Simpler Approximations for the Network Steiner Tree Problem

Vaidehi Srinivas
Advisor: Anupam Gupta

SCS Senior Thesis

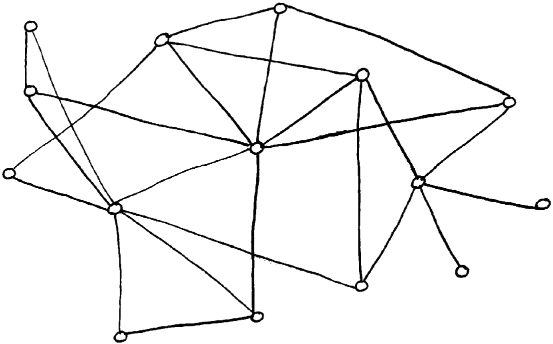
May 6, 2020

Overview

- Introduce the Steiner Tree Problem
- Describe previous results
- Introduce Submodular Function Maximization
- Show how we used this to simplify previous approximation algorithms

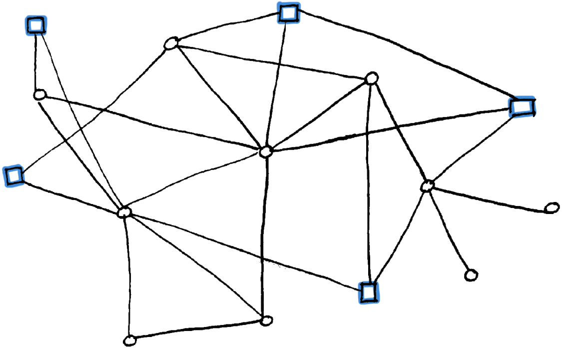
The Network Steiner Tree Problem

- Fix a graph $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$



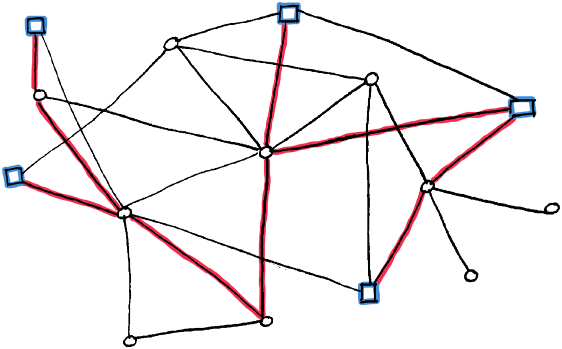
The Network Steiner Tree Problem

- Fix a graph $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$
- and **terminals** $V \subseteq V_{\text{full}}$ (blue)



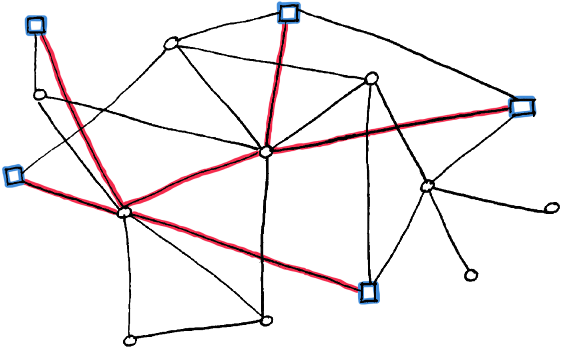
The Network Steiner Tree Problem

- Fix a graph $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$
- and **terminals** $V \subseteq V_{\text{full}}$ (blue)
- A **Steiner Tree** connects all terminals (red)



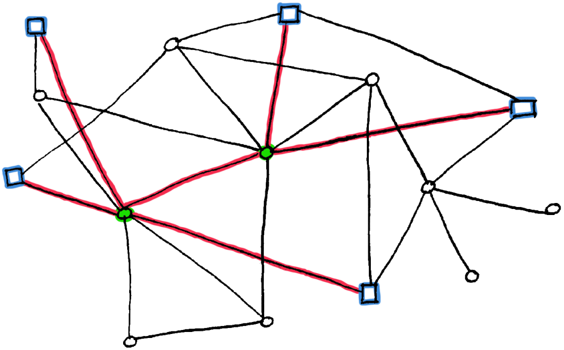
The Network Steiner Tree Problem

- Fix a graph $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$
- and **terminals** $V \subseteq V_{\text{full}}$ (blue)
- A **Steiner Tree** connects all terminals (red)
- Want to find minimum cost Steiner Tree



