5 May 2021

Greedy Approximation Algorithms
(§11.1, 11.3)

Announcement:
PSet 7 regrade deadline May 14, 2021.

Scheduling Jobs on Identical Machines.

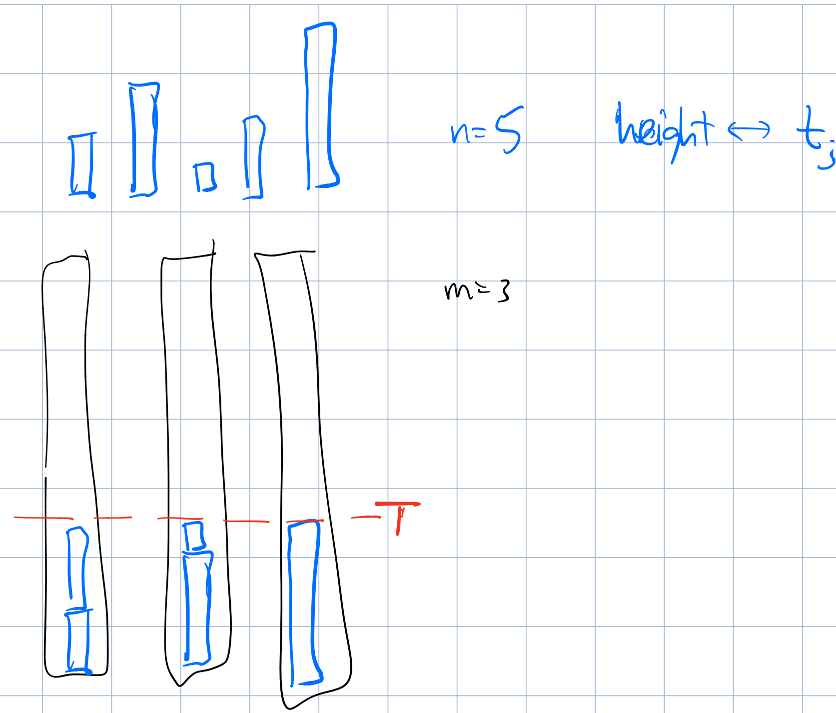
Given n jobs with processing times $t_1, t_2, \ldots, t_n$.
m machines.

Goal: Partition jobs into sets $J(1), \ldots, J(m)$.
Processing time of machine k is

$$T(k) = \sum_{j \in J(k)} t_j$$

Minimize $T = \max\{T(1), \ldots, T(m)\}$.
"makespan".

Pictorially.



GREEDY ALGORITHM 1

Initialize $J(k) = \emptyset$ and $T(k) = 0$, for all k.

for $j = 1, 2, \ldots, n$:

  Find a machine k with minimum $T(k)$.

  Assign job j to machine k:

    $J(k) = J(k) \cup \{j\}$

    $T(k) = T(k) + t_j$

endfor

output $J(1), \ldots, J(m)$.

How could this ever output a suboptimal plan?

Hint: there's an example with 3 jobs, 2 machines, where output is suboptimal.

$t_1 = 1, \quad t_2 = 1, \quad t_3 = 2. \quad m = 2.$

Optimum is $J(1) = \{1,2\}, \; J(2) = \{3\}$

$T = T(1) = T(2) = 2.$

Greedy will assign job 1 to M_1

job 2 to M_2

job 3 to M_1

$J(1) = \{1,3\} \quad J(2) = \{2\}$

$T(1) = 3 \quad T(2) = 1$ SUBOPTIMAL!

This job partitioning problem is NP-Complete even when $m=2$, by reduction from SUBSET SUM.

How to compare greedy solution with optimum?

Reasoning is analogous to "greedy stays ahead."

At the time job n is placed on some machine, k, we know that

$T(i) \geq T(k) \quad \forall \; i = 1, 2, \ldots, m.$

$$\sum_{i=1}^{m} T(i) \geq m \cdot T(k).$$

$$\| \quad \sum_{j=1}^{n-1} t_j$$

So $$\sum_{j=1}^{n} t_j \geq \sum_{j=1}^{n-1} t_j \geq m \cdot T(k).$$

$$\Downarrow$$

$$\boxed{T(k) \leq \frac{1}{m} \sum_{j=1}^{n} t_j.}$$

Before last job was placed on machine k, we managed to put T(k) work on each machine.

