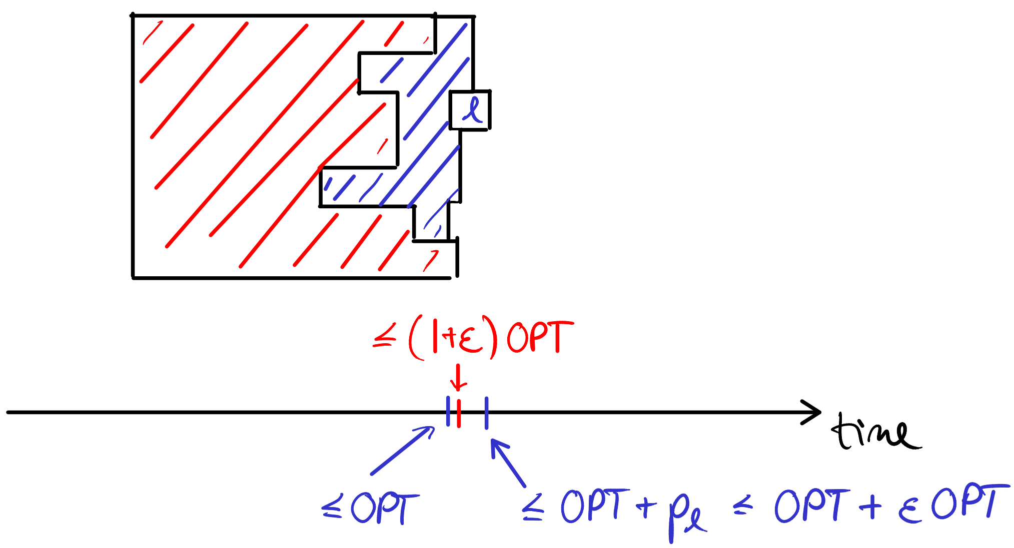
Section 3.2: Makespan Scheduling – A PTAS

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?

We need the short jobs to be $\le \varepsilon \cdot OPT$ 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 $\ge \varepsilon \cdot OPT$ to ensure the approx. factor as well as the running time.

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 $\le T$ for rounded long jobs.

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

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

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

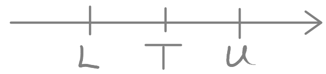
$B_k(I)$

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

Binary search for T

While $L \neq U$

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



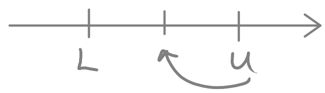
$\quad I' \leftarrow \{ \text{job } j \in I \mid p_j > T/k \}$ // Update set of long jobs

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

$\quad$ Use dyn. prg. to pack I'' in bins of size T

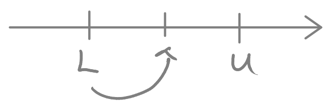
$\quad$ If #bins $\leq m$

$\quad\quad U \leftarrow T$



$\quad$ else

$\quad\quad L \leftarrow T+1$



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

$S' \leftarrow$ schedule of I' corresponding to S''

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

Dyn. prg. as for bin packing:

S'' 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'' also has size $\geq T/k$.

There are $\leq k^2$ different job sizes in I'', 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, $|\mathcal{C}| \leq (k+1)^{k^2}$.

Table (B):

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

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

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

Running time:

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

Time per entry: $|\mathcal{C}| \leq (k+1)^{k^2}$

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

Total time: $O(n^{k^2} (k+1)^{k^2} \log(p_{max}))$

Approximation ratio:

When B_k terminates the while loop,

$$\text{makespan}(S'') = T = \text{OPT}(I'')$$

S'' 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'' also has size $\geq T/k$.

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

$$\begin{aligned}\text{makespan}(S') &< \text{makespan}(S'') + k \cdot \frac{T}{k^2} \\ &= T + \frac{T}{k} \\ &= \left(1+\frac{1}{k}\right) \text{OPT}(I'') \\ &\leq \left(1+\frac{1}{k}\right) \text{OPT}(I)\end{aligned}$$

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

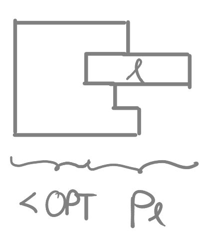
$$B_k(I) < \left(1+\frac{1}{k}\right)\text{OPT}(I).$$

Otherwise, the last job to finish has $p_\ell \leq \frac{T}{k} \leq \frac{\text{OPT}(I)}{k}$.

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>:

B_k achieves an approx. factor of $1+\varepsilon$ with running time

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

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. $p_{max} \leq q(n)$, for all input instances, is NP-complete.

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

Assume to the contrary that $\exists$FPTAS for the problem, i.e.,

$\forall \varepsilon > 0 : \exists\ (1+\varepsilon)$-approx alg. A_ε with running time poly. in n and $\frac{1}{\varepsilon}$.

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

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

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

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

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

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

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