CS-E400204: Approximation Algorithms **Spring (Fifth Period) 2022**

Lecture 2 (April 25): Steiner Tree, TSP and Multiway Cut

Lecturer: Kamyar Khodamoradi *Scribe: Sameep Dahal*

2.1 Introduction

In this lecture, we will discuss about the following problems.

1. Steiner Tree Problem
2. Metric Travelling Salesperson Problem (Metric TSP)
3. Multiway Cut problem

2.2 Steiner Tree Problem

In this section, we will design a 2 factor approximation algorithm for the Steiner Tree Problem.

2.2.1 Problem Description

An instance of an Steiner Tree Problem is defined by a connected graph $G = (V, E)$ and a cost function $c : E \mapsto \mathbb{Q}_{\geq 0}$ where the vertex set $V = S \sqcup T$ is partitioned into two sets S and T. The set of vertices in S are called Steiner (or optional) vertices whereas the set of vertices in T are known as Terminal (or mandatory) vertices. For an instance $I = (G = (V = S \sqcup T, E), c)$ of this problem, a feasible solution is a tree $B = (V', E')$ such that $T \subseteq V'$ and $E' \subseteq E$. Each feasible solution is called a Steiner Tree. The cost of the tree B is defined as $\sum_{e \in E'} c(e)$ which just denotes the sum of edge costs of the tree B. Our aim in this problem is to find the minimum cost Steiner Tree.

In short, the problem can be defined in the following manner.

1. input = (G, c) with connected graph $G = (V = S \sqcup T, E)$ and cost function $c : E \mapsto \mathbb{Q}_{\geq 0}$
2. output = minimum cost Steiner tree $B = (V', E')$ satisfying $T \subseteq V'$ and $E' \subseteq E$. Cost of B is defined as $\sum_{e \in E'} c(e)$.

2.2.2 Metric Steiner Tree Problem

We define a version of the Steiner Tree problem called the Metric Steiner Tree problem. The instance of this problem is also $(G = (V = S \sqcup T, E), c)$. Additionally, we have that G is a complete graph and thus c can be defined as $c : V \times V \mapsto \mathbb{Q}_{\geq 0}$. We require that this cost function c is a metric i.e. it satisfies the following 3 properties.

1. Identity: $\forall v \in V,\ c(v,v) = 0$.

2. Symmetry: $\forall u, v \in V,\ c(u,v) = c(v,u)$.

3. Triangle Inequality: $\forall u, v, w \in V,\ c(u,v) \leq c(u,w) + c(w,v)$.

The properties of a metric are closely related to that of distance in the usual sense. A common metric function is the Euclidean distance function in the space $\mathbb{R}^n$.

It turns out that designing efficient approximation algorithm for Metric Steiner Tree problem is sufficient for designing efficient approximation algorithm to the original Steiner Tree problem because we can create an approximation preserving reduction from the Steiner Tree problem to the Metric Steiner Tree problem.

2.2.3 Approximation Preserving Reductions

For a pair of minimisation problems Π_1, Π_2, a pair of polynomial time algorithms (f, g) is called an approximation preserving reduction from Π_1 to Π_2 if the following conditions hold.

1. For any instances $I_1 \in D_{\Pi_1}$, we define $I_2 = f(I_1)$. Then, we require that $I_2 \in D_{\Pi_2}$ and $\mathrm{OPT}_{\Pi_2}(I_2) \leq \mathrm{OPT}_{\Pi_1}(I_1)$. This condition enforces that the function f maps an instance of Π_1 to an instance of Π_2 without increasing the cost of the optimal solution.

2. For any feasible solution $t \in S_{\Pi_2}(I_2)$, we define $s = g(t)$. Then, we require that $s \in S_{\Pi_1}(I_1)$ and $obj_{\Pi_1}(I_1, s_1) \leq obj_{\Pi_2}(I_2, s_2)$. This condition enforces the function g to map from feasible solution of instance I_2 to some feasible solution of I_1 without increasing the cost of the feasible solution.

Theorem 1 *Let (f, g) be an approximation preserving reduction from Π_1 to Π_2. If Π_2 admits an α-approximation algorithm, so does Π_1.*

Proof. Suppose A is the polynomial time algorithm for the α-factor approximation for Π_2. Then, the following defines the algorithm for an α factor approximation for the problem Π_1.

