高等演算法 HW1

許博翔

March 21, 2024

Problem 1. Let OPT be the optimal solution, and $val(OPT)$ be its value.

If $c_1 + (n-1)c_m > B$, then no worker can select the first item, and we can remove all $p_{11}, p_{21}, \ldots, p_{n1}$, so let's suppose that $c_1 + (n-1)c_m \le B$.

Let $b = \frac{p_{11}\epsilon}{2n}$, and $p'_{ij} := \lceil \frac{p_{ij}}{b} \rceil b$, which we'll call it "new productivity".

Let $q_{ij} := \frac{p'_{ij}}{b} = \lceil \frac{p_{ij}}{b} \rceil$.

Let dp_{ij} := the minimum cost that can achieved with new productivity jb by $W_1, W_2, \ldots, W_i$, and r_{ij} the item that W_i should select to achieve such minimum cost. The range: $1 \le i \le n,\ 0 \le j \le Q$, $Q := \sum_{k=1}^{n} q_{k1}$.

The base case $i = 1$:

$$dp_{1j} = \begin{cases} \min\limits_{k: q_{1k}=j} (c_k), \text{ if } \exists k \text{ s.t. } q_{1k} = j \\ B + 1, \text{ otherwise} \end{cases}$$

$$r_{1j} = \begin{cases} k, \text{ where } c_k = dp_{1j} \text{ and } q_{1k} = j, \text{ if such } k \text{ exists} \\ -1, \text{ otherwise} \end{cases}$$

One can run from $i = 2$ to n, from $j = 0$ to Q to get the values of dp_{ij} using

$$dp_{ij} = \begin{cases} \min\limits_{k: q_{ik} \le j} (dp_{i-1, j-q_{ik}} + c_k), \text{ if } \exists k \text{ s.t. } q_{ik} \le j \\ B + 1, \text{ otherwise} \end{cases}$$

$$r_{ij} = \begin{cases} k, \text{ where } dp_{i-1, j-q_{ik}} + c_k = dp_{ij} \text{ and } q_{ik} \le j, \text{ if such } k \text{ exists} \\ -1, \text{ otherwise} \end{cases}$$

Denote the optimal solution as ALG, and the value of ALG (denote as $val'(ALG)$ is the maximum new productivity that can be achieved with cost at most B, which is $\max\limits_{j:dp_{nj}\le B}(jb)$, and we can recursively find the selected item that can achieve this using r_{ij}.

The above can be done in $O(nQm)$ time complexity.

Let j_i denote the selected item by W_i in ALG, and let $\sum\limits_{i=1}^{n} p_{ij_i}$ be the productivity value of this algorithm (denoted as $val(ALG)$).

Let k_i denote the selected item by W_i in OPT, and let the new productivity value of these selected item be $val'(OPT)$.

By the definition of OPT, $val(ALG) \le val(OPT)$.

Since the above dp algorithm obtains optimal solution of new productivity, $val'(ALG) \ge val'(OPT)$.

$val(ALG) = \sum\limits_{i=1}^{n} p_{ij_i} > \sum\limits_{i=1}^{n}(\lceil \frac{p_{ij_i}}{b} \rceil - 1)b = val'(ALG) - nb \ge val'(OPT) - nb = \sum\limits_{i=1}^{n} \lceil \frac{p_{ik_i}}{b} \rceil b - nb \ge \sum\limits_{i=1}^{n} p_{ik_i} - nb = val(OPT) - nb = val(OPT) - \frac{p_{11}\epsilon}{2}$.

By what we suppose in the first three lines, $p_{11} \le p_{11} + p_{2m} + p_{3m} + \cdots + p_{nm} \le val(OPT)$ (since W_1 can select 1, while $W_2, W_3, \ldots, W_n$ select m).

$\Rightarrow val(ALG) \ge val(OPT) - \dfrac{val(OPT)\epsilon}{2} = (1 - \frac{\epsilon}{2})val(OPT)$.

$\Rightarrow val(ALG) \le val(OPT) \le \dfrac{val(ALG)}{1-\frac{\epsilon}{2}} \le (1+\epsilon)val(ALG)$.

The time complexity of this algorithm $= O(nQm) = O(n\sum\limits_{k=1}^{n} q_{k1}m) = O(n\sum\limits_{k=1}^{n}\lceil \frac{p_{k1}}{b} \rceil m) = O(n\sum\limits_{k=1}^{n}\lceil \frac{2np_{k1}}{p_{11}\epsilon} \rceil m) = O(n\sum\limits_{k=1}^{n} \frac{2n}{\epsilon} m) = O(\frac{n^3 m}{\epsilon})$

Problem 2. Let OPT be the optimal solution, and $val(OPT)$ be its value.

