1 Robust Steiner tree problem

The input to the Steiner tree problem is an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, and a subset $T \subseteq V$ called “terminals”. The objective is to find a connected subgraph $H$ that includes all the terminals $T$ and has minimum cost $c(H) := \sum_{e \in H} c_e$.

In the robust version of the Steiner tree problem, the input also contains an integer $k$ and a real number $\lambda \geq 1$. There are two stages. In the first stage the algorithm has to identify a subset $E_1 \subseteq E$ of edges to buy. In the second stage, the cost of each edge in $E \setminus E_1$ increases by a factor of $\lambda$ and a subset $T' \subseteq T$ of at most $k$ terminals is revealed. We refer to $T'$ as a “scenario”. The algorithm, in the second stage, has to augment the solution $E_1$ by buying edges $E_2(T')$ so that the resulting graph $E_1 \cup E_2(T')$ includes a Steiner tree on terminals $T'$. The choice of edges $E_2(T')$ is allowed to depend on the subset $T'$. The overall cost of this solution is thus

$$\sum_{e \in E_1} c_e + \lambda \cdot \sum_{e \in E_2(T')} c_e.$$

The objective is to minimize the maximum overall cost over all scenarios, i.e., to minimize

$$\max_{T' \subseteq T, |T'| \leq k} \lambda \cdot \sum_{e \in E_2(T')} c_e.$$

The edge-costs $c_e$ induce a shortest-path metric on the vertices $V$: for any two vertices $u, v \in V$, we use $d(u, v)$ to denote the length of the shortest path between $u$ and $v$, under costs $c_e$ in graph $G$.

1.1 The algorithm

Let $E_1^*$ and $E_2^*(T')$ be the set of edges optimum buys in the first stage and the second stage for scenario $T'$ respectively. Let $OPT = OPT_1 + \lambda \cdot OPT_2$ be the overall cost of the optimum, where $OPT_1 = \sum_{e \in E_1} c_e$ is its cost in the first stage and $OPT_2 = \max_{T' \subseteq T, |T'| \leq k} \sum_{e \in E_2^*(T')} c_e$ is the maximum cost in the second stage divided by $\lambda$.

**First stage.** Our algorithm, in the first stage, guesses the value of $OPT_2$. It then computes a subset of terminals $C = \{c_1, c_2, \ldots, c_p\} \subseteq T$ called “centers” and an assignment $\pi : T \to C$ that satisfy:

- The centers are far apart: $d(c_i, c_j) > rOPT_2/k$ for all $i \neq j$, and
- Each terminal is close to its assigned center: $d(t, \pi(t)) \leq rOPT_2/k$ for all $t \in T$,

where $r > 1$ is a constant to be determined later. Such a clustering can be computed as follows. Pick any terminal and name it $c_1$. Assign all terminals within a distance of $rOPT_2/k$ from $c_1$ to $c_1$ and remove these terminals. Pick any one of the remaining terminals and name it $c_2$, and so on.

The algorithm then computes an approximate minimum-cost Steiner tree $T$ in $G$ on the centers $C$ under the costs $c_e$. Currently, the best known polynomial-time algorithm for the Steiner tree problem is $\gamma$-approximate, where $\gamma < 1.55$.

The algorithm buys the edges in the Steiner tree in the first stage.

**Second stage.** In the second stage a subset $T'$ of at most $k$ terminals is revealed. The algorithm, in the second stage, buys the shortest path from each terminal $t \in T'$ to its assigned center $\pi(t)$. 

1
Remark 1.1 (Guessing \( \text{OPT}_2 \)) The algorithm in fact tries all guesses of \( \text{OPT}_2 \) that are powers of \( (1 + \epsilon) \) and takes the cheapest solution for any of these guesses. To simplify the presentation below, we assume that the guess on \( \text{OPT}_2 \) is exact.

It is easy to see that the algorithm computes a feasible solution to the problem.

1.2 The analysis

It is easy to see that the algorithm pays at most \( \lambda \cdot r \cdot \text{OPT}_2 \) in the second stage. This holds since the distance of any terminal to its assigned center is at most \( r \cdot \text{OPT}_2 / k \). Since at most \( k \) terminals need to be connected to their centers, the total cost of these connections is at most \( \lambda \cdot k \cdot r \cdot \text{OPT}_2 / k \).

We now bound the cost of the algorithm in stage one using the following lemma.

Lemma 1.2 Assuming \( r > 4 \), there exists a Steiner tree on centers \( C \) in \( G \) that has cost at most \( \frac{1}{r - 1} \cdot \text{OPT}_1 + \text{OPT}_2 \).

