Prof. Friedrich Eisenbrand
Dr. Yuri Faenza

Discussions from: October 22, 2013

Combinatorial Optimization

Fall 2013

Assignment Sheet 3

Exercises marked with a ⋆ can be handed in for bonus points. Due date is November 05.

Exercise 1

1. Left to the reader.
2. We will go through the proof that the algorithm provides a 2−approximation, and suitably modify it. Consider an optimal solution T^* to the Steiner tree problem, and double its edges. We obtain an Eulerian graph (i.e. a graph where all nodes have even degree) $\tilde{G}$ of cost $2c(T^*)$. We can now order the terminals as follows: starting from a terminal t_1, compute an eulerian walk of $\tilde{G}$ (i.e. a closed walk that passes through all edges of $\tilde{G}$ exactly once). Order the terminals $t_1, \dots, t_{|X|}$ according to the order they are encountered for the first time. The shortest path in G between two successive terminals is at most the corresponding cost over the walk. This implies $2c(T^*) \geq 2\sum_{i=1}^{|X|} d(t_i, t_{i+1})$, where we set $t_{|X|+1} = t_1$. Hence, this implies that in the metric closure, the cycle $\{t_1, t_2 \dots, t_k, t_1\}$ has cost at most $2c(T^*)$. Hence, its most expensive edge has cost at least $2c(T^*)/|X|$. By removing it we obtain a path in the metric closure of cost at most $2c(T^*)(1-1/|X|)$. This a spanning tree in the closure, hence it cannot have cost bigger than the output solution.
3. Consider the graph with n terminals and one non-terminal node v. Edge costs are $1+\epsilon$ between v and any terminal, and 2 between any two terminals. The optimum spanning tree in the closure has cost $2(n-1)$, while the optimum solution has cost $n(1+\epsilon)$. By choosing ϵ small enough, the ratio can be made arbitrarily close to $1-1/n = 1-1/|X|$, as required.

Exercise 2

1. Left to the reader
2. Let D be the set of nodes sampled in the first part of the algorithm, and T^* the Steiner tree with terminals D constructed by the algorithm (those are the edges our algorithms "buys"). Let S the be the set of edges picked in the second part (with repetitions when edges are picked multiple times): those are the edges our algorithm "rents". The total

cost of the solution output by the algorithm is $Mc(T^*) + c(S)$. Recall that in the proof of the 4-approximation we first showed

$$E(Mc(T')) \leq c(OPT), \tag{1}$$

where T' is the *optimum* Steiner tree over D. This implies $E(Mc(T^*)) \leq \alpha c(OPT)$ (with the 2 approximation for Steiner Tree we deduced $E(Mc(T^*)) \leq 2c(OPT)$). Then we showed

$$E(c(S)) \leq E(\mathrm{T}) \leq 2c(OPT),$$

where T is the tree obtained applying Prim's algorithm to the metric closure of the graph. The last inequality follows (1) and from the fact that T is a 2-approximation of T' (hence, this upper bound is *not* related to the Steiner tree we constructed in the first part). Hence we can improve the first part of the analysis, not the second, and obtain a $(\alpha+2)$-approximation algorithm in expected value.

Exercise 3

1. Let S be the optimum solution, and suppose by contradiction it is not a tree. As seen in class, we can suppose that there exists two terminals t_1 and t_2 whose paths P_1 and P_2 to r meet at some vertex v, diverges, and then meets again at some vertex u. Call $e_1, \ldots, e_k$ the edges of the path between v and u in P_1, and $e'_1, \ldots, e'_j$ those of P_2. Set $\Delta(x_{e_1}, \ldots, x_{e_k}) = (f_{e_1}(x_{e_1}+1) - f_1(x_{e_1})) + \cdots + (f_{e_k}(x_{e_k}+1) - f_{e_k}(x_{e_k}))$ and $\Delta'(x_{e'_1}, \ldots, x_{e'_j}) = (f'_1(x'_1+1) - f'_1(x'_1)) + \cdots + (f_{e'_j}(x'_j+1) - f_{e'_j}(x_{e'_j}))$. Assume wlog $\Delta(x_{e_1}, \ldots, x_{e_k}) \leq \Delta'(x_{e'_1}, \ldots, x_{e'_j})$. Since the cost function is concave and non-decreasing, we know that $\Delta(x_{e_1}, \ldots, x_{e_k}) \leq \Delta(x_{e_1}-1, \ldots, x_{e_k}-1)$ and similarly for Δ'. Hence the solution S' obtained by moving the path of t_2 between v and u to $e_1, \ldots, e_k$ has cost

