Steiner Tree Notes

D-nice Kennon
Stewart Veliky
Jay Winkler

March 12, 2020

1 Defining the Precise Problem

Given a graph and a subset of vertices in the graph, find a Steiner Tree that spans through the subset. The Steiner Tree can contain vertices that are not present in the subset but are used to connect the vertices in the subset. For Minimum Steiner Tree, it will be the shortest path through the **Terminal Vertices**. (Vertices that must be included from the original graph into the Steiner Tree) Vertices that are not within the set of terminal vertices that are used from the existing graph or added to complete the tree are **Steiner Vertices**

Steiner Tree vs. Minimum Spanning Tree

A Steiner Tree shares several similarities with a minimum spanning tree, however there are distinct differences to note. A minimum spanning tree must span through all vertices within a graph whereas a Steiner Tree does not have that requirement. It is merely the shortest path between all terminal vertices.

Steiner Tree vs. Dijkstra's Algorithm

Using Dijkstra's algorithm, it will look at edges and work its way to the end node by following the least weighted edge. The Steiner Tree, however, will look at edges and could similarly work its way to the final node, or it could add additional vertices and edges which will be of less weight. By this definition, there will not always be an identical path found in Dijkstra's algorithm to that of a Minimum Steiner Tree.

2 Real World Problems and Uses

- Networking and Cellphone towers
  - Verizon determining where to add a new tower that will be most cost efficient and cover all desired locations

- GPS
  - Finding the shortest route when going on a road trip
- Highway and Railway construction
  - For city planners and contractors determining the best possible location for a new road to reduce travel distance
  - This can be especially useful when considering weights as some locations may be more ideal for a new road than others
    - Cutting through a citizen's property may hold greater weight
    - Using government property may hold smaller weight

3 Steiner Tree is in NP

The Steiner Tree problem can be verified in polynomial time given the following:

- Given a tree, breadth or depth first searches can be used
  - Either are polynomial time algorithms
- Touch all terminal edges
  - Given a graph, this can be computed in polynomial time or less
- Adding weights can be done n times
- Checking for tree validity can also be computed in polynomial time or less

4 Steiner Tree is NP-Hard

The Steiner Tree problem is at least as hard as the hardest problems in NP. To show this, we will use a reduction proof from Vertex Cover

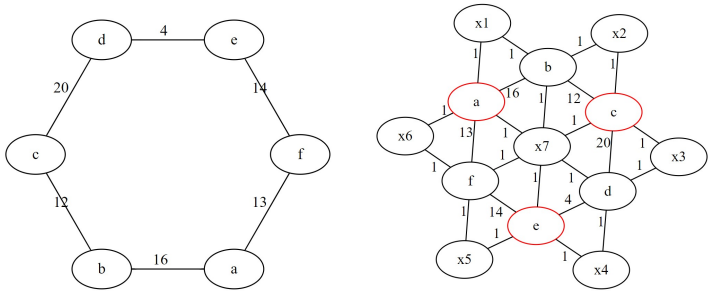
Recall, Vertex cover of a graph is a set of vertices s.t. each edge of the graph is incident to at least one vertex of the set

Reduction: Vertex Cover to Minimum Steiner Tree

- Given a graph, G, we will create a new graph, H to find the Minimum Steiner Tree (our gadget)
- Add a vertex for every edge in G to the new graph, H
- Create edges between the two vertices (a line graph)
- These new vertices for this reduction will be denoted as set X
  - We will need M new edges to connect the vertices

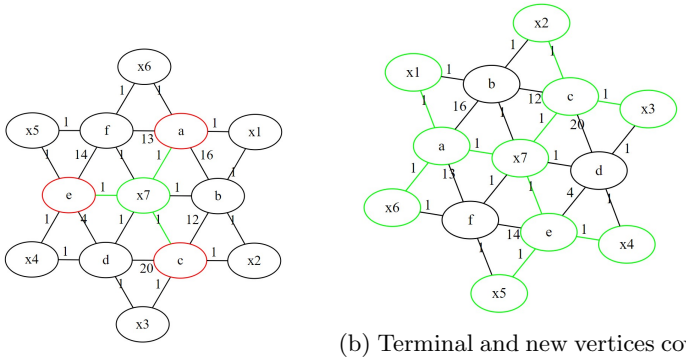
– K represents the original vertex cover size

