LECTURE 5 29/10/2020

GREEDY ALGORITHMS

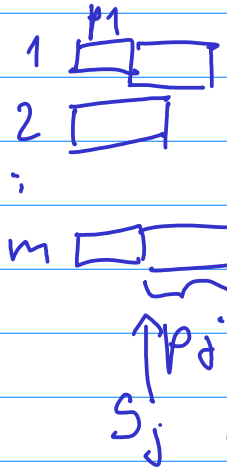
SCHEDULING JOBS ON IDENTICAL MACHINES & LOCAL SEARCH - cont'd

Recall

ALGORITHM LOCAL SEARCH FOR SCHEDULING

1. Start with any schedule
2. if there is a job j s.t. $C_j = C_{MAX}$ and $C_{MIN} < S_j$
   reassign it on a machine with minimum completion time.
   else OUTPUT the current schedule. STOP
3. repeat STEP 2.

$p_1, p_2, \dots p_n$ m



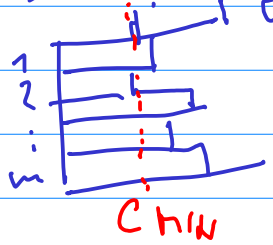
$C_{MAX} = \max_j C_j$

$C_{MIN} = \min_i \max_{j \in I_i} C_j$

APPROXIMATION RATIO OF LOCAL SEARCH ALGORITHM ?

i) $OPT \geq p_j \quad \forall j \in \{1, \dots n\}$ $\quad OPT \geq \max_j p_j$

ii) $OPT \geq C_{MIN}$ for any schedule

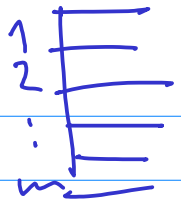


Consider the schedule constructed by the alg. and consider the last job ... p_j

$C_{MAX} = S_j + p_j \leq C_{MIN} + p_j \leq 2OPT$

All machines are busy till S_j.
How long can this take?

Note $S_j \le \frac{1}{m} \sum_{l=1}^{n} p_l$



and even

$S_j \le \frac{1}{m} \left(\sum_{l=1}^{n} p_l - p_j \right)$

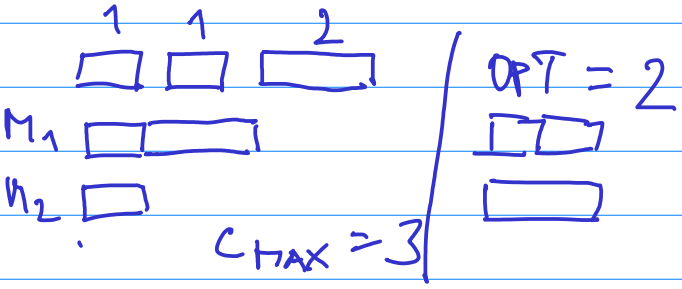
an upper bound on the completion time of the fastest machine after processing all jobs but p_j

$$C_{MAX} \le \frac{1}{m} \left(\sum_{l=1}^{n} p_l - p_j \right) + p_j = \underbrace{\frac{1}{m} \sum_{l=1}^{n} p_l}_{\le OPT} + \frac{m-1}{m} \underbrace{p_j}_{\le OPT}$$

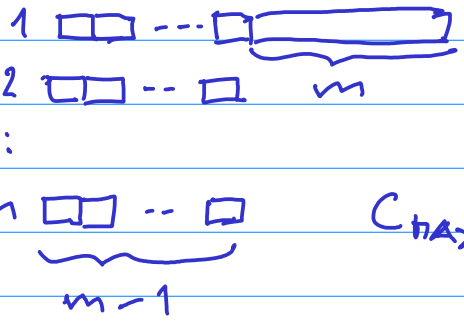
$$\le \left(2 - \frac{1}{m}\right) OPT .$$

Is a better bound possible?