The Network Steiner Tree Problem

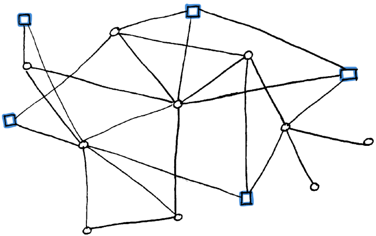
- Fix a graph $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$
- and **terminals** $V \subseteq V_{\text{full}}$ (blue)
- A **Steiner Tree** connects all terminals (red)
- Want to find minimum cost Steiner Tree
- Non-terminals in a Steiner Tree are **Steiner nodes** (green)

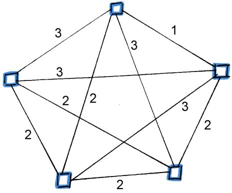


The Shortest Path Graph

In general, this is NP-hard, so we look for polynomial-time approximations.

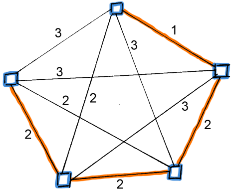
We can associate $G_{\text{full}} = (V_{\text{full}}, E_{\text{full}})$ with $G = (V, E)$, the shortest path graph on just the terminals.





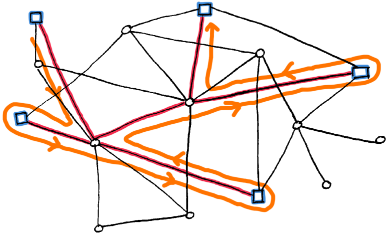
MST of Shortest Path Graph is a 2-Approximation

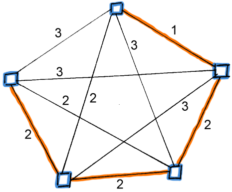
- The MST of G gives a 2-approximation.



MST of Shortest Path Graph is a 2-Approximation

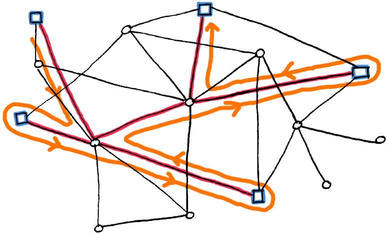
- The MST of G gives a 2-approximation.
- We use the optimal Steiner Tree in G_{full} to construct a spanning tree in G.

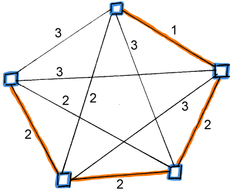




MST of Shortest Path Graph is a 2-Approximation

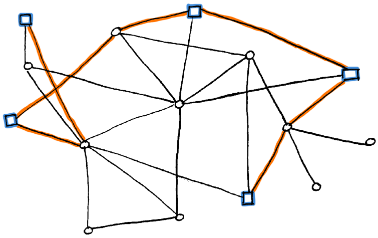
- The MST of G gives a 2-approximation.
- We use the optimal Steiner Tree in G_{full} to construct a spanning tree in G.
- Denote the MST of G as $\text{MST}(G)$. Denote the cost of $\text{MST}(G)$ as $\text{mst}(G)$.

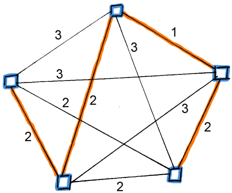




MST of Shortest Path Graph is a 2-Approximation

Where do we lose when we use the MST?

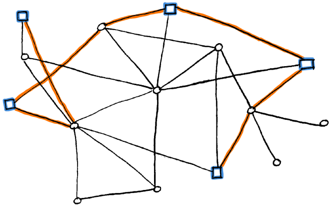


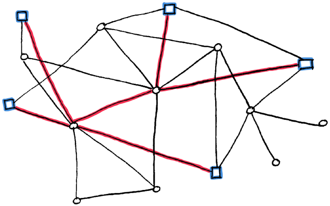


MST of Shortest Path Graph is a 2-Approximation

Where do we lose when we use the MST?

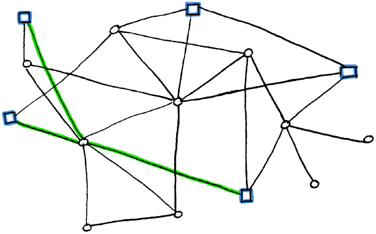
We don't reuse the Steiner nodes effectively, so we redo work.

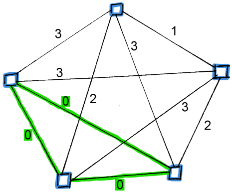




Contracting Components

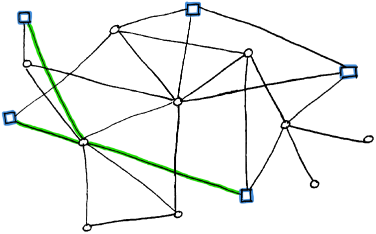
- We contract a component K by sinking the cost for K, and setting the connections within K to 0 in G.

