Exercises

Approximation Algorithms

Spring 2010

Sheet 1

Note: This is just one way, a solution could look like. We do not guarantee correctness. It is your task to find and report mistakes.

Exercise 1
Give family of undirected graphs \(G\) and terminals \(R\), such that asymptotically (i.e. for \(|V| \to \infty\)) the Minimum spanning tree is a factor 2 more expensive than the cheapest Steiner tree.

Proof. Consider a star graph with \(n\) terminals \(r_1, \ldots, r_n\) and a center \(v\) connected with unit cost edges. The cheapest Steiner tree costs \(OPT = n\). On the other hand the Minimum Spanning Tree costs \(2n - 2\). Hence the gap is \(\frac{2n - 2}{n} \to 2\). \(\square\)

Exercise 2
For the STEINER TREE problem, we are given an undirected weighted graph \(G = (V, E)\) and a set of terminals \(R \subseteq V\). It is the goal to find a tree \(T\) that connects all terminals. There exists a constant \(c_0 > 1\) such that the following gap version of the 3-SET COVER problem is NP-hard:

Given sets \(S_1, \ldots, S_m \subseteq \{1, \ldots, n\}\) with \(|S_i| = 3\) and a parameter \(k \in \mathbb{N}\), distinguish

- **YES:** There is a cover with \(\leq k\) sets
- **NO:** There is no cover with \(\leq c_0 \cdot k\) sets

Show that STEINER TREE is APX-hard, i.e. show that there is a constant \(c_1 > 1\) such that finding a \(c_1\)-approximate STEINER TREE is NP-hard.

**Hint:** Construct a STEINER TREE instance with 1 terminal for each element, 1 Steiner node per set and 1 special root terminal (unit cost edges should suffice).

Proof. Let \(U = \{1, \ldots, n\}\) be the elements. We create a graph \(G = (V, E)\) with terminals \(U \cup \{r\}\) and one potential Steiner node \(v_i\) for each set \(S_i\). We add edges \((r, v_i)\) and \((j, v_i)\) if \(j \in S_i\). All edges have unit cost. Let \(OPT\) be the cost of the optimum Steiner tree. Then

**Claim:** There is a cover with \(\leq k\) sets \(\Rightarrow\) \(OPT \leq n + k\)

Let \(I \subseteq \{1, \ldots, m\}\) be the index set of \(|I| \leq k\) sets with \(\bigcup_{i \in I} S_i = U\). Choose a Steiner Tree with edges \((r, v_i)\) for each \(i \in I\). Then attach for each terminal \(j\) an edge \((j, v_i)\) for one \(v_i\) with \(j \in S_i\) (recall that maybe \(j\) is covered several times). The cost of the tree is \(|U| + |I| \leq n + k\).
Claim: \( \text{OPT} \leq n + c_0 \cdot k \Rightarrow \) there is a cover of size \( c_0 \cdot k \).

Suppose we are given a Steiner tree \( T \) of cost \( c(T) \leq n + k \). In principle \( T \) might contain an edge \((j, j')\) of cost 2 between two terminals. But then we can replace it by 2 edges \((j, v_i), (v_i, r)\) for some \( i \) with \( j \in S_i \) (or with \((j', v_i), (v_i, r)\)) without disconnecting terminals and without increasing the cost.

Now we may assume that every terminal \( j \) is connected to the root using a path of the form \( j \to v_i \to r \).
Let \( I \) be the index set of all \( i \) with \((v_i, r) \in T \), then \(|I| \leq n + c_0 \cdot k - n = c_0 k \). Hence we found a set cover of size \( \leq c_0 k \).

We conclude

- \( \text{OPT} \leq c_0 k \Rightarrow \text{OPT}_{ST} \leq n + k \)
- \( \text{OPT} > c_0 k \Rightarrow \text{OPT}_{ST} > n + c_0 k \)

But

\[
\frac{n + c_0 k}{n + k} \geq \frac{3k + c_0 k}{3k + k} = \frac{3 + c_0}{4} =: c_1 > 1
\]

using that \( n \leq 3k \). If we had a Steiner Tree approximation algorithm with a ratio of \( < c_1 \) then we could solve the \( \text{NP} \)-hard Set Cover instance (and hence any problem in \( \text{NP} \)).

Exercise 3
Consider the \textbf{MAXIMUM COVERAGE} problem: Given sets \( S_1, \ldots, S_m \) over a universe of elements \( U = \{1, \ldots, n\} = \bigcup_{i=1}^{m} S_i \) and a parameter \( k \in \mathbb{N} \). Choose \( k \) sets that cover as many elements as possible, i.e.

\[
\text{OPT} := \max \left\{ |\bigcup_{i \in I} S_i| : |I| = k \right\}
\]

Show that a straightforward greedy algorithm gives a \( \frac{\text{OPT}}{\frac{e}{e-1}} \approx 1.58 \)-approximation.

Solution:

We consider a greedy algorithm, where in any iteration we choose that set, which covers the largest amount of new elements. Let \( n_i \) be the number of elements which are covered in the \( i \)th iteration. Suppose we are in the \( i \)th iteration, then there are still the \( \geq \text{OPT} - \sum_{j=1}^{i-1} n_j \) many elements that the optimum solution manages to cover with \( k \) sets. Hence

\[
n_i \geq \frac{\text{OPT} - \sum_{j=1}^{i-1} n_j}{k}
\]

Thus

\[
\sum_{j=1}^{i} n_j \geq \sum_{j=1}^{i-1} n_j + \frac{\text{OPT} - \sum_{j=1}^{i-1} n_j}{k} = \frac{\text{OPT}}{k} + \left(1 - \frac{1}{k}\right) \sum_{j=1}^{i-1} n_j
\]

Iterating this we obtain

\[
\sum_{j=1}^{k} n_j \geq \frac{\text{OPT}}{k} \cdot \left[ \sum_{i=0}^{k-1} \left(1 - \frac{1}{k}\right)^i \right] \cdot \text{geom.series:} \sum_{i=0}^{k-1} i \cdot \frac{e^x - 1}{e^x} = \frac{e^x - 1 - 1}{1 - e^{-1}} \cdot \text{OPT} \cdot \frac{1 - \left(1 - \frac{1}{k}\right)^k}{1 - (1 - \frac{1}{k})} \geq \left(1 - \frac{1}{e}\right) \cdot \text{OPT}
\]