$m = 2$ ... $2 - \frac{1}{2} = \frac{3}{2}$



$m > 2$



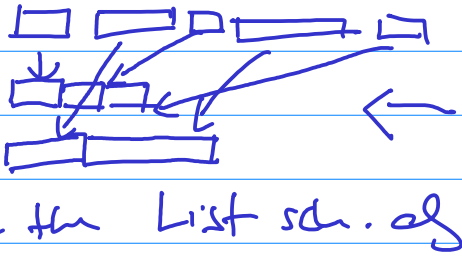
$C_{MAX} = 2m - 1$ $\quad$ $OPT = m$

$\Rightarrow$ the bound on the approx. ratio of the local search alg. is tight

---

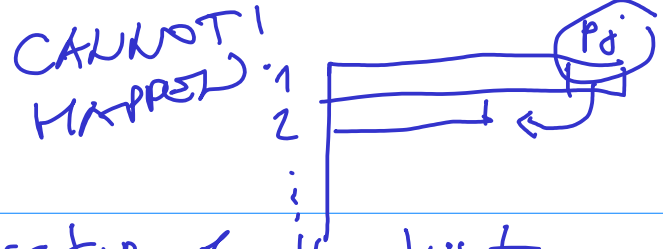
LIST SCHEDULING ALGORITHM

1. order the jobs arbitrarily
2. process the jobs one by one and assign the job to the least heavily loaded machine



APPROXIMATION RATIO?

HINT: What will the local search algorithm do with the schedule of the List sch. alg.?



⇒ THM: The approximation ratio of the list scheduling algorithm is 2 (resp., $2 - \frac{1}{m}$).

ONLINE ALGORITHMS

- the input is revealed in a step by step manner - job by job, and the decissions have to be make before knowing the future jobs

E.g. you know m .. # of machines
for each job p_j you have to schedule it before knowing $p_{j+1}, \ldots$

.... COMPETITIVE RATIO

THM: The GREEDY algorithm has competitive ratio 2 (resp. $2 - \frac{1}{m}$).

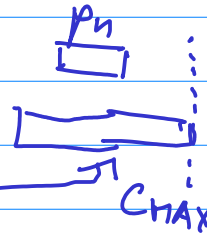
LARGEST PROCESSING TIME first algorithm (LPT)

a variant of the greedy list scheduling:

LPT
1. order jobs according to their processing time $p_1 \geq p_2 \geq \ldots \geq p_n$
2. as in greedy

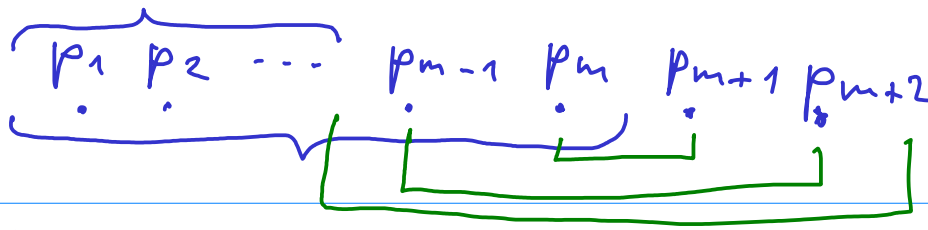
Analysis: without loss of generality assume that the job p_n completed last

why? if not, discard all jobs in the sequence after the last completed job ⇒ new instance I'



→ OPT will not increase — $OPT(I') \leq OPT$
→ LPT schedule will not shorten — $ALG(I') \geq ALG$

⇒ if Apx ratio OK for I', then for I too.



Distinguish two cases

- $p_n \le \frac{OPT}{3}$ $\quad C_{MAX} = S_n + p_n \le OPT + \frac{OPT}{3} = \frac{4}{3} OPT$
- $p_n > \frac{OPT}{3}$ ☝ in the opt schedule, at most two jobs are scheduled on any machine

☝ if $n \ge m+1$ $\quad OPT \ge OPT\{p_1, \dots, p_{m+1}\} \ge p_m + p_{m+1}$

☝ if $n \ge m+2$ $\quad$ at most $m-2$ machines will have a single job

a job of size $\ge p_{m-2+1}$ will be scheduled in a pair

$$OPT \ge p_{m-2+1} + p_{m+2}$$

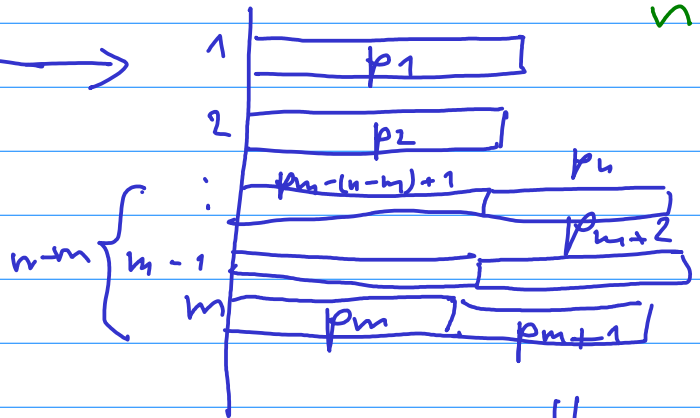
☝ if $n \ge m+1$ $\quad$ in the same way

$$OPT \ge p_{m-1+1} + p_{m+1}$$

$$\vdots$$

☝ $\quad OPT \ge p_{m-(n-m)+1} + p_{m+(n-m)}$ ($m+(n-m) = n$)

