Advanced Algorithms 2024 2024

Sample Solutions GHW 01

Lecturer: Johannes Lengler *Teaching Assistant: Antti, Richard*

1 Covering Cliques

Richard Hladík

1. Let us abbreviate $E_i := \binom{V_i}{2} \cap E$. The integer LP has a binary variable for every edge $e \in E$ saying whether the edge is included in F. We have a constraint for every V_i, requiring that at least s edges in E_i are included in F. The criterion being minimized is the number of edges in F. Written formally, the integer LP is as follows:

$$\begin{aligned} \text{given } x_e \in \{0,1\} \quad & \forall e \in E \\ \text{minimise } \sum_{e \in E} x_e & \\ \text{subject to } \sum_{e \in E_i} x_e \geq s \quad & \forall V_i \in \mathcal{S} \end{aligned}$$

2. We create an LP relaxation of the integer LP above by changing the domain of all variables to $x_e \in [0,1]$. Note that the LP relaxation has a solution by setting all x_e to one, so let $\mathbf{x}^* \in [0,1]^E$ be the optimal solution of the LP relaxation. Define

$$y_e = \begin{cases} 1 & \text{if } x_e^* \geq \binom{k}{2}^{-1}; \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we take $F = \{\, e \in E \mid y_e = 1 \,\}$.

Now we prove that (a) $\mathbf{y}$ is a solution to the ILP and (b) it is a $\binom{k}{2}$-approximation.

(a) Take arbitrary $V_i \in \mathcal{S}$. We know that $\sum_{e \in E_i} x_e^* \geq s$ and want to prove that $\sum_{e \in E_i} y_e \geq s$. Assume this is not the case. Then this means that there are at most $s-1$ edges $e \in E_i$ such that $x_e^* \geq \binom{k}{2}^{-1}$. Denote the set of these edges as S. Now we can write

$$\sum_{e \in E_i} x_e^* = \sum_{e \in E_i \cap S} x_e^* + \sum_{e \in E_i \setminus S} x_e^* < s - 1 + \frac{\binom{k}{2} - s + 1}{\binom{k}{2}} < s - 1 + 1 = s,$$

which is a contradiction with x_e^* being a solution to the relaxed LP.

(b) By definition, we have $y_e \leq \binom{k}{2} x_e^*$. Denote by OPT^* the value of the relaxed LP optimum, and by OPT the value of the integral LP optimum. We have $\mathrm{OPT}^* \leq \mathrm{OPT}$, since the solution space of the integral LP is a subset of that of the relaxed LP. We can write:

$$|F| = \sum_{e \in E} y_e \leq \sum_{e \in E} \binom{k}{2} x_e^* = \binom{k}{2} \mathrm{OPT}^* \leq \binom{k}{2} \mathrm{OPT},$$

just as desired.

2 Graph Coloring

Johannes Lengler

As usual, we denote by n the number of vertices and m the number of edges. Let m_r, m_b and m_v denote the number of red, blue, and violet edges, respectively.

If $m_r \geq m/3$ or $m_b \geq m/3$, color everything red or blue respectively. This gives a score of at least $m/3$. Since OPT $\leq m$, this is a 3-approximation of OPT.

Otherwise, color everything randomly, and let S be the resulting score. We satisfy each violet edge with a probability of $1/2$ and each blue and red edge with a probability of $1/4$. Hence, $\mathbb{E}[S] = \frac{m_v}{2} + \frac{m_r+m_b}{4}$. Using $m_r + m_b = m - m_v$ and $m_v \geq m/3 + 1$, we obtain

$$\mathbb{E}[S] = \frac{m_v}{2} + \frac{m_r + m_b}{4} = \frac{2m_v + m - m_v}{4} = \frac{m_v + m}{4} \geq \frac{\frac{4}{3}m + 1}{4} \geq \frac{m+1}{3}.$$

Now let $S' := m - S$. Then $\mathbb{E}[S'] \leq \frac{2m-1}{3}$. By Markov's inequality, we have

$$\Pr\left[S' \geq \frac{2m}{3}\right] \leq \frac{(2m-1)/3}{2m/3} = 1 - \frac{1}{2m}.$$

