the last job on it.



Still have $T_i - t_j \le T^*$.

If $j \le m$, done.

If $j > m$, use 3.

So $T^* \ge 2t_{m+1} \ge 2t_j$.

rewrite: $t_j \le \frac{T^*}{2}$.