Let $p_j := p_{1j} = p_{2j} = \cdots = p_{nj}$.

First, there is a 4-approximation.

Let $k = \min\limits_{i:p_{p_{4i+1}+p_{4i+2}+p_{4i+3}+p_{4i+4}<P}}(i)$.

That is, for $i = 1, 2, \ldots, k$, $p_{4i+1} + p_{4i+2} + p_{4i+3} + p_{4i+4} \ge P$.

$\Rightarrow k \le val(OPT)$.

Since $p_{4k+1} \ge p_{4k+2} \ge \cdots \ge p_m$, for all 4 distinct elements a, b, c, d of the multiset $\{p_{4k+1}, p_{4k+2}, \ldots, p_m\}$, $a + b + c + d \le p_{4i+1} + p_{4i+2} + p_{4i+3} + p_{4i+4} < P$.

$\Rightarrow$ a worker with productivity at least P should take at least one of the $1, 2, \ldots, 4k$-th machine.

$\Rightarrow val(OPT) \leq 4k$.

$\therefore k \leq val(OPT) \leq 4k$.

Since in the OPT solution, one would use at most $16k$ machines, and using the machines with larger productivity will not decrease the number of workers with productivity at least P.

$\therefore$ set $M := \min(16k, m)$, and there is an OPT solution s.t. only the i-th ($1 \leq i \leq M$) machine will be used.

Let $a := \lfloor \frac{M\epsilon}{32} \rfloor$, and $b := \lceil \frac{M}{a} \rceil$.

Let $q_i := p_{M+1-i}$. (That is, q is p's reverse, which is increasing.)

Partition $\{q_1, q_2, \ldots, q_M\}$ into $S_1, S_2, \ldots, S_b$, where $S_i := \{q_j : a(i-1)+1 \leq j \leq ai\}$.

That is, the i-th machine is of the $\lceil \frac{i}{a} \rceil$-th type, and define the new productivity of the machines of the j-th type as $f(S_j)$.

There are at most $c := \left(\!\!\binom{b}{4}\!\!\right) + \left(\!\!\binom{b}{3}\!\!\right) + \left(\!\!\binom{b}{2}\!\!\right) + \left(\!\!\binom{b}{1}\!\!\right) + \left(\!\!\binom{b}{0}\!\!\right)$ ways to select the types of at most 4 different machines.

There are n identical workers in total, and c different ways to select the types of the machines they take.

$\Rightarrow$ there are at most $\left(\!\!\binom{c}{n}\!\!\right)$ possibilities.

Bruteforce through all (at most) $\left(\!\!\binom{c}{n}\!\!\right)$ possibilities, for each possibility, check if the i-th type of machine is used by at most $|S_i|$ workers for $i = 1, 2, \ldots, b$, and then calculate the number of workers with new productivity ≥ 4. The value of this algorithm with new productivity function f (denote as $val(f)$) is the maximum number of workers with new productivity ≥ 4. The complexity of this part is $O(\left(\!\!\binom{c}{n}\!\!\right) \times (b+n)) = O(n^{c+1})$.

Define f_1: $f_1(S_i) := \begin{cases} \min(S_i), \text{ if } i \geq 2 \\ 0, \text{ if } i = 1 \end{cases}$.

Define f_2: $f_2(S_i) := \begin{cases} \min(S_{i+1}), \text{ if } i \leq b-1 \\ \max(P, q_M), \text{ if } i = b \end{cases}$.

Since the difference of the new productivities using f_1, f_2 are a 0s, and $a \max(P, q_M)$, and the a worker taking only $\max(P, q_M)$ have new productivities $\geq P$.

$\therefore val(f_2) \leq val(f_1) + a$.

Also, the new productivity of the i-th machine in f_1 is not larger than the original productivity, and in f_2 is not smaller than the original productivity.
$\therefore val(f_1) \leq val(OPT) \leq val(f_2) \leq val(f_1)+a = val(f_1)+\lfloor\frac{M\epsilon}{32}\rfloor \leq val(f_1)+\frac{M\epsilon}{32} \leq val(f_1)+\frac{16k\epsilon}{32} \leq val(f_1)+\frac{val(OPT)\epsilon}{2}$.
$\Rightarrow (1-\frac{\epsilon}{2})val(OPT) \leq val(f_1)$.
$\Rightarrow val(f_1) \leq val(OPT) \leq \frac{val(f_1)}{1-\frac{\epsilon}{2}} \leq (1+\epsilon)val(f_1)$.
$b = \lceil\frac{M}{a}\rceil < \frac{M}{a}+1 = \frac{M}{\lfloor\frac{M\epsilon}{32}\rfloor} < \frac{M}{\frac{M\epsilon}{32}-1} = \frac{1}{\frac{\epsilon}{32}-\frac{1}{M}} < \frac{1}{\frac{\epsilon}{32}-\frac{\epsilon}{64}} = \frac{64}{\epsilon} = O(1)$.
$c = \left(\!\binom{b}{4}\!\right) + \left(\!\binom{b}{3}\!\right) + \left(\!\binom{b}{2}\!\right) + \left(\!\binom{b}{1}\!\right) + \left(\!\binom{b}{0}\!\right) = O(1)$.
$\therefore$ the time complexity is $O(n^{c+1})$, which is a polynomial of n.

