Makespan minimization

Largest Processing time rule (LPTR) algorithm

Input: m, n jobs $a_1, \ldots, a_n$.

Algorithm: (1) Consider the jobs in non-increasing order of processing times. Assume that the order is $a^{(1)}, a^{(2)}, \ldots, a^{(n)}$.

(2) In the i-th iteration, assign $a^{(i)}$ to the least loaded machine.

Theorem 1: LPTR is a $\frac{4}{3}$-approximation algorithm.

Proof: Let j be the machine of maximum makespan and let i be the last job that is assigned to it. Just before i is assigned to it, its makespan is at most $\sum_{k=1}^{n} a_k / m \leq OPT$.

Case 1: $a_i \leq \frac{OPT}{3}$. Then $M(j) \leq OPT + \frac{OPT}{3}$
$= \frac{4}{3} \cdot OPT$

Case 2: $a_i > \frac{OPT}{3}$.

Claim: In this case LPTR computes an optimal schedule.

proof: $a^{(1)} \geq a^{(2)} \geq \cdots\cdots \geq a^{(n)}$. Towards a contradiction, assume that LPTR does not compute an optimal schedule. Let $a^{(k)}$ be the first job that finishes after OPT. $k \leq i$ (since i finishes after OPT). Therefore, $a^{(k)} \geq a^{(i)} > \frac{OPT}{3}$. Just before a_k is processed, each machine contains one or two jobs. (No machine has 0 job because then the k-th job can be assigned to that machine and made to finish before OPT. No machine can have three or more jobs, as then one of those jobs will finish after OPT.) Let machines 1 to l contain one jobs (long jobs) and machines l to m contain two jobs (short jobs (including k)) after iteration $k-1$.

$$k-1 = l + 2(m-l).$$

Consider an optimal schedule. I claim that each long job is the only jobs in its machine. Let t be the processing time of a long job. $t + a_k > OPT$. (otherwise k would finish before OPT in the schedule computed by LPTR). This implies that $t + a_j > OPT$ for all $j = 1, \cdots, k$. Thus t cannot be paired with any other job in an optimal schedule. Also no machine can have more than two jobs from $\{1, \cdots, k\}$. The total number of jobs must be at most $l + (m-l) \cdot 2 = k-1 < k$. This is a contradiction.

Theorem 1 is tight; there is an instance on which LPTR produces a solution with makespan $\simeq \frac{4}{3} \cdot OPT$.

A PTAS for makespan minimization: Makespan minimization is strongly NP-hard. So, an FPTAS does not exist (unless $P = NP$). We will obtain a PTAS for it.

Let $\overbrace{a_1, \ldots, a_n}^{\bar{a}}$, m be an input to makespan minimization. Let $BIN(\bar{a}, t)$ be the minimum number of bins of capacity t that are needed to pack $a_1, \ldots, a_n$. Then,

$$OPT(\bar{a}; m) = \min_{t \in \mathbb{R}^{\geq 0}} \{ t : BIN(\bar{a}, t) \leq m \}. \quad \text{①}$$

Let s be the makespan in some schedule.

Step 1: An efficient Bin-Packing algorithm for instances having at most k distinct sizes.

Input: A Bin-Packing instance where there are n_1 items of size s_1, n_2 items of size $s_2, \ldots, n_k$ items of size s_k. $n_1 + n_2 + \cdots + n_k = n$. Capacity of each bin is t.

Let $$Q = \{ (q_1, \ldots, q_k) : 0 \leq q_i \leq n_i, \ 1 \leq i \leq k, \ BIN((q_1, \ldots, q_k), t) = 1 \}$$

$$|Q| \leq (n_1 + 1) \cdot (n_2 + \cdots) \cdot \cdots (n_k + 1) = O(n^k).$$

$T(q_1, \dots, q_k) := 1$ if $(q_1, \dots, q_k) \in Q$.

$$BIN((i_1, \dots, i_k), t) = 1 + \min_{\substack{(q_1, \dots, q_k) \\ \in Q}} BIN((i_1 - q_1, \dots, i_k - q_k), t).$$

$O(n^{2k})$ time.

Step 2:

$$OPT \geq \begin{cases} \max_{i=1}^{n} a_i \\ \frac{1}{m} \cdot \sum_{i=1}^{n} a_i \end{cases}$$

Input: $a_1, \dots, a_n, m =: \overline{a}$

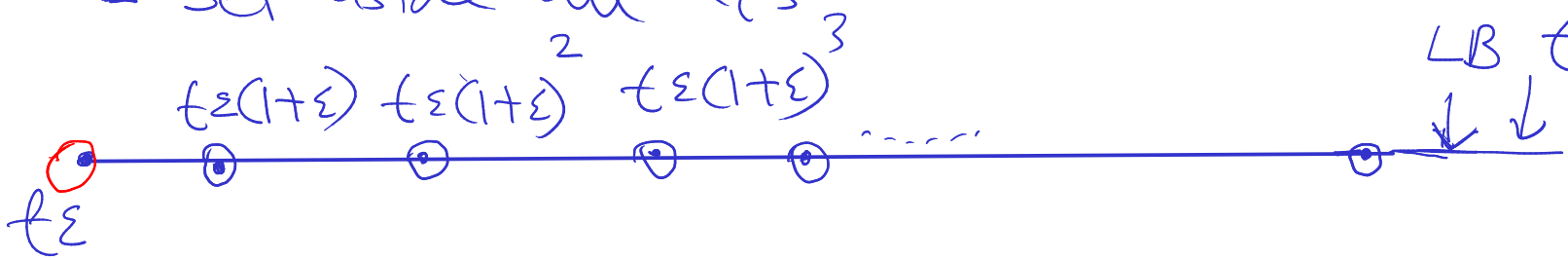
$$LB := \max \left\{ \max_{i=1}^{n} a_i, \frac{1}{m} \sum_{i=1}^{n} a_i \right\}$$

The greedy algo returns a solution with cost $\leq 2 \cdot LB$.