Algorithm 1 An α approximation algorithm for Π_1

Input: An instance I_1 of Π_1
Output: A feasible solution s of I_1

$I_2 \leftarrow f(I_1)$
$t \leftarrow A(I_2)$
$s \leftarrow g(t)$
return s

From the definitions it follows that,

$$obj_{\Pi_1}(I_1, s) \leq obj_{\Pi_2}(I_2, t) \leq \alpha \mathrm{OPT}(I_2) \leq \alpha \mathrm{OPT}(I_1)$$

■

2.2.4 Approximation Preserving Reduction from Steiner Tree problem to Metric Steiner Tree Problem

Theorem 2 *There exists an approximation preserving reduction (f, g) from the Steiner Tree problem to the metric Steiner Tree problem*

Proof.

First, we show a polynomial time function f that reduces an instance $I_1 = (G_1 = (V = S \sqcup T, E_1), c_1)$ of Steiner Tree problem (Π_1) to an instance $I_2 = (G_2 = (V, E_2), c_2)$ of Metric Steiner Tree problem (Π_2) with same set S and T.

The edge set E_2 consists of all possible edges between vertices in V. The cost function c_2 is just the metric completion of c_1. It means that for any pair of vertices $u, v \in V$,

$$c_2(u, v) = \text{min cost path from } u \text{ to } v$$

First we verify that c_2 is indeed a metric. The identity and symmetry property follows directly from the definition. The triangle inequality follows from the fact that any counter example to the inequality would produce a shorter path from u to v which contradicts the definition of c_2. Since, computing shortest path can be done in polynomial time, it follows that f can be computed in polynomial time.

We notice that for any pair of vertices $u, v \in V$, $c_2(u, v) \leq c_1(u, v)$. Since the vertex set of both the graphs are the same and we are just adding edges in E_1, it follows that every feasible solution of I_1 is also feasible for I_2. Consider x^* to be the optimal solution for I_1. We see that x^* is feasible for I_2. We see that,

$$\text{OPT}_{\Pi_2}(I_2) \leq c_2(x^*) \leq c_1(x^*) = \text{OPT}_{\Pi_1}(I_1)$$

Now, we design a polynomial time algorithm g that maps a feasible solution of I_2 to I_1. We notice that feasible solution to I_2 is a tree $B_2 = (V_2', E_2')$. We create an instance $\hat{B_1}$ of I_1 from B_2 in the following way. First, we create a graph $B_1 = (V_1', E_1')$ in the following way. For each edge $uv \in E_2'$, add all the edges in min cost path from u to v responsible for the value of $c_2(u, v)$ to E_1'. V_1 consists of all vertices in E_1'. In this way, we see that the cost of edges in E_1' is currently atmost the cost of edges in E_2'. Since, we are adding edges, the vertices in V_1' are still connected however B_1 may have cycles. Thus, we can find any spanning tree of B_1 and call it $\hat{B_1}$. We see that the sum of edge costs in $\hat{B_1} \leq$ sum of edge cost in $B_1 \leq$ sum of edge costs in B_2.

■

Because of the above theorem, we can now focus on designing good approximation algorithms for the Metric Steiner Tree problem to guarentee good approximation factor for the general Steiner Tree Problem.

Approximation Algorithm for Metric Steiner Tree problem

Theorem 3 *Metric Steiner Tree admits a $(2 - \frac{2}{|T|})$ approximation algorithm where T denotes the set of terminals.*

Proof.

Our algorithm just returns the minimum cost spanning tree of the subgraph induced by the set of terminals T.

Algorithm 2 Approximation Algorithm for Metric Steiner Tree problem

Input: Complete graph $G = (V = S \sqcup T, E)$ with metric c
Output: Steiner tree B

$B \leftarrow$ minimum cost spanning tree of $G[T]$
return B

In order to analyse our algorithm, we consider the following 2 sets.

- P = Minimum cost Hamiltonian path of $G[T]$
- C = Minimum cost Hamiltonian Cycle of $G[T]$

Hamiltonian path of $G[T]$ is a path that visits every vertex of $G[T]$ exactly once whereas a Hamiltonian cycle of $G[T]$ is a cycle in $G[T]$ which visits all vertices of $G[T]$.

