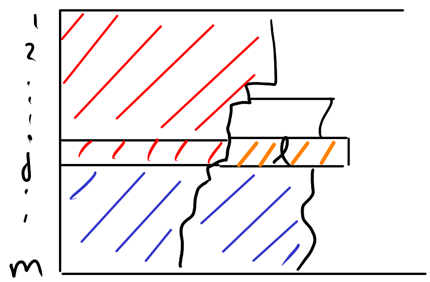
Exercise 2.2:

If $p_\ell > \frac{1}{3} OPT$, $LPT = OPT$

Proof:

$n \leq 2m$



Assume for the sake of contradiction $p_j + p_\ell > OPT$, and consider an optimal schedule.

In that schedule no two red jobs are combined, and job ℓ is not combined with a red job.

Furthermore, no blue job can be combined with a red job, since the blue jobs are at least as large as job ℓ.

Thus, the j red jobs must be scheduled on separate machines, and they cannot be combined with job ℓ or any of the $2(m-j)$ blue jobs.

This gives a total of $2(m-j)+1$ jobs that must be scheduled on $m-j$ machines.
Thus, there must be a machine with at least three of these jobs that each have a size of $\geq p_\ell > \frac{1}{3} \cdot OPT$.

Hence, the machine has a total load > OPT ↯

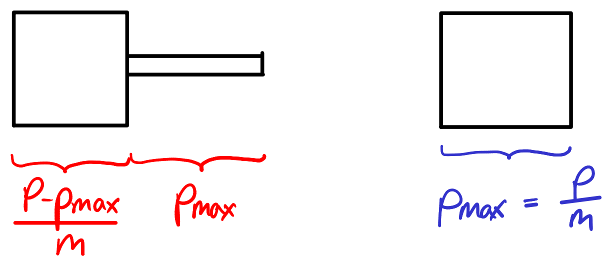
Exercise: Give an instance I, where $LS(I) = (2-\frac{1}{m})\,OPT$.

To prove $LS(I) \leq (2-\frac{1}{m}) \cdot OPT(I)$, we used

- $OPT \geq \frac{P}{m}$
- $OPT \geq p_{max}$
- $LS \leq \frac{P-p_{max}}{m} + p_{max}$

For $LS = OPT$, all inequalities need to be tight:

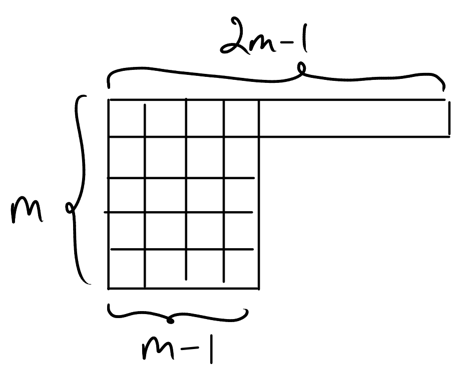
LS: OPT:



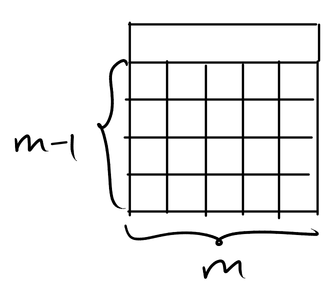
$m\cdot(m-1) \times \boxed{1}$

$1 \times \boxed{\quad m \quad}$

LS:



OPT:



Section 3.2: Makespan Scheduling – A PTAS

Sketch of PTAS:

1. Schedule long jobs ($> \varepsilon \cdot OPT$) using rounding and dyn. prg.
   $\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

2. Add short jobs ($\leq \varepsilon \cdot OPT$) to the schedule using LPT.
   $\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

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

- We need the short jobs to be $\leq \varepsilon \cdot OPT$ to ensure the approx. factor. For this purpose, we could use any lower bound on OPT, like P/m.

- But we also need the long jobs to be $\geq \varepsilon \cdot OPT$ to ensure the running time.

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

Scheduling the long jobs:

(1) „Guess" an optimal makespan T.
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

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

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

where P is the total size of long jobs.

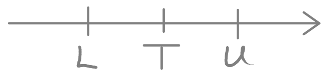
$B_k(I)$

$L \leftarrow \max\left\{ \left\lceil \frac{P}{m} \right\rceil, p_{max} \right\}$; $\quad U \leftarrow \left\lceil \frac{P}{m} + \left(1 - \frac{1}{m}\right) p_{max} \right\rceil$

Binary search for T

While $L \neq U$

$T \leftarrow \frac{1}{2} \lceil L + U \rceil$



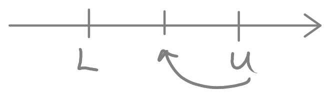
$I_\ell \leftarrow \{ \text{job } j \in I \mid p_j > T/k \}$ // Update set of long jobs

$I'_\ell \leftarrow$ I_ℓ with each job size rounded down to nearest multiple of T/k^2

Use dyn. prg. to pack I'_ℓ in bins of size T

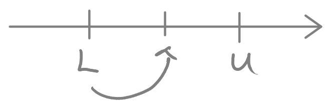
If #bins $\leq m$

$U \leftarrow T$



else

$L \leftarrow T + 1$



$S'_\ell \leftarrow$ schedule of I'_ℓ corresponding to the packing found by dyn. prg.