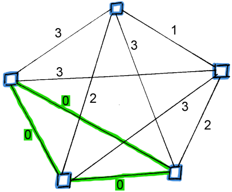




Contracting Components

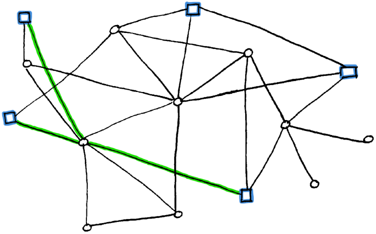
- We contract a component K by sinking the cost for K, and setting the connections within K to 0 in G.
- We denote this $G[K]$.

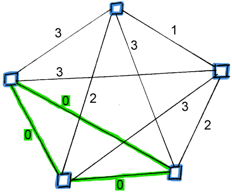




Contracting Components

- We contract a component K by sinking the cost for K, and setting the connections within K to 0 in G.
- We denote this $G[K]$.
- Define: $\text{save}_G(K) = \text{mst}(G) - \text{mst}(G[K])$.





Zelikovsky's 11/6-Approximation [Zel93]

- Zelikovsky's algorithm contracts a series of triples in G.

Zelikovsky's 11/6-Approximation [Zel93]

- Zelikovsky's algorithm contracts a series of triples in G.
- It greedily chooses triples K to maximize save_G, such that $\mathrm{save}_G(K) \geq \mathrm{cost}(K)$.

Zelikovsky's 11/6-Approximation [Zel93]

- Zelikovsky's algorithm contracts a series of triples in G.
- It greedily chooses triples K to maximize save_G, such that $\text{save}_G(K) \geq \text{cost}(K)$.

In the analysis, Zelikovsky proves that:

- A greedy choice of triples does well compared to the optimal choice of triples.
- There is a set of triples that we can contract that approximates the best Steiner Tree well.

How do we simplify these?

- This is a great result, but the analysis is very technical.

How do we simplify these?

- This is a great result, but the analysis is very technical.
- Can we simplify the argument?

How do we simplify these?

- This is a great result, but the analysis is very technical.
- Can we simplify the argument?

Key Observation [Cha18]

The save_G function that we are optimizing is **submodular**.

Submodular Functions

For a universe U, a function f over subsets of U

$$f : \mathcal{P}(U) \to \mathbb{R}$$

is **submodular**, if for any $A \subseteq B \subseteq U$, and new element $e \notin B$

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B).$$

Submodular Functions

For a universe U, a function f over subsets of U

$$f : \mathcal{P}(U) \to \mathbb{R}$$

is **submodular**, if for any $A \subseteq B \subseteq U$, and new element $e \notin B$

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B).$$

The **marginal value** of adding e to a subset is larger.

An Example of a Submodular Function

- Task: build cell towers

An Example of a Submodular Function

- Task: build cell towers
- Universe (U): potential tower locations

An Example of a Submodular Function

- Task: build cell towers
- Universe (U): potential tower locations
- Value (f): sum of data connection speeds to each house

An Example of a Submodular Function

Consider the marginal value of e (blue) to set A (solid towers).

An Example of a Submodular Function

Consider the marginal value of e (blue) to set B (solid towers), a superset of A.

An Example of a Submodular Function

- Each potential tower has a cost.

An Example of a Submodular Function

- Each potential tower has a cost.
- Question: Given a budget, what is the highest value set we can build?

An Example of a Submodular Function

- Each potential tower has a cost.
- Question: Given a budget, what is the highest value set we can build?
- This is an example of a **knapsack constraint**.

Submodularity: Back to the General Case

There is a polynomial time $(1 - \frac{1}{e})$-approximation for submodular function maximization with respect to a knapsack constraint [NWF78].

Submodularity: Back to the General Case

There is a polynomial time $\left(1 - \frac{1}{e}\right)$-approximation for submodular function maximization with respect to a knapsack constraint [NWF78].

- Algorithm greedily chooses item that has largest (marginal value) / (cost), with some edge cases.

Submodularity: Back to the General Case

There is a polynomial time $\left(1 - \frac{1}{e}\right)$-approximation for submodular function maximization with respect to a knapsack constraint [NWF78].

- Algorithm greedily chooses item that has largest (marginal value) / (cost), with some edge cases.
- $1 - \frac{1}{e}$ is the best polynomial-time approximation unless $\mathrm{P} = \mathrm{NP}$.