Proof: Recall that \( E^*_1 \) is the set of edges optimum buys in stage one and \( \text{OPT}_1 = \sum_{c \in E^*_1} c_e \). Let \( H \) be a graph obtained from \( G \) by shrinking the edges in \( E^*_1 \). We now perform another clustering of the centers \( C \) in the shortest-path metric on \( C \) induced by the graph \( H \) as follows. We identify a subset of centers \( L = \{l_1, l_2, \ldots, l_t\} \) and a mapping \( \phi : C \rightarrow L \) such that

- \( d_H(l_i, l_j) > 2 \cdot \text{OPT}_2 / k \) for all \( i \neq j \), and
- \( d(c, \phi(c)) \leq 2 \cdot \text{OPT}_2 / k \) for all centers \( c \in C \),

where \( d_H \) denotes the shortest-path distance in the graph \( H \). Such a clustering can be computed as follows. Pick any center and name it \( l_1 \). For all centers \( c \in C \) with \( d_H(c, l_1) \leq 2 \cdot \text{OPT}_2 / k \), define \( \phi(c) = l_1 \). Remove all such centers from \( C \) and repeat.

We now argue that \( |L| < k \). Assume on the contrary that \( |L| \geq k \) and let \( T' \subseteq L \) be any subset of size \( k \). Consider the scenario \( T' \). Since even after shrinking the edges in \( E^*_1 \) that optimum bought in the first stage, any two centers in \( L \) are more than \( 2 \cdot \text{OPT}_2 / k \) apart, the spheres of radius \( \text{OPT}_2 / k \) centered at the centers in \( T' \) in \( H \) are disjoint. Therefore the minimum Steiner tree on \( T' \) in \( H \) has cost more than \( \text{OPT}_2 \). This is a contradiction since the optimum pays at most \( \text{OPT}_2 \) in the second phase to connect all the centers in \( T' \) after shrinking the edges in \( E^*_1 \).

Since \( |L| < k \), we now consider scenario \( L \). There exists a Steiner tree \( E^*_L \) on \( L \) in \( H \) with cost at most \( \text{OPT}_2 \). Thus \( E^*_1 \cup E^*_L \) has cost at most \( \text{OPT}_1 + \text{OPT}_2 \) and contains a Steiner tree on \( L \) in \( G \). We now show how to extend this into a subgraph with low cost and which contains a Steiner tree on \( C \) in \( G \).

Now recall that the pairwise distance between centers \( C \) in \( G \) is at least \( r \cdot \text{OPT}_2 / k \). Thus spheres of radius \( r \cdot \text{OPT}_2 / (2k) \) around the centers \( C \) are disjoint in \( G \). Note however that \( d_H(c, \phi(c)) \leq 2 \cdot \text{OPT}_2 / k \) for all centers \( c \in C \). Thus at least \( r \cdot \text{OPT}_2 / (2k) - 2 \cdot \text{OPT}_2 / k = (r/2 - 2) \cdot \text{OPT}_2 / k \) cost of \( E^*_1 \cup E^*_L \) must lie inside the sphere of radius \( r \cdot \text{OPT}_2 / (2k) \) around each center \( c \in C \). We can thus extend the subgraph \( E^*_1 \cup E^*_L \) by adding shortest paths from each \( c \) to \( \phi(c) \) in \( H \) and charge this additional cost to the contribution of \( E^*_1 \) in the respective spheres around \( c \in C \). The resulting subgraph clearly constains a Steiner tree on \( C \) in \( G \). The overall cost of this subgraph is thus at most

\[
\text{OPT}_1 + \text{OPT}_2 + \frac{2}{r/2 - 2} \cdot \text{OPT}_1 = \frac{r}{r - 4} \cdot \text{OPT}_1 + \text{OPT}_2.
\]

Hence the proof. \( \square \)
Since we use a $\gamma$-approximation algorithm to compute a Steiner tree in stage one, the overall cost of stage one is at most
\[ \frac{\gamma \cdot r}{r - 4} \cdot \text{OPT} + \gamma \cdot \text{OPT}_2. \]
Combining this with the second stage cost, the overall cost of our solution is
\[ \frac{\gamma \cdot r}{r - 4} \cdot \text{OPT} + (\gamma + \lambda r) \cdot \text{OPT}_2. \]
Comparing this with the optimum cost $\text{OPT}_1 + \lambda \cdot \text{OPT}_2$ and setting $r = 2(1 + \sqrt{1 + \gamma})$, we get that the algorithm is an $5.2$-approximation.

(In the above calculation, I have taken the worst case $\lambda = 1$. But in this case “don’t buy anything in stage one” does better. So the approximation factor can be reduced further by balancing this with the above algorithm. Calculations deferred.)