INTRO TO APPROXIMATION – HW1

TSP and friends

Every task is worth two points. Deadline: **16. 11. 2016 23:59** via email. Please send PDFs only. High-quality scans (with high-quality handwriting) are acceptable. Do not be afraid to email me if any task is unclear to you.

EXERCISE ONE In the *Steiner tree problem* we get on input a connected undirected graph $G = (V, E)$, an edge cost function $c : E \to \mathbb{R}^+$, and finally a list of *terminals* $S \subseteq V$. A feasible solution to our problem is any subset of edges $E' \subseteq E$ so that the graph $G' = (V, E')$ has all the terminals in one connected component. We aim to minimize the cost, i.e. $\sum_{e \in E'} c(e)$. Your task is to design a 2-approximation algorithm.

Hint: The graph does not need to satisfy the triangle inequality. First, think about the case when it does (it should be easy then). To solve the general case, try to use some of the techniques from the TSP approximation.

EXERCISE TWO Consider the problem MAXSAT – given a CNF Boolean formula with clauses of any size, we need to find an assignment that satisfies as many clauses as possible (even if the full formula is unsatisfiable).

We will analyze the following algorithm:

"Try setting all variables to 0 and compute how many clauses we have satisfied. Then, try setting all variables to 1 and compute how many clauses we have satisfied. Take the bigger of the two values and return it as the approximation."

1. Thoroughly analyze the approximation ratio of the algorithm; that is, prove that it is an r-approximation algorithm and also prove that it is not an r'-approximation for any $r' < r$.
2. Let us consider any *constant-testing* algorithm – such an algorithm does the same thing as the one described above, but instead of two assignments, it tests c pre-selected assignments, where c is a constant not dependent on the input. (An assignment is any infinite sequence of 0/1 values, where we assign the first value to x_1, the second value to x_2 and so on, until we run out of variables.)
   What is the tight approximation ratio of any constant-testing algorithm? Again, you need to find a number r_2 such that some constant-testing algorithm is an r_2-approximation and prove that *no* constant-testing algorithm is strictly better than an r_2-approximation.

EXERCISE THREE Given a directed graph $\vec{G}$ with distance function $d\colon \vec{E} \to \mathbb{R}^+$, design and analyze a polynomial-time algorithm for finding the directed circuit which is shortest *on average* – it is a circuit minimizing $\frac{\sum_{\vec{e} \in \vec{C}} d(\vec{e})}{|\vec{C}|}$.

Keep in mind that the shortest averaged circuit can often be longer and have more edges than the shortest circuit overall.

EXERCISE FOUR Assume there is a polynomial-time algorithm for finding the shortest averaged circuit (see above).

Consider the following algorithm for asymmetric TSP on n elements with the asymmetric function $d\colon \{1, \ldots, n\} \times \{1, \ldots, n\} \to \mathbb{R}^+$ which satisfies the triangle inequality:

1. We find a directed circuit $\vec{C}$ in the metric which minimizes $\frac{\sum_{\vec{e} \in \vec{C}} d(\vec{e})}{|\vec{C}|}$.
2. We add all the edges $\vec{E}(\vec{C})$ to the solution.
3. We remove all vertices of $\vec{C}$ except one. We continue recursively until only one vertex remains.
4. We use shortcutting on the Eulerian walk and return a directed Hamiltonian circuit.

Prove that the previous algorithm is an $\mathcal{O}(\log n)$-approximation for asymmetric TSP.

Bonus exercise

We know from the lecture that the Christofides algorithm satisfies $\mathrm{ALG} \leq \frac{3}{2}\mathrm{OPT}$, where ALG is the value of the solution for the algorithm and OPT is the value of the minimum/optimum solution.

Let us refresh linear programming by proving that for Christofides algorithm, it is also true that $\mathrm{ALG} \leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$, where $\mathrm{OPT}_{\mathrm{LP}}$ is the optimum value of the following linear relaxation:

$$\begin{aligned}
(P): \quad & \min \sum_{e \in E} c_e x_e \\
\forall v \in V: \quad & \sum_{e=vx} x_e = 2 \\
\forall S \subsetneq V, S \neq \emptyset: \quad & \sum_{e \in E(S, V \setminus S)} x_e \geq 2 \\
\forall e \in E: \quad & 0 \leq x_e \leq 1
\end{aligned}$$

The battle plan is as follows:

1. First verify that $\mathrm{ALG} \leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$ implies the original claim of $\mathrm{ALG} \leq \frac{3}{2}\mathrm{OPT}$.
2. Next, prove that for an optimum solution x^* of the LP (P) (that is precisely the point of value $\mathrm{OPT}_{\mathrm{LP}}$) it holds that $\frac{n-1}{n}x^*$ is a point inside the spanning tree polytope for the same graph.
3. Finally, use point 2 and finish the claim that $\mathrm{ALG} \leq \frac{3}{2}\mathrm{OPT}_{\mathrm{LP}}$.

If you do not remember, the *spanning tree polytope* is the polytope given by these linear constraints:

$$\begin{aligned}
& \sum_{e \in E} x_e = n - 1 \\
\forall S \subsetneq V, S \neq \emptyset: \quad & \sum_{e \in E(S, V \setminus S)} x_e \geq 1 \\
\forall e \in E: \quad & x_e \geq 0
\end{aligned}$$

The *matching polytope* looks like this:

$$\begin{aligned}
\forall v \in V: \quad & \sum_{e=vx} x_e \leq 1 \\
\forall S \subsetneq V, S \neq \emptyset, |S| \text{odd}: \quad & \sum_{e \in E(S, V \setminus S)} x_e \geq 1 \\
\forall e \in E: \quad & x_e \geq 0
\end{aligned}$$