$\Rightarrow$ $\boxed{LB \leq OPT \leq 2 \cdot LB}$

Let $t \in [LB, 2 \cdot LB]$.

- Set aside all a_i's that are less than $t\varepsilon$.



Let there be u circled points.

$$\Rightarrow t\varepsilon(1+\varepsilon)^u \le t \quad \& \quad t\varepsilon(1+\varepsilon)^{u+1} \ge t.$$

$$u \le \frac{1}{\varepsilon} \Rightarrow u \le \log_{1+\varepsilon}\frac{1}{\varepsilon}.$$

$$(1+\varepsilon)^{\ell}$$

Let $t\varepsilon(1+\varepsilon)^{\ell} \le a_i < t\varepsilon(1+\varepsilon)^{\ell+1}$

$$a'_i := t\varepsilon(1+\varepsilon)^{\ell}.$$

$a' = (a'_1, \ldots, a'_{n'})$. Number of distinct item sizes is at most $\log_{1+\varepsilon}\frac{1}{\varepsilon}$. Pack a' optimally in bins of size t.

[(*) Now, pack the smaller items greedily; open new bins if necessary. Let the number of bins used be $\alpha(a,t,\varepsilon)$]

Imagine replacing each a'_i by a_i and the capacity of each bin is $t(1+\varepsilon)$ (*) Now, its a valid packing of $a_1, \ldots, a_n$ in bins of size $t(1+\varepsilon)$.

Claim 1: $\alpha(a,t,\varepsilon) \le BIN(a,t)$

Proof: Case 1: No new bin opened to pack the smaller items.

$\alpha(a,t,\varepsilon)$ = optimal no. of bins needed to pack items $a'_1,\dots,a'_{n'}$ in bins of size t.

$\leq BIN(a,t)$

Case 2: All but possibly one bin are full upto the capacity of t, and all bins are non-empty.

$$\therefore \sum_{i=1}^{n} a_i > (\alpha(a,t,\varepsilon)-1)\cdot t$$

$\Rightarrow$ To pack $(a_1,\dots,a_n)$ using bins of capacity t, at least $\alpha(a,t,\varepsilon)$ bins are needed

$\Rightarrow BIN(a,t) \geq \alpha(a,t,\varepsilon)$. ▨

Above, we designed an algorithm that has the following input-output behavior.

Input: ① $a_1, \overset{a}{\dots}, a_n$. This defines LB.

② $t \in [LB, 2LB]$

③ ε

Output: A packing of $a_1, \dots, a_n$ using at most $BIN(a,t)$ many bins of capacity $t(1+\varepsilon)$.

Step 3: Recall

$$OPT(\overline{a}, m) = \min_{t \in \mathbb{R}^+} \{t : BIN(a,t) \le m\}.$$

$t \in [LB, 2 \cdot LB]$.

Idea: Do binary search in the interval $[LB, 2 \cdot LB]$.

Problems: ① $BIN(a,t)$ cannot be computed efficiently & exactly.

② (If $a_1, \dots, a_n$ are not integers) convergence of the binary search.

Final algorithm $\mathcal{A}$:

- Do binary search on $[LB, 2 \cdot LB]$ to

find the minimum t s.t. $\alpha(a,t,\varepsilon) \le m$.

- Stop when the length of the current interval becomes at most $\varepsilon \cdot LB$. Let the current interval by $[c,d]$.
- Pass (a,d,ε) to the algorithm in step 2 and output the computed schedule.

$$d \ge \min_{t \in [LB, 2\cdot LB]} \{t : \alpha(a,t,\varepsilon) \le m\} =: t'$$

$$\alpha(a,d,\varepsilon) \le \alpha(a,t',\varepsilon) \quad \text{(Claim 1)}$$

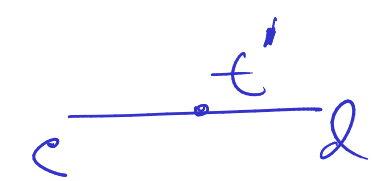
(check).

$$\le m.$$

$\Rightarrow$ Algorithm $\mathcal{A}$ computes a valid schedule.

Makespan of the computed solution

$$\le d(1+\varepsilon).$$



$t' \in [c,d]$: $d - c \le \varepsilon \cdot LB$

$\Rightarrow d - t' \le \varepsilon \cdot LB$

$\Rightarrow d \le t' + \varepsilon \cdot LB \le OPT \cdot (1+\varepsilon)$.

[$t' \le OPT$ (SEE LATER) (*)]

$\therefore$ Makespan of the computed solution is at most

$$d(1+\varepsilon) \leq OPT(1+\varepsilon)^2$$
$$= OPT(1+2\varepsilon+\varepsilon^2)$$
$$\leq OPT(1+3\varepsilon) \quad [\because \varepsilon \leq 1].$$

(*) $$t' = \min_t \{t : \alpha(a,t,\varepsilon) \leq m\}$$
$$\leq \min_t \{t : BIN(a,t) \leq m\}$$
$$= OPT.$$

To get a sol^n of cost $\leq OPT \cdot (1+\varepsilon)$, replace ε by $\frac{\varepsilon}{3}$ in the algorithm.