Since a Hamiltonian path is also a spanning tree of $G[T]$ and the solution returned by the above algorithm B is the minimum cost spanning tree of $G[T]$, we get that $cost(B) \leq cost(P)$. On the other hand, we notice that removing an edge from any cycle results in a path. In particular, if we remove the largest cost edge from the cycle C, we get a Hamiltonian path of $G[T]$ of cost atmost $(1 - \frac{1}{|T|})cost(C)$. Since, P is the minimum cost Hamiltonian path of $G[T]$, we get $cost(P) \leq (1 - \frac{1}{|T|})cost(C)$.

Overall, we have: $cost(B) \leq cost(P) \leq (1 - \frac{1}{|T|})cost(C)$.

Now, we desire to upper bound $cost(C)$ by the optimal solution of the Metric Steiner Tree problem. For this purpose, let's say that B^* is an optimal solution to the given instance for the Metric Steiner Tree problem and its cost be OPT. Now, we double each edge of B^* with the same cost and call this graph $\hat{B}$. We notice that each vertex of $\hat{B}$ now has even degree which means that $\hat{B}$ is Eulerian and has a Eulerian tour.

Since we doubled each edge from B^* to obtain $\hat{B}$, we have $cost(\hat{B}) = 2\text{OPT}$. Now, we use $\hat{B}$ to obtain a Hamiltonian cycle $\hat{C}$ of $G[T]$ such that $cost(\hat{P}) \leq cost(\hat{B}) = 2\text{OPT}$. Once, we obtain such a $\hat{P}$, it follows that

$$cost(B) \leq (1 - \frac{1}{|T|})cost(C) \leq (1 - \frac{1}{|T|})cost(\hat{C}) \leq (2 - \frac{2}{|T|})\text{OPT}$$

So, the only thing remaining is to obtain $\hat{C}$ from $\hat{B}$. For that, we consider the Eulerian tour $\hat{R} = \{v_1, v_2, \ldots, v_r\}$ of $\hat{B}$. We can shortcut $\hat{R}'$ to obtain $\hat{C}$ in the following way.

- Start from a terminal vertex in the tour $\hat{R}'$, call is v_{cur}.
- Look at the next unvisited terminal vertex to the right of the tour, call it v_{next}.
- Delete all the nodes in the path from v_{cur} to v_{next} and directly add an edge between v_{cur} and v_{next}. Because our cost function is a metric, the triangle inequality gives us that cost of moving directly from v_{cur} to v_{next} cannot exceed the cost of the path from v_{cur} to v_{next} in the tour. This process is called short-cut.
- Assign v_{cur} to v_{next} and continue the process until we are left with a hamiltonian cycle in $G[T]$.

The process can be visualised better through figure 2.1.

Since the process of short-cut never increases the overall cost, it follows that the obtained Hamiltonian cycle $\hat{C}$ satisfes $cost(\hat{C}) \leq cost(\hat{B})$ and this completes our proof.

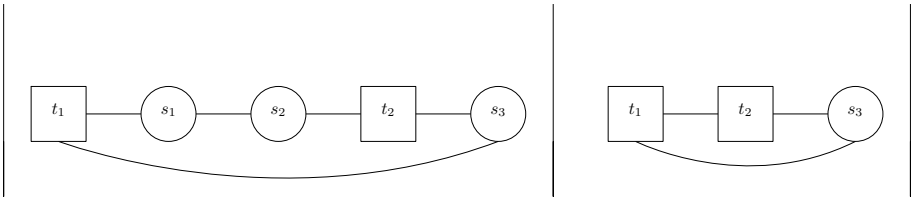

Figure 2.1: The graph on the right shows the graph formed by taking a direct shortcut from t_1 to t_2 by removing the vertices in the middle of the path

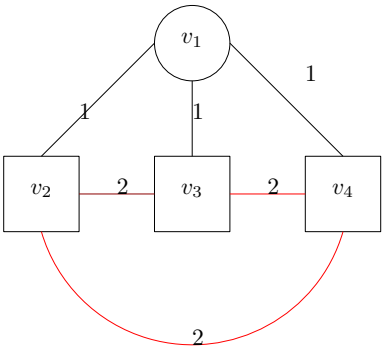

Figure 2.2: Tight example with 3 terminals (squares) and 1 steiner (circle) vertex

