(1991) The Minimum Steiner Tree Problem

Peter Gordon, Casey Cao-Son, & Brennan Truong

California State University, Fullerton

Mathematical Modeling, Semester Project

peter.gordon@csu.fullerton.edu
dangkhoa27@csu.fullerton.edu
brennantruong@gmail.com

Dec. 10, 2014

Introduction / Goal

- Modern communication is based on wired networks.

Introduction / Goal

- Modern communication is based on wired networks.
- Devices are connected through cables to transmit information.

Introduction / Goal

- Modern communication is based on wired networks.
- Devices are connected through cables to transmit information.
- Total cost of network proportional to amount of cabling used:

$$\text{Cost} \propto \sum_{i \in \{\text{cables}\}} \text{Length}_i$$

Introduction / Goal

- Modern communication is based on wired networks.
- Devices are connected through cables to transmit information.
- Total cost of network proportional to amount of cabling used:

$$\text{Cost} \propto \sum_{i \in \{\text{cables}\}} \text{Length}_i$$

- Propagation delay ($\frac{d}{s}$) is also proportional to total amount of cabling:

$$\frac{d}{s} \propto \sum_{i \in \{\text{cables}\}} \text{Length}_i$$

Introduction / Goal

- Modern communication is based on wired networks.
- Devices are connected through cables to transmit information.
- Total cost of network proportional to amount of cabling used:

$$\text{Cost} \propto \sum_{i \in \{\text{cables}\}} \text{Length}_i$$

- Propagation delay ($\frac{d}{s}$) is also proportional to total amount of cabling:

$$\frac{d}{s} \propto \sum_{i \in \{\text{cables}\}} \text{Length}_i$$

- **Goal:** Minimize cost and delay by reducing amount of cabling.

Definition: Weighted Graph / Adjacency

Weighted Graph

- A *weighted graph* $G = (V, E)$ is a pair of a set of points (called *vertices*) V and the set of *edges*, E, where each edge in E is a 3-tuple consisting of two vertices in V combined with a nonnegative real number *weight*.
  That is, each edge is an element of the form $(v_a, v_b, w_{a,b})$ with v_a, $v_b \in V$ and $w_{a,b} \in \{x \in \mathbb{R} \,|\, x > 0\}$.

Definition: Weighted Graph / Adjacency

Weighted Graph

- A *weighted graph* $G = (V, E)$ is a pair of a set of points (called *vertices*) V and the set of *edges*, E, where each edge in E is a 3-tuple consisting of two vertices in V combined with a nonnegative real number *weight*.
  That is, each edge is an element of the form $(v_a, v_b, w_{a,b})$ with v_a, $v_b \in V$ and $w_{a,b} \in \{x \in \mathbb{R} \,|\, x > 0\}$.
- The *weight of G* is the sum of all its edge weights.

Definition: Weighted Graph / Adjacency

Weighted Graph

- A *weighted graph* $G = (V, E)$ is a pair of a set of points (called *vertices*) V and the set of *edges*, E, where each edge in E is a 3-tuple consisting of two vertices in V combined with a nonnegative real number *weight*.
  That is, each edge is an element of the form $(v_a, v_b, w_{a,b})$ with v_a, $v_b \in V$ and $w_{a,b} \in \{x \in \mathbb{R} \,|\, x > 0\}$.
- The *weight of G* is the sum of all its edge weights.

Adjacent Vertices

- Given a weighted graph $G = (V, E)$, two vertices v_1 and v_2 in V are *adjacent* provided that there exists an edge between them.

Definition: Connectedness / Vertex Order

Connected Vertices

- Any two vertices v_a and v_b are *connected* provided that there exists an ordered sequence P of vertices $(v_{P_0} = v_a, v_{P_1}, v_{P_2}, \ldots, v_{P_{n-1}}, v_{P_n} = v_b)$ such that every sequential pair of vertices is adjacent. That is, v_a is adjacent to v_{P_1}, v_{P_1} is adjacent to v_{P_2}, and so on.

Definition: Connectedness / Vertex Order

Connected Vertices

- Any two vertices v_a and v_b are *connected* provided that there exists an ordered sequence P of vertices $(v_{P_0} = v_a, v_{P_1}, v_{P_2}, \ldots, v_{P_{n-1}}, v_{P_n} = v_b)$ such that every sequential pair of vertices is adjacent. That is, v_a is adjacent to v_{P_1}, v_{P_1} is adjacent to v_{P_2}, and so on.
- P is called the *path between* v_a and v_b (or alternatively, v_a and v_b are *connected through* P).