$S_\ell \leftarrow$ schedule of I_ℓ corresponding to S'_ℓ

$S \leftarrow$ schedule of I obtained by adding short jobs to S_ℓ using LPT

Dyn. prg. as for bin packing:

S_ℓ' places $\leq k$ jobs on each machine:

Each long job has size $\geq T/k$

Since T/k is a multiple of T/k^2, each job in I_ℓ' also has size $\geq T/k$.

There are $\leq k^2$ different job sizes in I_ℓ', since no job is longer than T.

Hence, the configuration of a machine can be represented by a vector $(s_1, s_2, \dots, s_{k^2})$, where $0 \leq s_i \leq k$.

Thus, $|\mathcal{C}| \leq k^{k^2}$.

Table (B):

$\leq k^2$ dimensions (one for each size in I_ℓ')

n_i+1 rows in dim. i (n_i = # items of size $i \cdot T/k^2$ in I_ℓ')

$$\boxed{B(n_1, \dots, n_{k^2}) = 1 + \min_{\vec{s} \in \mathcal{C}} \{ B(n_1 - s_1, \dots, n_{k^2} - s_{k^2}) \}}$$

is the min. #bins of size T it takes to pack n_i items of size $i \cdot T/k^2$, $0 \leq i \leq k^2$.

Running time:

table entries: $O(n^{k^2})$

Time per entry: $|\mathcal{C}| \leq k^{k^2}$

iterations of while loop: $\log(U-L) \leq \log(p_{max})$

Total time: $O\left((nk)^{k^2} \cdot \log(p_{max})\right)$

Approximation ratio:

When B_k terminates the while loop,

$$\text{makespan}(S_\ell') = T = \text{OPT}(I_\ell')$$

Since each of the $\leq k$ jobs on a machine loses $< T/k^2$ in the rounding,

$$\text{makespan}(S_\ell) < \text{makespan}(S_\ell') + k \cdot \frac{T}{k^2}$$

$$= T + \frac{T}{k}$$

$$= \left(1+\frac{1}{k}\right) \text{OPT}(I_\ell')$$

$$\leq \left(1+\frac{1}{k}\right) \text{OPT}(I)$$, since $I_\ell \subseteq I$, and the job sizes are rounded <u>down</u> to obtain I_ℓ'.

Thus, if the last job to finish is a long job,

$$B_k(I) < \left(1+\frac{1}{k}\right)\text{OPT}(I).$$

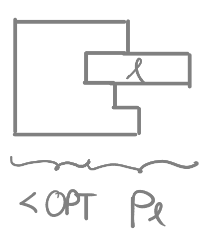
Otherwise, the last job to finish has

$$p_\ell \leq \frac{T}{k} \leq \frac{\text{OPT}(I_\ell')}{k} \leq \frac{\text{OPT}(I)}{k}$$

Hence,

$$B_k(I) < \text{OPT}(I) + p_\ell \leq \left(1+\frac{1}{k}\right)\text{OPT}$$

By the same argument as in the analysis of LS:



Thus, in both cases, $B_k(I) < \left(1+\frac{1}{k}\right)\text{OPT}.$

Theorem 3.7 : $\{B_k\}$ is a PTAS

<u>Proof</u>:
Let $k = \lceil \frac{1}{\varepsilon} \rceil$. Then,
B_k achieves an approx. factor of $1+\varepsilon$ with running time

$$O\left(\left(\frac{n}{\varepsilon}\right)^{\left(\frac{1}{\varepsilon}\right)^2} \log(p_{max}) \right).$$

If $\varepsilon \in O(1)$, this is poly. in the input size, since it takes $\geq \log(p_{max})$ bits to represent the job sizes. $\square$

$\{B_k\}$ is <u>not</u> a FPTAS, since the running time is exponential in $\frac{1}{\varepsilon}$.
Note that we did not expect a FPTAS, since the problem is <u>strongly</u> NP-complete...

The problem is strongly NP-complete, meaning that even the special case where $\exists$ polynomial q s.t. $p_{max} \le q(n)$, for all input instances, is NP-complete.

(This means that, in contrast to Knapsack, $\nexists$ pseudopoly. alg., unless $P = NP$.)

This implies that $\boxed{\nexists \text{FPTAS, unless } P = NP}$:

Assume to the contrary that $\exists$FPTAS for the problem, i.e., $\exists$ family of algorithms $\{A_\varepsilon\}$, $\varepsilon > 0$, with approx. factor $1+\varepsilon$ and running time poly. in n and $\frac{1}{\varepsilon}$.

Consider the special case of the problem where $\exists$ polynomial q s.t. $p_{max} \le q(n)$, for all instances.

In this case, $P \le n \cdot q(n) \equiv p(n)$.

For $\varepsilon = \frac{1}{p(n)}$,

- $\frac{1}{\varepsilon} = p(n)$. Since the running time of A_ε is poly. in $\frac{1}{\varepsilon}$ and n, the running time of A_ε is a poly. of n.
- $A_\varepsilon(I) \le \left(1 + \frac{1}{p(n)}\right) \cdot OPT(I)$, for any input I

  $< OPT(I) + 1$, since $OPT(I) < P \le p(n)$

  Thus, since $A_\varepsilon(I)$ is integer, $A_\varepsilon(I) = OPT(I)$.

If $P \ne NP$, this contradicts the fact that the problem is strongly NP-complete.