Section 2.3: Scheduling to minimize makespan

Makespan Scheduling on Parallel Machines

Input:

m machines

n jobs with processing times $p_1, p_2, \dots, p_n \in \mathbb{Z}^+$

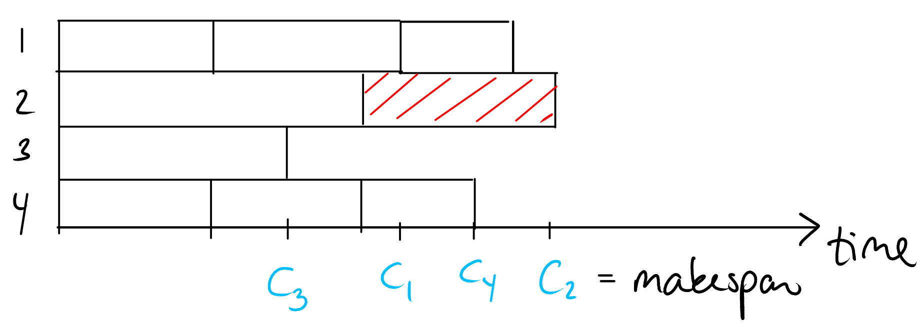
Output:

Assignment of jobs to machines s.t. the makespan is minimized

time when last job finishes

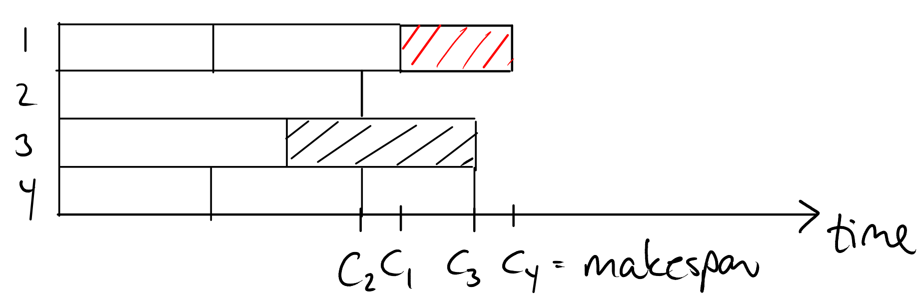
Ex:

Machines



$$\text{makespan} = \max\{C_1, C_2, C_3, C_4\} = C_2$$

How could this schedule be improved?



Local Search Alg:

Repeat
    job ℓ $\leftarrow$ job that finishes last
    If there is any machine i where job ℓ would finish earlier
        Move job ℓ to machine i
Until job ℓ is not moved

Theorem 2.5

The local search alg. is a $(2-\frac{1}{m})$-approx. alg.

Proof:

Lower bounds on OPT:

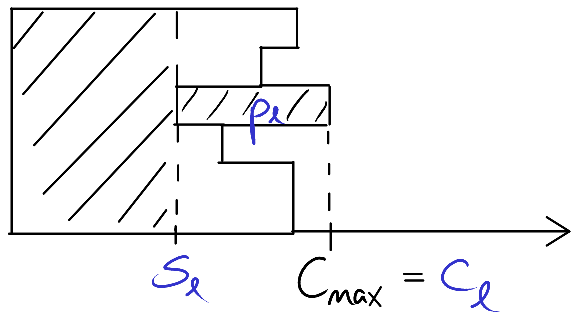
$$OPT \geq p_{max} = \max_{1 \leq j \leq n} p_j ,$$

because the machine i with the longest job j has $C_i \geq p_j$.

$$OPT \geq \frac{P}{m}, \quad \text{where } P = \sum_{j=1}^{n} p_j$$

since this is the average completion time of the machines.