Problem 3.

Problem 4. Let w_v be the weight of the vertex $v \in V(G)$.
Transform the vertex cover problem to an ILP problem (like that taught in class):
Variables: $\{x_v : v \in V(G)\}$.
$\min \sum_{v\in V} w_v x_v$.
Subject to:
$x_v \in \{0,1\},\ \forall v \in V(G)$, where $x_v = 1$ iff the vertex cover contains v.
$x_u + x_v \geq 1,\ \forall uv \in E(G)$.
Relax the above to LP (that is, relax the condition $x_v \in \{0,1\}$ to $0 \leq x_v \leq 1$), then we can solve it in polynomial time complexity, and suppose the solution is $x_v = y_v^*$.
Let $I \subseteq V(G)$ be an independent set.
One can see that $y_v^{(I)} := \begin{cases} 0, \text{ if } y_v^* < \frac{1}{2} \text{ or } (x_v = \frac{1}{2} \text{ and } v \in I) \\ 1, \text{ otherwise} \end{cases}$ satisfy that $y_v^{(I)} \in \{0,1\},\ \forall v \in V(G)$.
Since $y_u^* + y_v^* \geq 1$, WLOG suppose that $y_u^* \geq y_v^*$, there is $y_u^* \geq \frac{1}{2}y_u^* + y_v^* \geq \frac{1}{2}$.
If $y_u^* > \frac{1}{2}$ or ($y_u^* = \frac{1}{2}$ and $u \notin I$), then $y_u^{(I)} = 1$.
Otherwise, $y_u^* = \frac{1}{2}$ and $u \in I$.
$\Rightarrow y_v^* \geq 1 - y_u^* = \frac{1}{2}$.
Since I is an independent set and $uv \in E$ and $u \in I$, there must be $v \notin I$.

$\Rightarrow y_v^{(I)} = 1$.

$\therefore$ at least one of $y_u^{(I)}, y_v^{(I)} = 1$.

$\Rightarrow$ the condition "$y_u^{(I)} + x_v^{(I)} \geq 1, \ \forall uv \in E(G)$" is satisfied.

In class, we learn that the solution to this LP problem satisfies $\forall v \in V(G), \ y_v^* \in \{0, \frac{1}{2}, 1\}$.

Since G has a k-coloring, one can partition $V(G)$ into k independent sets $I_1, I_2, \ldots, I_k$.

If $y_v^* = \frac{1}{2}$, then $\sum\limits_{v \in V(G)} y_v^{(I_i)} = 1(k-1) + 0 = k - 1 = (2k-2)y_v^*$.

If $y_v^* = 0$ or 1, then $\sum\limits_{v \in V(G)} y_v^{(I_i)} = ky_v^* \leq (2k-2)y_v^*$.

$\therefore \sum\limits_{i=1}^{k} \sum\limits_{v \in V(G)} y_v^{(I_i)} \leq \sum\limits_{v \in V(G)} (2k-2)y_v^*$.

By pigeonhole principle, $\exists i$ s.t. $\sum\limits_{v \in V(G)} y_v^{(I_i)} \leq (2 - \frac{2}{k}) \sum\limits_{v \in V} y_v^*$.

Let $val(OPT)$ be the value of the ILP.

Since LP relaxes some condition of the ILP, $\sum\limits_{v \in V} y_v^* \leq val(OPT)$.

$\Rightarrow \sum\limits_{v \in V(G)} y_v^{(I_i)} \leq (2 - \frac{2}{k}) \sum\limits_{v \in V} y_v^* \leq (2 - \frac{2}{k}) val(OPT)$.

The time complexity is polynomial since:

1. The time complexity creating and solving the LP problem is polynomial.

2. The time complexity running through all $i = 1$ to k, finding the i such that $\sum\limits_{v \in V(G)} y_v^{(I_i)} \leq (2 - \frac{2}{k}) \sum\limits_{v \in V(G)} y_v^*$ is $O(kV(G)) \overset{k \leq V(G)}{\leq} O(V(G)^2)$.