Hence,

$$\Pr\left[S \geq \frac{m}{3}\right] = \Pr\left[S' \leq \frac{2m}{3}\right] \geq \Pr\left[S' < \frac{2m}{3}\right] \geq \frac{1}{2m}.$$

Now we repeat the coloring $k := \lceil 2m \ln m \rceil$ times. Then the probability that none of those yield $S \geq \frac{m}{3}$ is

$$\Pr\left[\text{no trial yields } S \geq \frac{m}{3}\right] \leq \left(1 - \frac{1}{2m}\right)^{2m \ln m} \leq (e^{-\frac{1}{2m}})^{2m \ln m} = \frac{1}{m}.$$

Therefore, w.h.p. at least one of the k trials gives a value of S of at least $m/3$, which is a 3-approximation.

3 Item Distribution

Antti Roeyskoe

1. For an easier description, consider just assigning sizes to vertices, with the restriction that any size may be assigned at most as many times as there are items with that size, and no restriction on number of times the size 0 is assigned.

   We perform dynamic programming on the tree. Fix an arbitrary root vertex, we then proceed from leaves to the root. For every vertex u, size $s_u \in S$ and counts $\{c_s\}_{s \in S}$ (c_s being at most the number of items with size s or n, whichever is smaller), we compute $\mathrm{DP}(u, s_u, \{c_s\}_{s \in S})$: the assignment of sizes to vertices in the subtree of u that, among all assignments of sizes to vertices in the subtree of u satisfying that

   - the size $s \in S$ is assigned to exactly c_s vertices in the subtree of u (including u itself), and
   - the root u of the subtree is assigned the size s_u,

   minimizes the sum of sizes assigned to children of u. If no such assignment exists, we store this information instead.

   We compute these assignments in a bottom-up manner, starting from the leaves, where $\mathrm{DP}(u, s_u, \{c_s\}_{s \in S})$ is an assignment of the size s_u to u if $c_{s_u} = 1$ and $c_{s'} = 0$ for all other $s' \in S$, and doesn't exist otherwise.

   Suppose that for vertex u, all DP-values for its children have been computed. Fix an arbitrary ordering of the children of u, and for k at most the number of children $|\mathrm{ch}(u)|$ of

u, define $\mathrm{DP}_k(u, s_u, \{c_s\}_{s\in S})$ as the DP-value you would obtain with the restriction that the size 0 must be assigned to every vertex in the subtrees of the $|\mathrm{ch}(u)| - k$ last children of u. We have $\mathrm{DP}_{|\mathrm{ch}(u)|}(u, s_u, \{c_s\}_{s\in S}) = \mathrm{DP}(u, s_u, \{c_s\}_{s\in S})$, and as with leaves, we have the base case where $\mathrm{DP}_0(u, s_u, \{c_s\}_{s\in S})$ is an assignment of the size s_u to u if $c_{s_u} = 1$ and $c_{s'} = 0$ for all other $s' \in S$, and doesn't exist otherwise.

Now, we can describe how to compute $\mathrm{DP}_{k+1}(u, \cdot, \cdot)$ from $\mathrm{DP}_k(u, \cdot, \cdot)$ and $\mathrm{DP}(u', \cdot, \cdot)$ where u' is the $(k + 1)$-th child of u. This can be done by simply iterating over $s_{u'}$ and all partitions of c_s's into $c_s^1 + c_s^2 = c_s$, and offering the assignment $\mathrm{DP}_k(u, s_u, \{c_s^1\}_{s\in S}) \cup \mathrm{DP}(u', s_{u'}, \{c_s^2\}_{s\in S})$ if both of these individual assignments exist and the sum of sizes of children of u' in the assignment is at most $h - s_u$. For every triplet $(u, s_u, \{c_s\}_{s\in S})$, there are at most $C^2 n^{2C}$ such offered assignments, so we can iterate over all of them and take the one minimizing the sum of sizes assigned to children of u.

Correctness follows by correctness of the two combined DP tables, as the sum of sizes of the children of u after combining the assignments is exactly $s_{v'}$ plus the sum of sizes of the first k children of u (which is minimized by $\mathrm{DP}_k(u, \cdot, \cdot)$), and the combined assignment is valid if the sum of sizes of children of u' (which is minimized by $\mathrm{DP}(u', \cdot, \cdot)$) is at most $h - s_u$. Thus, for any assignment satisfying the requirements, the offered assignment with the same split $c_s^1 + c_s^2 = c_s$ exists and has sum of children of u at most that of the assignment.