■

Now, we show that our analysis for the algorithm is tight as well. This means that we produce an instance of the problem where the solution given by our algorithm is exactly $(2 - \frac{2}{|T|})$-factor. For this, consider a graph G with $n+1$ vertices $v_1, \ldots, v_{n+1}$. v_1 is the only Steiner vertex and $v_2, \ldots, v_{n+1}$ are terminals. We define the cost function as $c(v_1, v_i) = 1$ for $i \geq 2$ and $c(v_i, v_j) = 2$ for $i, j \geq 2$. The optimal solution for this problem would be a star graph rooted as v_1 of cost n. However, any spanning tree of only the terminals has a cost of $2(n-1)$. Thus, the approximation ratio is $\frac{2(n-1)}{n} = (2 - \frac{2}{|T|})$. Fig 2.2 shows an example when $n = 4$.

2.3 Metric TSP

In this section, we will design a 2 factor approximation algorithm for the Metric TSP problem.

2.3.1 Problem Description

An instance of this problem is a complete graph $G = (V, E)$ along with a metric $c : V \times V \mapsto \mathbb{Q}_{\geq 0}$. The goal of the problem is to find the cheapest hamiltonian cycle of the graph.

Unlike the Steiner Tree Problem, enforcing Metric conditions is essential for designing good approximation algorithms for the TSP problem. Under the assumption of P $\neq$ NP, we know that there can be no α factor approximation algorithm for the general TSP problem for any $\alpha > 0$.

2.3.2 Approximation Algorithm for Metric TSP

Theorem 4 *Metric Steiner Tree admits a* $(2 - \frac{2}{n})$ *approximation algorithm where* n *is the number of vertices of the graph.*

Proof.

The following algorithm returns a hamiltonian cycle $\hat{C}$ of G.

Algorithm 3 Approximation Algorithm for Metric TSP problem

Input: Complete graph $G = (V, E)$ with metric c
Output: Hamiltonian cycle C

$B \leftarrow$ minimum cost spanning tree of G
$\hat{B} \leftarrow$ Eulerian Tour of graph with edges in B doubled
$\hat{C} \leftarrow$ short-cut of $\hat{B}$
return $\hat{C}$

As done in the analysis for Metric Steiner Tree Problem, we consider the following 2 sets.

- $P =$ Minimum cost Hamiltonian path of G
- $C =$ Minimum cost Hamiltonian Cycle of G (The optimum solution for the problem)

With the same reasoning as before, the minimum spanning tree B obtained in the algorithm satisfies

$$cost(B) \leq cost(P) \leq (1 - \frac{1}{n})cost(C) = (1 - \frac{1}{n})OPT$$

Now our solution $\hat{C}$ satisfies,

$$cost(\hat{C}) \leq 2cost(B) \leq (2 - \frac{2}{n})\text{OPT}$$

■

Remark 1 In case of the Metric TSP problem, there is an algorithm by Christofies that achieves an approximation factor of $\frac{3}{2}$. Recently, this factor has been slightly improved to $(\frac{3}{2} - \epsilon)$ for $\epsilon \geq 10^{-36}$ [Karlin, Klein and Gharan, STOC 2021 best paper].

2.4 Multiway Cut Problem

In this section, we will design a 2 approximation algorithm for the multiway cut problem.

Definition 1 (Multiway Cut) Given a graph $G = (V, E)$ and a subset of vertices $T \subseteq V$, we say that a subset of edges $E' \subseteq E$ is a multiway cut of set T in graph G, if no pair of vertices $t_i, t_j \in T$ are connected in the graph $(V, E \setminus E')$.

2.4.1 Problem Formulation

An instance of the problem is a graph $G = (V, E)$, a cost function $c : E \mapsto \mathbb{Q}_{\geq 0}$ and a set $T = \{t_1, \ldots, t_k\} \subseteq V$ of terminals. The objective of the problem is to find the cheapest multiway cut of T in the given graph.

Let $k = |T|$ throughout this section. When $k = 2$, we have a well known s-t cut problem which can be solved in polynomial time. For $k \geq 3$, the problem is known to be NP-hard.

In order to design an approximation algorithm for the multiway cut problem, we introduce the notion of isolating cut.

Definition 2 (Isolating Cuts) For a vertex $t \in T$, $E' \subseteq E$ is called an isolating cut for t if it seperated t from the vertices in T. In other words, there is no path from t to vertices in $T \setminus \{t\}$ in the graph $(V, E \setminus E')$.