- The new graph will have a Steiner tree of cost C = M + K IFF the original graph has a vertex cover of size K



(a) Original Graph G (b) New Graph H

Figure 1: Creating the Gadget



(a) Connect Center Point First (b) Terminal and new vertices covered

Figure 2: Determining Gadget Validity

5 Examples

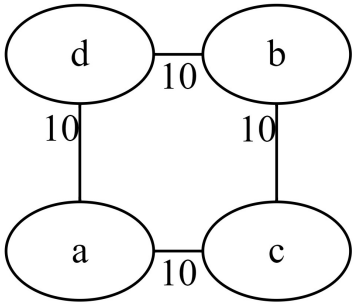
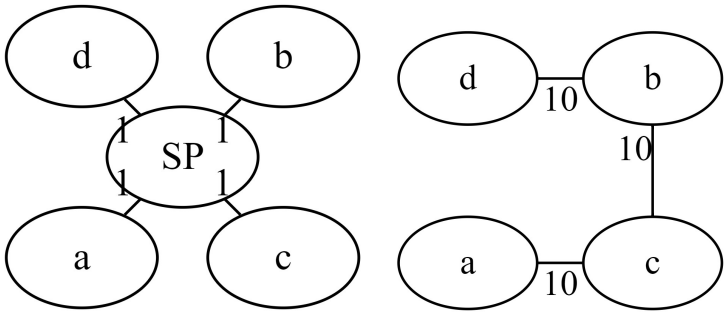
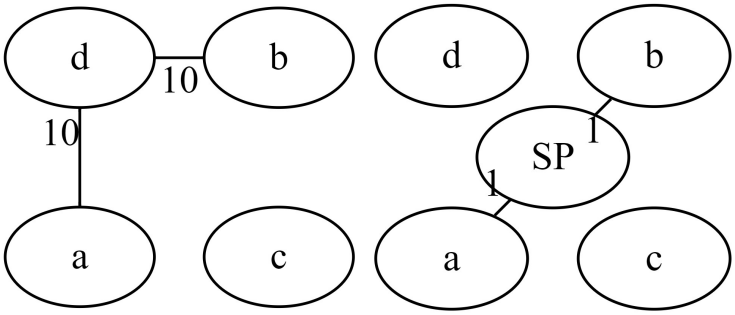


Figure 3: Graph G



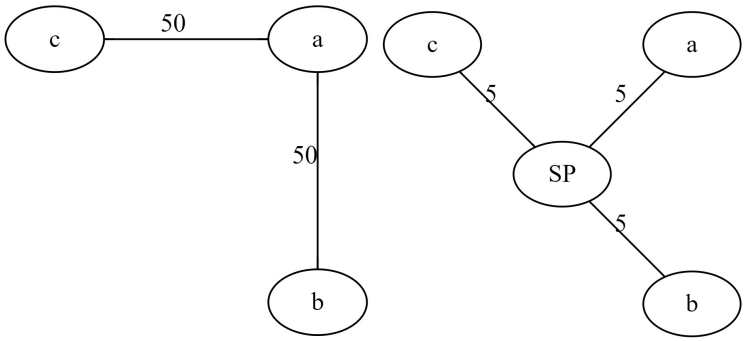
(a) Steiner Tree (b) MST

Figure 4: Steiner Tree vs. Minimum Spanning Tree given Figure 3



(a) Dijkstra (b) Steiner Tree

Figure 5: Dijkstra vs Steiner Tree Given Vertices a to b of Figure 3



(a) Graph G (b) Steiner Tree of G

Figure 6: Example Steiner Tree