Finally, for the root vertex r, we can iterate over all s_r and $\{c_s\}_{s\in S}$, and out of those where the sum of sizes of children of r is at most h, return the one with maximum $\sum c_s$.

2. Fix the desired constant $\epsilon > 0$. We may assume $\epsilon \leq 1$. Let $C = \lfloor 1/\epsilon \rfloor \geq 1$. To obtain a PTAS, we round up the sizes of items so that at most C sizes remain, then apply part 1.

   To describe the rounding, suppose k is the optimal number of items assigned (we can iterate over k as it is at most n). First, note that if there is a way to assign some number k of items, it is possible to assign the k items with minimum sizes. Thus, if $k \leq C$, we can delete all but the k items with minimum sizes and apply part 1, obtaining an exact solution. Otherwise, let $k' := \lceil k/C \rceil$, and let I_j be the range of the $(jk' + 1)$-th to $(j + 1)k'-$th item in increasing order of size. For every valid j, replace the size of every item in I_j with the minimum size of an item in I_{j+1}. Then, delete every item except the first k, and finally, apply part 1.

   With this approach, there are at most C distinct item sizes as $k/k' \leq C$, and since the sizes of items only increased, the returned assignment of items remains valid for the original sizes. It remains to show that after the operation, there still exists a sufficiently large assignment of items, and indeed, one with at least $k - k'$ items exists, as after the rounding, the i-th item has size at most that of the $(i + k')$-th item's original size. Assigning the first $k - k'$ items to the same vertices the $(k' + 1)$-th to k-th items are assigned in the optimal solution is thus valid even with rounded sizes.

   Thus, the algorithm is a $(1 - \epsilon)$-approximation, as from $C \leq \frac{1}{\epsilon}$ it follows that $k - k' = k - \lceil k/C \rceil \leq k - k/C \leq k - k\epsilon = (1 - \epsilon)k$.

4 Rental Problem

Richard Hladík

1. Let $\sigma_1, \ldots, \sigma_n$ be the sequence of winter sports days, with $\sigma_i = \textit{ski}$ or $\sigma_i = \textit{snowboard}$. Fix any deterministic online algorithm A. We will construct an adversarial sequence σ as follows.

   Until A buys skis or the bundle deal, we let $\sigma_i = \textit{ski}$. Let k be the day on which A buys skis or the bundle deal. We then distinguish three cases:

- A **never buys skis or the bundle deal.** In this case, the competitive ratio is infinite, since for fixed X, Y, we can make the algorithm pay arbitrarily much while the offline algorithm pays a constant amount.
- A **buys the bundle deal on the k-th day.** Then we terminate σ after the k-th day, i.e., we set $n := k$. A has paid at least $Y + k - 1$. On the other hand, the optimal offline algorithm will either buy skis at the start, or rent skis for the whole time, whichever is cheaper. Hence, $c_{\mathrm{OPT}}(\sigma) = \min(X, k)$, and the competitive ratio satisfies
$$\alpha_A \geq \frac{c_A(\sigma)}{c_{\mathrm{OPT}}(\sigma)} \geq \frac{Y + k - 1}{\min(X, k)} \geq \frac{k}{k} + \frac{Y - 1}{X} = 1 + \frac{Y - 1}{X}.$$
- A **buys skis on the k-th day.** Then, we continue σ as follows: if A has bought a snowboard already, we terminate σ after the k-th day and set $k' = k$. Otherwise, until A buys either the snowboard or the bundle deal, we set $\sigma_i = \textit{snowboard}$, and we let k' to be the day on which A made the second purchase. We terminate σ after the k'-th day.

  In total, A has paid at least $2X + k' - 1$. On the other hand, an offline algorithm can buy the bundle deal at the start, or rent for the whole time, whichever is cheaper. Hence, $c_{\mathrm{OPT}}(\sigma) \leq \min(Y, k')$, and the competitive ratio satisfies
$$\alpha_A \geq \frac{c_A(\sigma)}{c_{\mathrm{OPT}}(\sigma)} \geq \frac{2X + k' - 1}{\min(Y, k')} \geq 1 + \frac{2X - 1}{Y},$$
  where for the last inequality, we used the same trick as in the previous part.