Submodularity: Back to the General Case

There is a polynomial time $(1 - \frac{1}{e})$-approximation for submodular function maximization with respect to a knapsack constraint [NWF78].

- Algorithm greedily chooses item that has largest (marginal value) / (cost), with some edge cases.
- $1 - \frac{1}{e}$ is the best polynomial-time approximation unless $\mathsf{P} = \mathsf{NP}$.
- Small edit allows us to increase cost of S, in exchange for increased value.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

Our Results

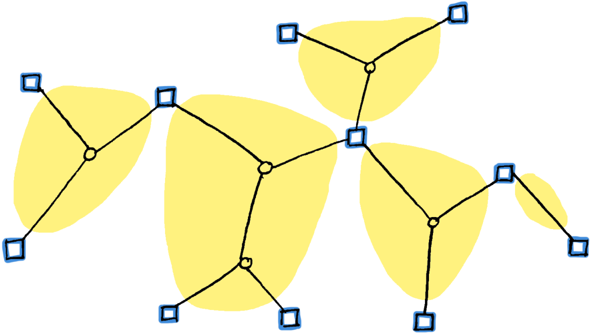
We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.



Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.
- If we contract the components of the optimal tree, T^*, the remaining MST has cost 0.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.
- If we contract the components of the optimal tree, T^*, the remaining MST has cost 0.
- This is a set of components Z^* with cost OPT, and save_G value $\text{mst}(G)$.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.
- If we contract the components of the optimal tree, T^*, the remaining MST has cost 0.
- This is a set of components Z^* with cost OPT, and save_G value $\text{mst}(G)$.
- Using submodular maximization, we can find a set of components, Z, with cost OPT, and save_G value $\left(1 - \frac{1}{e}\right)\text{mst}(G)$.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.
- If we contract the components of the optimal tree, T^*, the remaining MST has cost 0.
- This is a set of components Z^* with cost OPT, and save_G value $\text{mst}(G)$.
- Using submodular maximization, we can find a set of components, Z, with cost OPT, and save_G value $\left(1 - \frac{1}{e}\right) \text{mst}(G)$.
- We get a tree that has cost $\text{OPT} + \frac{1}{e}\text{mst}(G)$.

Our Results

We adapt the idea of contracting full components, as they do in [Zel93], and apply submodular function maximization.

- Any Steiner Tree is a set of disjoint components.
- If we contract the components of the optimal tree, T^*, the remaining MST has cost 0.
- This is a set of components Z^* with cost OPT, and save_G value $\text{mst}(G)$.
- Using submodular maximization, we can find a set of components, Z, with cost OPT, and save_G value $\left(1 - \frac{1}{e}\right) \text{mst}(G)$.
- We get a tree that has cost $\text{OPT} + \frac{1}{e}\text{mst}(G)$.
- This is $\left(1 + \frac{2}{e}\right) \text{OPT}$.

Our Results

Improving this idea for [Zel93]:

- Tuning cost v. benefit in submodular maximization gives us an approximation factor

$$1 + \ln 2 + \epsilon \approx 1.693.$$

- Compare to the original 11/6-approximation.

Our Results

Improving this idea for [Zel93]:

- Tuning cost v. benefit in submodular maximization gives us an approximation factor

$$1 + \ln 2 + \epsilon \approx 1.693.$$

- Compare to the original 11/6-approximation.

Based on modified contraction idea from Robins and Zelikovsky [RZ05]:

- Approximation factor:

$$1 + \frac{\ln 3}{2} + \epsilon \approx 1.693.$$

- This matches their bound.

Takeaways

- We simplified combinatorial analyses of Steiner Tree approximations by viewing them as instances of Submodular Function Maximization.

Takeaways

- We simplified combinatorial analyses of Steiner Tree approximations by viewing them as instances of Submodular Function Maximization.
- In the future we could look at whether this approach can simplify other approximations for this problem, based on linear programming and other techniques.

References

Deeparnab Chakrabarty.
Personal conversation, August 2018.

George Nemhauser, Laurence Wolsey, and M. Fisher.
An analysis of approximations for maximizing submodular set functions—i.
Mathematical Programming, 14:265–294, 12 1978.

Gabriel Robins and Alexander Zelikovsky.
Tighter bounds for graph steiner tree approximation.
SIAM J. Discrete Math., 19(1):122–134, 2005.

A. Zelikovsky.
An 11/6-approximation algorithm for the network steiner problem.
Algorithmica, 9:463–470, 05 1993.