The task of finding a minimum cost isolating cut for any vertex $t \in T$ can be reduced to the task of finding a minimum s-t cut. The reduction creates a new graph G' by introducing a new source vertex s and connects it to all vertices in $T \setminus \{t\}$ with infinite cost. Because the new introduced edges are of infinite cost and the set of all edges incident to t is already an isolating cut of finite cost, it follows that the minimum s-t cut in G' will not include any of the new edges. Due to our construction, it implies that this set of edges that seperates s from t in G' must seperate t from $T \setminus \{t\}$ as well.

2.4.2 Approximation Algorithm for the Multiway Cut Problem

Theorem 5 *The multiway cut problem admits a* $(2 - \frac{2}{k})$ *approximation algorithm.*

Proof.

The following algorithm returns a multiway cut C' for the set T.

Algorithm 4 Approximation Algorithm for Multiway Cut Problem

Input: Complete graph $G = (V, E)$ with cost function $c : E \mapsto \mathbb{Q}_{\geq 0}$ and set of terminals $T = \{t_1, \ldots, t_k\}$
Output: Multiway cut C' for T

$\forall t_i \in t$, compute $C_i \leftarrow$ minimum cost isolating cut for t_i
Without loss of generality we consider $C_1, \ldots, C_k$ be the ordering of the isolating cuts from cheapest to most expensive
return $C = \bigcup_{i=1}^{k-1} C_i$, the $k-1$ cheapest isolating cut

Consider A to be the optimum multiway cut for T. For each $i = 1, \ldots, k$, consider $A_i \subseteq A$ to be the minimum cost subset of edges of A that seperates t_i from the rest of the vertices. For each $i = 1, \ldots, k$, A_i is a feasible isolating cut whereas C_i is an optimum isolating cut. So, it follows that $cost(C_i) \leq cost(A_i)$. Since we consider the $k-1$ cheapest isolating cuts,we get that,

$$cost(C) \leq (1 - \frac{1}{k}) \sum_{i=1}^{k} cost(C_i) \leq (1 - \frac{1}{k}) \sum_{i=1}^{k} cost(A_i) \leq 2(1 - \frac{1}{k})\text{OPT}$$

.

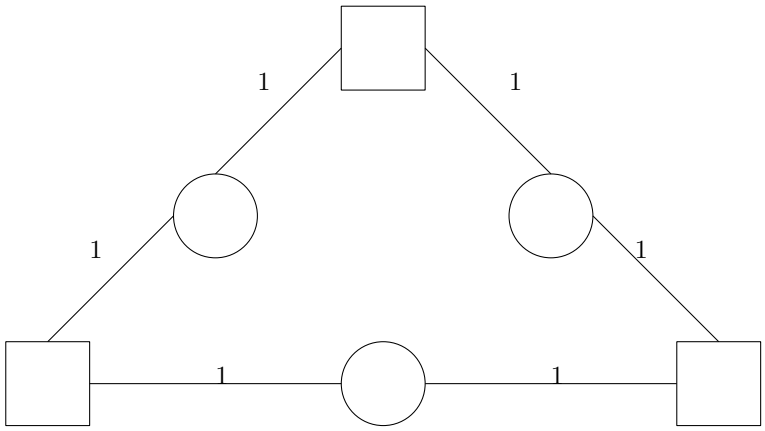

Figure 2.3: Tight example for multiway cut with 3 terminals (squares)

The last inequality $\sum_{i=1}^{k} cost(A_i) \leq 2\text{OPT}$ comes from the fact that each edge in A will belong to atmost 2 isolating cuts A_i because each edge will connect the 2 components in the graph $(V, E \setminus A)$.

■

Our analysis for the designed algorithm is tight as well. We call this graph where the approximation factor is achieved as G. G has k terminals, The graph G is a clique K_k all of whose edges have been subdivided to form a vertex. Thus, there is a path of length 2 between each pair of terminals. The cost of each edge in G is 1. For this graph G, our algorithm returns a solution of cost $(k-1)^2$ as each isolating cut has cost $(k-1)$ whereas the optimal cost is $k(k-1)/2$. Thus, the ratio is $\frac{2(k-1)^2}{k(k-1)} = 2 - \frac{2}{k}$ as desired.