LPT assigns jobs ⟶



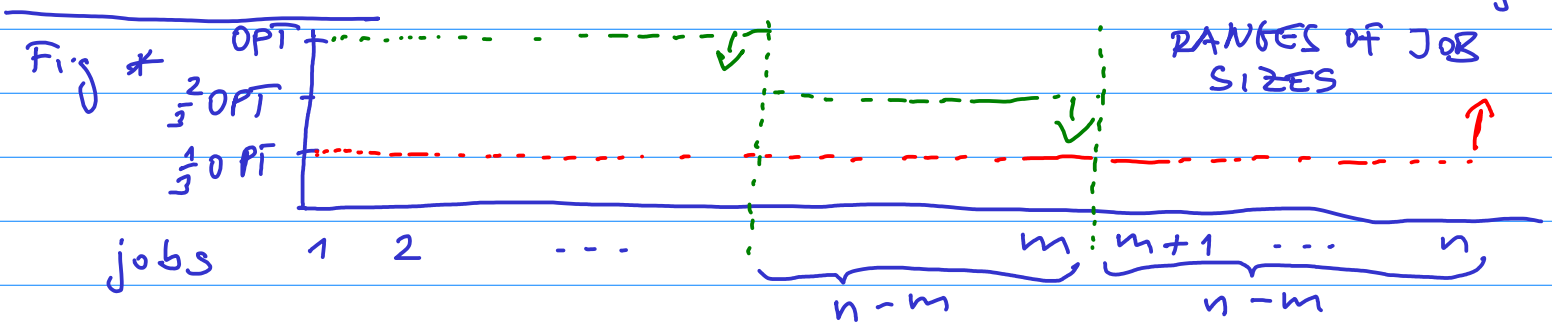
☝ At least $n-m$ jobs among $p_1, \dots, p_m$ have size $< \frac{2}{3} OPT$.

Proof: Fig * and ☝ above.

$\Rightarrow$ LPT finds the optimal schedule!! ... $C_{MAX} = OPT$

THM: the LPT algorithm has approximation ratio $\le \frac{4}{3}$.

Fig *



THE BIN PACKING PROBLEM

INPUT: n items ... $a_1, a_2, \dots, a_n \in [0,1]$

OUTPUT: partitioning of $\{1,2,\dots,n\}$ into $I_1, \dots, I_m$ s.t. $\sum_{j \in I_i} a_j \le 1 \quad \forall i$

OBJECTIVE: minimize m -- # of bins

GREEDY ALGORITHM -- FIRST FIT

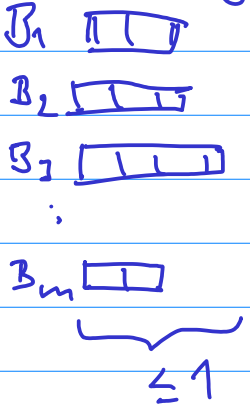
1. for $j = 1$ to n
2. let i be the first bin that item j fits into
   place the item j in the bin i
3. if there is no such bin, use a new bin

BEST FIT -- put j in the most full in which j fits

ANY FIT -- put j in any bin where it fits

THM: Every ANY FIT algorithm is a 2-approx. alg.

Proof: let $B_\ell = \sum_{j \in I_\ell} a_j$



i) ii) $\forall \ell \in \{1, \dots, m-1\}: B_\ell + B_{\ell+1} > 1$

$B_1 + B_m > 1$

$$2\left(\sum_{j=1}^{n} a_j\right) = 2\sum_{\ell=1}^{m} B_\ell > m$$

iii) $OPT \ge \sum_{j=1}^{n} a_j \Rightarrow m < 2\,OPT$ ∎

(HW) Is $\left(\frac{3}{2} - \frac{1}{n}\right)$-approximation possible? HINT: Partition problem.