# Section 3.2: Makespan Scheduling - A PTAS

Longest Processing Time:
From the proof of Thm 2.7 and Exercise 2.2, we learned that if the job $\ell$ to finish last has length $p_\ell > \frac{1}{3} \cdot OPT$, then $LPT = OPT$.
Otherwise, $LPT \leq OPT + p_\ell$.

Idea for PTAS:

Partition the jobs into two sets:

$$\overbrace{\underbrace{p_1, p_2, \dots, p_x}_{\text{Schedule optimally}}}^{> \varepsilon \cdot OPT}, \overbrace{\underbrace{p_{x+1}, \dots, p_n}_{\text{Use LPT}}}^{\leq \varepsilon \cdot OPT}$$

Schedule optimally, if $m, x \in O(1)$.
Otherwise, use dyn. prg. as for bin packing

We will derive a family of algorithms with an algorithm for each $k \in \mathbb{Z}^+$

Let $P = \sum_{j=1}^{n} p_j$ (as before).

Job $j$ is short, if $p_j \leq \frac{P}{km}$, i.e., if it is at most $1/k$ of the average machine load.

Otherwise, it is long.

Algorithm:

Schedule long jobs first.

Then, add short jobs using LPT.

#long jobs $< km$

Hence, #schedules of long jobs $< m^{km}$ (choose one of $m$ machines for each job).

Thus, if $k, m \in O(1)$, we can find an optimal schedule for the long jobs in time $O(1)$.

Otherwise, we can round job sizes and do dyn. prg. as for the bin packing problem.

The alg. will be poly. in $m$, but not in $k$.

Thus, the algorithm will be a PTAS, not an FPTAS.

Idea for the long jobs:

(1) „Guess" an optimal makespan $T$

(2) Round each job size down to the nearest multiple of $\frac{T}{k^2}$

(3) Use dyn. prg. to check whether $\exists$ schedule of makespan $\leq T$ for the rounded jobs.
If not, then $OPT > T$ (for the rounded job sizes, and hence, for the original job sizes)

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

$$L = \max\{\lceil P/m \rceil, p_{max}\}$$

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

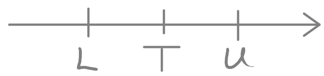
## $B_k(I)$

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

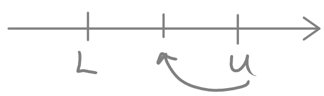
$U \leftarrow \left\lceil \frac{P + (m-1)\, p_{max}}{m} \right\rceil$

While $L \neq U$

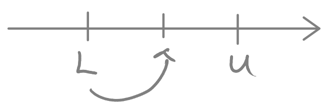
- $T \leftarrow \left\lceil \frac{1}{2}(L+U) \right\rceil$




- $I_\ell \leftarrow \{ \text{job } j \in I \mid p_j > T/k \}$ // long jobs
- $I'_\ell \leftarrow I_\ell$ with each job length rounded down to nearest multiple of $T/k^2$
- If $\exists$ schedule $S'$ of $I'_\ell$ s.t. makespan$(S') \leq T$
  - $U \leftarrow T$




- Else
  - $L \leftarrow T$




$S \leftarrow$ schedule of $I_\ell$ corresponding to $S'$.

Add the short jobs to $S$, using LPT.

## Approximation ratio:

When $B_k$ terminates the while loop,

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

Each job $j$ in $I_\ell$ has $p_j > \frac{T}{k}$.
Since $\frac{T}{k}$ is a multiple of $\frac{T}{k^2}$, each job $j$ in $I_\ell'$ has $p_j \geq \frac{T}{k}$.
Thus, $S'$ has at most $k$ jobs on each machine.
Hence,

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

each of the at most $k$ jobs on a machine is rounded down by less than $T/k^2$.

Thus, if the last job $\ell$ to finish belongs to $I_\ell$
$B_k(I) = \text{makespan}(S) < \left(1+\frac{1}{k}\right) OPT(I)$.

Otherwise, $p_\ell \leq \frac{T}{k} \leq \frac{OPT}{k}$

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

By the same arguments as in the proof that LPT is a 4/3-approx. alg.:


since job $\ell$ is a short job

Thus, in both cases,

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

## Running time:

Dyn. prg. as for bin packing:
$\leq k$ jobs on each machine
$\leq k^2$ different job sizes.
Hence, the configuration of a machine can be represented by a vector $(s_1, s_2, \dots, s_{k^2})$, where $0 \leq s_i \leq k$.
$\leq (k+1)^{k^2}$ possible conf.

$$OPT(n_1, n_2, \dots, n_{k^2}) = 1 + \min_{\vec{s} \in \mathcal{C}} \{ n_1 - s_1, \dots, n_{k^2} - s_{k^2} \}$$

set of possible conf.

Dyn. prg. table:

$\begin{cases} k^2 \text{ dimensions (one for each rounded job size)} \\ k+1 \text{ entries in each dimension} \end{cases}$

$\Downarrow$

$k^2(k+1) = O(k^3)$ entries in the table.

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

Total time: $O(k^{k^2+3})$.

# iterations of while loop $\leq \log U \leq \log P$

Total time: $O(k^{k^2+3} \log P)$

**Theorem 3.7 : $B_k$ is a PTAS**

*Proof*: $B_k$ achieves an approx. ratio of $1+\varepsilon$ with running time $O\left(\left(\frac{1}{\varepsilon}\right)^{\left(\frac{1}{\varepsilon}\right)^2+3} \cdot \log P\right)$.

If $\varepsilon$ is a constant, this is poly. in the input size, since it takes at least $\log P$ bits to write the processing time in binary. $\square$

Note that we did not expect a FPTAS:

The problem is strongly NP-hard, meaning that even if $p_{max} \le q(n)$, for some polynomium $q$, the problem is still NP-hard.

This implies that $\nexists$ FPTAS, unless $P = NP$:

Assume to the contrary that $\exists$ FPTAS $A_k$ with relative error $\frac{1}{k}$.

Then, with $k = \lceil 2nq(n) \rceil$,

$$
\begin{aligned}
A_k &\le \left\lfloor \left(1+\frac{1}{k}\right) OPT \right\rfloor, \quad \lfloor\ \rfloor \text{ since proc. times are integers} \\
&= \left\lfloor OPT + \frac{OPT}{k} \right\rfloor \\
&\le \left\lfloor OPT + \frac{1}{2} \right\rfloor, \quad \text{since } OPT \le n p_{max} \le n q(n) \\
&= OPT
\end{aligned}
$$

Thus, with $k = \lceil 2nq(n) \rceil$, $A_k$ produces an optimal schedule in time poly. in $n$ and $q(n)$, i.e., poly. in $n$.