The optimum solution partitions all jobs into sets $J^*(1), \ldots, J^*(m)$ such that

$$\forall i \quad \sum_{j \in J^*(i)} t_j \leq T^*$$

where T^* = makespan of optimal solution.

$$\sum_{j=1}^{n} t_j = \sum_{i=1}^{m} \sum_{j \in J^*(i)} t_j \leq m \cdot T^*$$

$$\Downarrow$$

$$\boxed{T^* \geq \frac{1}{m} \sum_{j=1}^{n} t_j}$$

"opt. solution can't do better than perfect load balancing, $\frac{1}{m}$ of work on each machine."

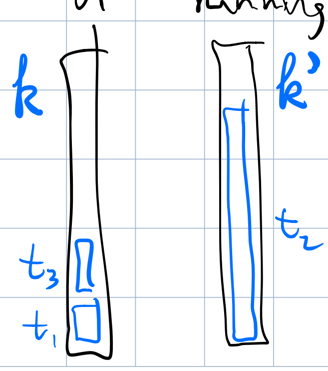
The last job we place, job n, adds its processing time to machine k,

So k ends up with load
$T(k) + t_n$.

Above, we saw $T(k) \leq \frac{1}{m}\sum_{j=1}^{n} t_j \leq T^*$.
Trivially $t_n \leq T^*$.
(opt solution must place job n on some machine. That machine's proc time is at least t_n.)

Summing up, $T(k) + t_n \leq 2 \cdot T^*$.
If we were guaranteed that the heaviest loaded machine in the greedy solution is the one with job n, we'd be done now: we have shown $T(k) + t_n \leq 2 \cdot T^*$ so greedy is a 2-approx to OPT.

Maybe some other machine k' has higher load than k at the end of running Greedy.



Suppose k' is the most heavily loaded machine in the greedy solution and j' is the last job placed on it.

Then just before j' was assigned to k',

$$\forall i \quad T(i) \geq T(k') \leftarrow \text{greedy rule.}$$

$$\sum_{j=1}^{j'-1} t_j = \sum_{i=1}^{m} T(i) \geq m \cdot T(k').$$

$$\Downarrow$$

$$T(k') \leq \frac{1}{m} \sum_{j=1}^{j'-1} t_j \leq T^*.$$

$$t_{j'} \leq T^*$$

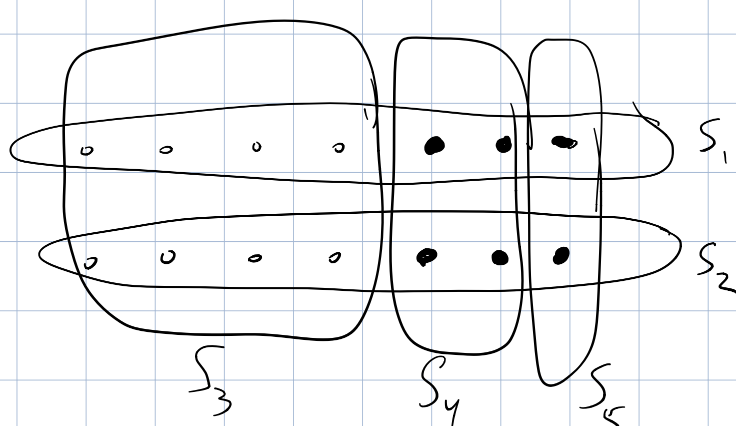
$$\underbrace{T(k') + t_{j'}}_{\substack{= \text{final load on } k' \\ = \text{makespan of greedy solution}}} \leq 2 \cdot T^*$$

SET COVER: Given n-element set U.
Subsets $S_1, \ldots, S_m \subseteq U$.
Assume $U = \bigcup_{i=1}^{m} S_i$.

Find min # of subsets whose union is U.

Greedy Set Cover

Initialize $T = \{$ uncovered elements $\} = U$.
Initialize $J = \{$ indices of chosen sets $\} = \emptyset$.
While T non-empty:
  find i such that $|S_i \cap T|$ is maximum.
  $J = J \cup \{i\}$
  $T = T \setminus S_i$
endwhile
output $\{ S_j \mid j \in J \}$.



Greedy picks S_3, S_4, S_5.

Generalize this to $2 \cdot (2^k - 1)$ elements such that S_1 & S_2 cover them all but greedy can be fooled into picking **k** sets rather than **2**.

$$k \approx \log_2(n) - 1.$$

We'll see (next lecture) greedy set cover is a $O(\log n)$ approximation in the worst case.