Upper bound on alg.'s makespan:



$P \geq m \cdot S_\ell + p_\ell$, since all machines are busy until S_ℓ

$\Downarrow$

$S_\ell \leq \frac{P - p_\ell}{m}$

$p_\ell \leq p_{max}$

$$
\begin{aligned}
C_{max} &= S_\ell + p_\ell \\
&\leq \frac{P - p_\ell}{m} + p_\ell \\
&= \frac{P}{m} + \left(1 - \frac{1}{m}\right) p_\ell \\
&\leq OPT + \left(1 - \frac{1}{m}\right) OPT \\
&= \left(2 - \frac{1}{m}\right) OPT
\end{aligned}
$$

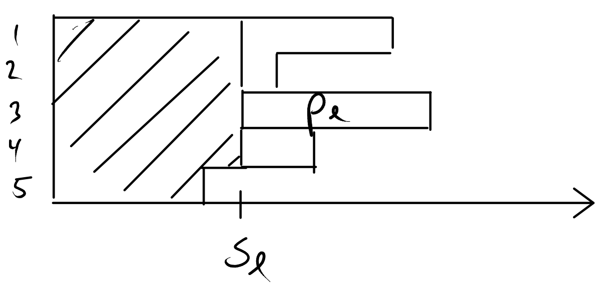
$\square$

What would be a natural greedy alg.?

List Scheduling (LS)

For $j \leftarrow 1$ to n
  schedule job j on currently least loaded machine

What is the approx. ratio of LS?
What properties of the local search alg. did we use to prove $2-\frac{1}{m}$?
We used only the fact that all machines are busy at least until S_ℓ.
Is this also true for LS?
Yes:



LS would not have placed job ℓ on machine 3.

Theorem 2.6: LS is a $(2-\frac{1}{m})$-approx. alg.

Note that $\frac{C_\ell}{OPT} < 2-\frac{1}{m}$, unless $p_\ell = p_{max}$

Thus, it seems advantageous to schedule short jobs last.

Longest Processing Time (LPT)

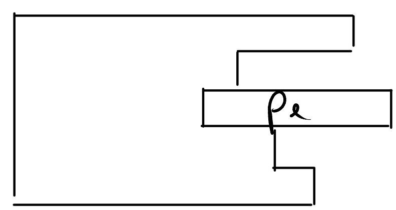
For each job j, in order of decreasing processing times
Schedule job j on currently least loaded machine

Theorem 2.7: LPT is a $\left(4/3 - \frac{1}{3m}\right)$-approx. alg.

<u>Proof</u>:

Number the jobs s.t. $p_1 \geq p_2 \geq \dots \geq p_n$.
Then the indices indicate the order in which the jobs are scheduled.

Let job ℓ be a job to finish last:



We can assume that $\ell = n$:

Let $I = \{p_1, p_2, \dots, p_n\}$ and $I' = \{p_1, p_2, \dots, p_\ell\}$.
Then, $LPT(I) = LPT(I')$, since jobs $\ell+1, \dots, n$ finish no later than job ℓ.
Moreover, $OPT(I') \leq OPT(I)$.
Thus, if we prove $LPT(I')/OPT(I') \leq 4/3$, we have proven $LPT(I)/OPT(I) \leq 4/3$ (since $LPT(I)/OPT(I) \leq LPT(I')/OPT(I')$).

(Or said in a different way, we can ignore the jobs $\ell+1, \dots, n$.)

Thus, we can assume that no job is shorter than job ℓ. (This will be used in Case 2 below.)

Case 1: $p_\ell \leq \frac{1}{3} \cdot OPT$

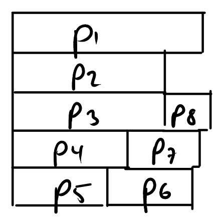
By the proof of Thm 2.5,

$$LPT \leq OPT + \frac{m-1}{m} p_\ell \leq OPT + \frac{m-1}{m} \cdot \frac{1}{3} OPT$$
$$= \left(\frac{4}{3} - \frac{1}{3m}\right) OPT$$

Case 2: $p_\ell > \frac{1}{3} \cdot OPT$

In this case, all jobs are longer than $\frac{1}{3} \cdot OPT$. Hence, in OPT's schedule, each machine has ≤ 2 jobs, i.e., $n \leq 2m$.