Since every algorithm falls into one of those three cases, this proves that for any valid choice of $1 \leq X \leq Y \leq 2X$ and for any algorithm, we have
$$\alpha_A \geq 1 + \min\left(\frac{Y - 1}{X}, \frac{2X - 1}{Y}\right) \geq 1 + \min\left(\frac{Y}{X}, \frac{2X}{Y}\right) - \frac{1}{X}.$$
As α_A is defined as the supremum over all valid X, Y, our goal is to pick them such that the above expression is maximized. For any fixed X, the first argument of min is increasing with Y and the other one is decreasing. Hence, for a fixed X, the expression is maximized when the two terms are equal, i.e. when
$$\frac{Y}{X} = \frac{2X}{Y} \iff Y^2 = 2X^2 \iff Y = \sqrt{2}X.$$
Thus, by setting $Y = \sqrt{2}X$, we get $\alpha_A \geq 1 + \sqrt{2} - 1/X$. For any $\varepsilon > 0$, we can make this greater than $1 + \sqrt{2} - \varepsilon$ by choosing X large enough.

2. Our algorithm will mimic the two cases in the lower bound. When Y/X is small, it intuitively pays off more to take the bundle deal, otherwise it pays off more to buy the two items separately. Therefore, in our algorithm, we will distinguish two cases:

   - $Y \leq \sqrt{2}X$. Then the algorithm always rents until the $\lfloor X \rfloor$-th day, on which it buys the bundle deal.
   - $Y > \sqrt{2}X$. The algorithm counts how many days of a given type have elapsed, and on the $\lfloor Y/2 \rfloor$-th day of that type, it buys the equipment of that type.

   To prove competitiveness, we will look at both cases separately.

- $Y \leq \sqrt{2}X$. Assume the season ended after n days. If $n < X$, the algorithm is in fact 1-competitive. If $n \geq X$, we paid $\lfloor X \rfloor + Y \leq X + Y$, while the optimal offline algorithm has paid at least X. Thus, in this case, we have

$$\frac{c_A(\sigma)}{c_{\mathrm{OPT}}(\sigma)} \leq \frac{X+Y}{X} = 1 + \frac{Y}{X} \leq 1 + \sqrt{2}.$$

- $Y > \sqrt{2}X$. Assume the season ended after A ski days and B snowboard days. We will distinguish four cases:
  - $A, B < Y/2$. Then the algorithm is in fact 1-competitive, as we can check that renting is the optimal solution (recall that $Y/2 \leq X$).
  - $A, B \geq Y/2$. The algorithm spent $2\lfloor Y/2 \rfloor + 2X \leq Y + 2X$. The optimal offline algorithm needs to spend $\min(Y, \min(A, X) + \min(B, X)) = Y$, and thus

$$\frac{c_A(\sigma)}{c_{\mathrm{OPT}}(\sigma)} \leq 1 + \frac{2X}{Y} \leq 1 + \sqrt{2}.$$

  - $A < Y/2, B \geq Y/2$. The algorithm spent $c_A(\sigma) = \lfloor Y/2 \rfloor + X + A \leq Y/2 + X + A$. The optimal offline algorithm needs to spend $c_{\mathrm{OPT}}(\sigma) = \min(Y, \min(A, X) + \min(B, X)) = \min(Y, A + \min(B, X)) \geq \min(Y, A + Y/2) = A + Y/2$. The competitive ratio can now be bounded as

$$\frac{c_A(\sigma)}{c_{\mathrm{OPT}}(\sigma)} \leq \frac{Y/2 + X + A}{A + Y/2} \leq 1 + \frac{X}{A + Y/2} \leq 1 + \frac{X}{Y/2} \leq 1 + \sqrt{2}.$$

  - $A \geq Y/2, B < Y/2$. Analogously to the previous case.

We have shown that the competitive ratio of both sub-algorithms is always at least $1+\sqrt{2}$, and the whole algorithm is thus $(1 + \sqrt{2})$-competitive, as desired.