Homework

Exercise 1.

Overflowed backpack of minimum cost

Given n items of non-negative sizes $t_1, \ldots, t_n$, of costs $c_1, \ldots, c_n$, and a constant W, the *minimum cost overflowed backpack problem* is to find a set $S \subseteq \{1, \ldots, n\}$ of minimum cost, such that $\sum_{i \in S} t_i \geq W$.

Question 1. *By reduction from the partition problem (which is NP-hard), show that this problem is NP-hard.*

We study the following greedy algorithm :

Algorithm 1 Greedy

1. Sort the items by non-decreasing c_i/t_i : $c_1/t_1 \leq c_2/t_2, \ldots \leq c_n/t_n$.
2. Let i be the smallest index such that $\sum_{j \leq i} t_j \geq W$ and return $S = \{1, \ldots, i\}$.

Question 2. *Show that this algorithm may return arbitrary bad solutions, even if every item has size $< W$.*

We consider two arrays C and V :
– $C[i, v]$ is the minimum feasible cost with a total volume exactly v obtained with a subset of the i first items.
– $V[i, c]$ is the maximum feasible volume with a total cost of exactly c obtained with a subset of the i first items.

Question 3. *Give a recursive definition of V and C and propose two pseudo-polynomial algorithms for the minimum overflowed backpack problem.*

Let $C = \max_i c_i$.

Question 4. *Propose a polynomial algorithm in time $poly(n, 1/\epsilon)$ such that the cost of the returned solution is at most* $\mathrm{OPT} + \epsilon C$.

Hint : *Round the instance in the following way : let $K = \frac{\epsilon C}{n}$ and for each item, round its cost to $c'_i = \lceil \frac{c_i}{K} \rceil$.*

Question 5. *By parametric pruning, deduce a fully polynomial time approximation scheme for this problem.*

Hint : *What can you say about the previous algorithm when $C \leq \mathrm{OPT}$?*

Exercise 2.

Minimum makespan scheduling

Given n jobs $J_1, \ldots, J_n$ of processing times $p_1, \ldots, p_n$, and m identical machines, a schedule of the jobs is an assignment (m_i, t_i) of the jobs J_i to the machine m_i such that $\forall i, j$, if $m_i = m_j$, then $t_j \notin [t_i, t_i + p_i)$. We say that J_i is scheduled at time t_i on machine m_i. The completion time of J_i is $CT_i = t_i + p_i$. The makespan of a schedule is the maximum completion time of the jobs, i.e. $\max_i CT_i$.

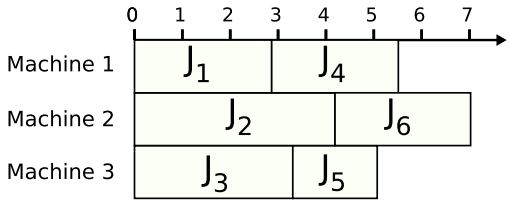

FIG. 1 – Example : A schedule of 6 jobs on 3 machines. Job 6 ends at time 7 and thus the makespan of this schedule is 7.

The minimum makespan scheduling problem is to find a schedule of n jobs on m machines, so that the makespan is minimized.

Question 1. *Show that this problem is NP-hard.*
Hint : *Reduce the partition problem to it.*

Question 2. *Show that the average time a machine has to run,* $\frac{1}{m}(\sum_i p_i)$*, and the largest processing time are two lower bounds on* OPT.

We consider the following greedy algorithm :

Algorithm 2 Greedy

1. Order the jobs arbitrarily.
2. Schedule the jobs on machines in this order, scheduling the next job on the machine that has been assigned the least amount of work so far. Figure is an example of the execution of this algorithm.

Question 3. *Show that the greedy algorithm is a* $(2 - 1/m)$*-approximation algorithm and that it is not better.*
Hint : *Let MS denote the makespan of the greedy algorithm and* p_n *the size of the last scheduled job. What can you say about the total amount of work assigned to each machine at time* $MS - p_n$ *?*

For an integer k, let $J_1, \ldots, J_k$ be the k longest jobs of the n jobs. We consider the following algorithm (which is not necessarily polynomial) :

Algorithm 3

1. Schedule $J_1, \ldots, J_k$ optimally on the m machines.
2. Order the remaining jobs $J_{k+1}, \ldots, J_n$ arbitrarily.
3. Schedule this remaining jobs by placing each of them iteratively on the machine that has been assigned the least amount of work so far.

Let $\omega(k)$ be the makespan of the solution returned by this algorithm.
We aim to show the following :

$$\omega(k) \leq (1 + \frac{1 - 1/m}{1 + \lceil k/m \rceil}) \, \mathrm{OPT} \, .$$

Let p^* be the longest processing time of the remaining jobs $J_{k+1}, \ldots, J_n$ after step 1, i.e. $p^* = \max_{k<i\leq n} p_i$.

Question 4. *Show that* $\mathrm{OPT} \geq \omega(k) - \left(\frac{m-1}{m}\right) p^*$.
Hint : *Recall Question 3*

Question 5. *Show that* $\mathrm{OPT} \geq (1 + \lceil k/m \rceil) \, p^*$ *by remarking that there is at least* $(k + 1)$ *jobs of size greater than* p^*.

Question 6. *Conclude that* $\omega(k) \leq \left(1 + \frac{1 - 1/m}{1 + \lceil k/m \rceil}\right) \mathrm{OPT}$.

We now consider a modification of the greedy algorithm, where instead of taking the jobs in an arbitrary order, the algorithm considers it by non-increasing processing time :

Algorithm 4 Greedy decreasing

1. Order the jobs by non-increasing processing times.
2. Schedule the jobs on machines in this order, scheduling the next job on the machine that has been assigned the least amount of work so far.

Let p_{min} be the smallest processing time of any job.

Question 7. *We first suppose that* $p_{min} > \frac{1}{3}\,\mathrm{OPT}$. *Show that if* $n \leq 2m$ Greedy decreasing *returns an optimal solution.*

Hint : *Remark that in an optimum solution, when* $p_{min} > \frac{1}{3}\,\mathrm{OPT}$, *there is at most* 2 *jobs per machine, and show that* OPT *can be modified into the solution returned by the Greedy decreasing algorithm without increasing the makespan.*

Question 8. *Show that the Greedy decreasing algorithm is a* $4/3$*-approximation algorithm.*

Hint : *If* $p_{min} > \frac{1}{3}\,\mathrm{OPT}$, *remark that Greedy decreasing is a special case of algorithm* 3 *and use Question* 6. *If* $p_{min} \leq \frac{1}{3}\,\mathrm{OPT}$, *use the analysis in Question* 3.