$$\begin{aligned} c(S') &= c(S) + \Delta(x_{e_1}, \ldots, x_{e_k}) - \Delta(x'_{e'_1} - 1, \ldots, x'_{e'_j} - 1) \\ &\leq c(S) + \Delta(x_{e_1}, \ldots, x_{e_k}) - \Delta'(x'_{e'_1}, \ldots, x'_{e'_j}) \\ &\leq c(S). \end{aligned}$$

   Since the optimum solution was assumed to be unique, we obtain $c(S') < c(S)$, a contradiction.

2 This is a simple generalization of what we saw in Exercise 4 of the first assignment (in this case, we have to consider that each edges may assume values from 1 to $n-1$). We leave it to the reader. We can then deduce that any instance of ssrob can be perturbed so that it has a unique optimum solution, which is also the optimum of the original problem. By point [1.], it is a tree. Hence the original instance has an optimal solution that is a tree.

3.i Proceed as in 1, and deduce $\Delta(x_{e_1}, \ldots, x_{e_k}) < \Delta(x_{e_1}-1, \ldots, x_{e_k}-1)$ from strict concavity. Then $c(S') < c(S)$, hence S is not an optimal solution.

3.ii Let $f_e(x)$ be the cost function relative to edge e, and define $\tilde{f}_e(x) = f_e(x) + \alpha \log(1+x)$, with $\alpha > 0$ a constant to be fixed. It is the sum of a concave and a strictly concave function, hence it is strictly concave. Now we show how to set α so that the optimum solutions of the new problem are also optimum solutions of the original problem. Set $\epsilon = \min_{e\in E, x\in\{1,\ldots,n\}}(f_e(x) - f_e(x-1))$, and $\alpha = m/\epsilon$. Then pick two solutions S, S' to the original problem of different cost, say $f(S) < f(S')$. Then $f(S) + \epsilon \le f(S')$. Set $\alpha < \frac{\epsilon}{|E|\log|V|}$. We obtain

$$\tilde{f}(S) = \sum_e \tilde{f}_e(x_e) = \sum_e f_e(x_e) + \alpha \sum_e \ln(1+x_e) \le f(S) + \alpha|E|\ln|V| < f(S) + \epsilon \le f(S') \le \tilde{f}(S'),$$

as required. Now we proceed as in 2 and deduce that every instance of gssrob has an optimal solution that is a tree.

Exercise 4

FYI: this problem is known as *Set cover.*

1. Note that $\sum_e \text{price}_e = C(\mathscr{S})$. Let e_i be the element that is inserted into C as the i-th (break ties arbitrarily). Note that, when it is selected, there are at least $n-i+1$ elements that need to be covered. Call them T. Hence $|T| \ge n-i+1$.

   We now show that $\text{price}_{e_i} \le \frac{c(OPT)}{n-i+1}$. Suppose not. Then all the remaining elements will be covered with sets of price at least price_{e_i} (the algorithm first covers elements with smaller price) and we have:

   $$\sum_{e\in T} \text{price}_e > (n-i+1)\cdot\frac{c(OPT)}{n-i+1} = c(OPT).$$

   This is a contradiction, since in the optimum solution, elements from T are covered by set of cost at most $c(OPT)$, and those must all be available when e_i is covered (otherwise, some element from T was already covered). Hence

   $$\sum_e \text{price}_e \le \sum_{i=1}^{n} \frac{c(OPT)}{n-i+1} = (1 + \frac{1}{2} + \cdots + \frac{1}{n})c(OPT),$$

   concluding the proof.

2. Let $S = \{e_1, \ldots, e_n\}$, and $\mathscr{F}$ be formed by the sets $S_i = \{e_1, \ldots, e_i\}$ with cost $\frac{1}{n-i+1}$ for $i = 1, \ldots, n$, plus the set S' with cost $1+\varepsilon$. The optimum is S', while one can easily check that the algorithm will choose all sets S_i for a total cost $(1 + 1/2 + \cdots + 1/n)c(OPT)$.