In this case, $LPT = OPT$:



Proof of this claim: Exercise 2.2

$\square$

From the proof of Thm 2.7 we learned:
If job ℓ is longer than $1/3 \cdot OPT$, then $LPT = OPT$.
Otherwise, $LPT \leq OPT + p_\ell \leq 4/3 \cdot OPT$.
(Recall that job ℓ is the job to finish last.)

Could we balance the two cases better?

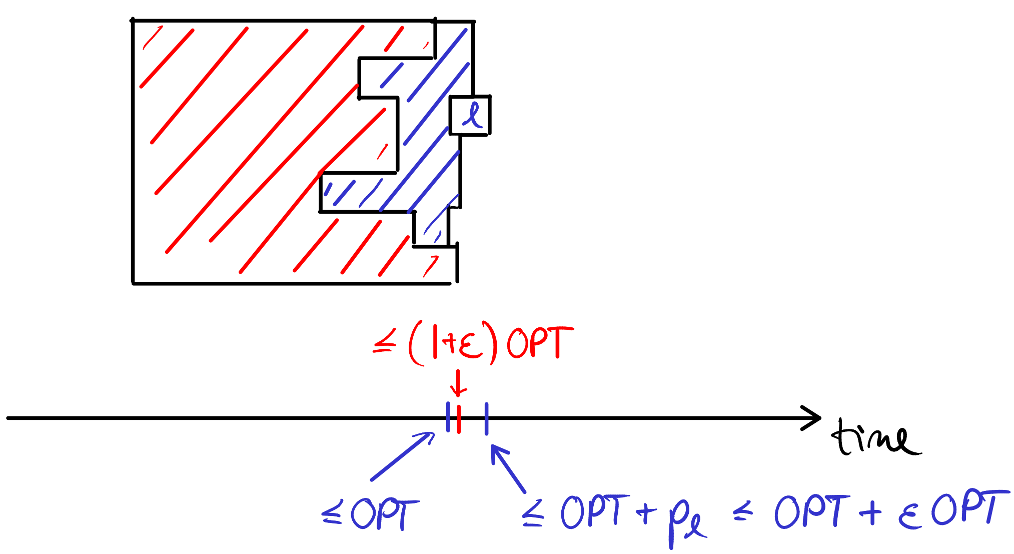
What if we first schedule all jobs of length $\geq \frac{1}{4} \cdot OPT$ optimally, and then use LPT for the remaining jobs?
What would the approximation ratio be?
Does the schedule of the long jobs have to be optimal?

Section 3.2: Makespan Scheduling – A PTAS

Idea for PTAS:

Partition the jobs into two sets (long and short jobs):

$$\underbrace{\overbrace{p_1, p_2, \dots, p_x}^{> \varepsilon \cdot OPT}}_{\text{Schedule using rounding and dyn. prg. as for bin packing}}, \quad \underbrace{\overbrace{p_{x+1}, \dots, p_n}^{\le \varepsilon \cdot OPT}}_{\text{Then use LPT}}$$



We will derive a family of algorithms with an algorithm, B_k, for each $k \in \mathbb{Z}^+$. $(\varepsilon = \frac{1}{k})$

How to identify long / short jobs when we don't know OPT?

Scheduling the long jobs:

(1) „Guess" an optimal makespan T

(2) The long jobs are those longer than T/k^2.
Round down each job size to the nearest multiple of T/k^2.

(3) Use dyn. prg. to check whether optimal makespan $\leq T$ for rounded long jobs.

Do binary search for T on the interval $[L, U]$, where

$$L = \max\left\{\left\lceil \frac{P}{m} \right\rceil, p_{max}\right\}$$

$$U = \left\lfloor \frac{P - p_{max}}{m} + p_{max} \right\rfloor = \left\lfloor \frac{P + (m-1)\, p_{max}}{m} \right\rfloor$$