Definition: Connectedness / Vertex Order

Connected Vertices

- Any two vertices v_a and v_b are *connected* provided that there exists an ordered sequence P of vertices $(v_{P_0} = v_a, v_{P_1}, v_{P_2}, \ldots, v_{P_{n-1}}, v_{P_n} = v_b)$ such that every sequential pair of vertices is adjacent. That is, v_a is adjacent to v_{P_1}, v_{P_1} is adjacent to v_{P_2}, and so on.
- P is called the *path between* v_a and v_b (or alternatively, v_a and v_b are *connected through* P).
- A graph G is a *connected graph* when all possible pairs of vertices are connected.

Definition: Connectedness / Vertex Order

Connected Vertices

- Any two vertices v_a and v_b are *connected* provided that there exists an ordered sequence P of vertices $(v_{P_0} = v_a, v_{P_1}, v_{P_2}, \ldots, v_{P_{n-1}}, v_{P_n} = v_b)$ such that every sequential pair of vertices is adjacent. That is, v_a is adjacent to v_{P_1}, v_{P_1} is adjacent to v_{P_2}, and so on.
- P is called the *path between* v_a and v_b (or alternatively, v_a and v_b are *connected through* P).
- A graph G is a *connected graph* when all possible pairs of vertices are connected.

Vertex Order

Given a weighted graph $G = (V, E)$, The *order (degree) of a vertex* v is the number of edges which connect v to another vertex in G.

Definition: MST / Rectilinear Distance

Minimum Spanning Tree

- Given a connected graph $G = (E, V)$, the *spanning tree* T of G is a connected graph with the same vertices as G, whose edges form a subset of E, and in which each pair of vertices is connected through exactly one path.

Definition: MST / Rectilinear Distance

Minimum Spanning Tree

- Given a connected graph $G = (E, V)$, the *spanning tree* T of G is a connected graph with the same vertices as G, whose edges form a subset of E, and in which each pair of vertices is connected through exactly one path.
- T is the *minimum spanning tree* of G provided that the weight of T is the smallest among all spanning trees of G.

Definition: MST / Rectilinear Distance

Minimum Spanning Tree

- Given a connected graph $G = (E, V)$, the *spanning tree* T of G is a connected graph with the same vertices as G, whose edges form a subset of E, and in which each pair of vertices is connected through exactly one path.
- T is the *minimum spanning tree* of G provided that the weight of T is the smallest among all spanning trees of G.

Rectilinear Distance

The *rectilinear distance*, d_1, between two points is the sum of the absolute values of the difference of their like coordinates.

$$d_1(p, q) = |p_x - q_x| + |p_y - q_y|$$

Problem Statement

The General Minimum Rectilinear Steiner Tree Problem

(#3) Given a set of Cartesian points $S = \{p_1, p_2, \ldots, p_n\}$, find the minimum spanning tree, using rectilinear distances, which connects all points in S using only horizontal and vertical line segments, adding any intermediary points as needed.

Problem Statement

The General Minimum Rectilinear Steiner Tree Problem

(#3) Given a set of Cartesian points $S = \{p_1, p_2, \ldots, p_n\}$, find the minimum spanning tree, using rectilinear distances, which connects all points in S using only horizontal and vertical line segments, adding any intermediary points as needed.

Specific Data

(#1) Find the minimal-cost rectilinear spanning tree for a network with the following nine stations:

$$a(0,15), \quad b(5,20), \quad c(16,24), \quad d(20,20), \quad e(33,25)$$

$$f(23,11), \quad g(35,7), \quad h(25,0), \quad i(10,3) \quad .$$

Problem Statement

Weighted Stations

(#2) Find a minimal-cost rectilinear spanning tree for the above network, but where each station adds a weight of $d^{3/2}w$, where $w = 1.2$ and d is the degree of the vertex.

Problem Statement

Weighted Stations

(#2) Find a minimal-cost rectilinear spanning tree for the above network, but where each station adds a weight of $d^{3/2}w$, where $w = 1.2$ and d is the degree of the vertex.

Method to Our Madness

- Questions answered out of their given order?

Problem Statement

Weighted Stations

(#2) Find a minimal-cost rectilinear spanning tree for the above network, but where each station adds a weight of $d^{3/2}w$, where $w = 1.2$ and d is the degree of the vertex.

Method to Our Madness

- Questions answered out of their given order?
- Creating a general solution first makes the other two "plug-and-chug"!

Hanan's Theorem

Hanan Grid

Given a set of points, its *Hanan Grid* is a grid composed of horizontal and vertical lines that intersect at all of the points.

Hanan's Theorem

Hanan Grid

Given a set of points, its *Hanan Grid* is a grid composed of horizontal and vertical lines that intersect at all of the points.

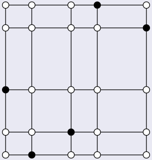


Figure: Hanan grid generated for 5 points, by Jeffrey Sharkey, on Wikipedia.

Hanan's Theorem

Hanan Grid

Given a set of points, its *Hanan Grid* is a grid composed of horizontal and vertical lines that intersect at all of the points.

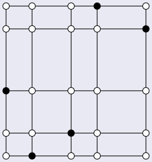
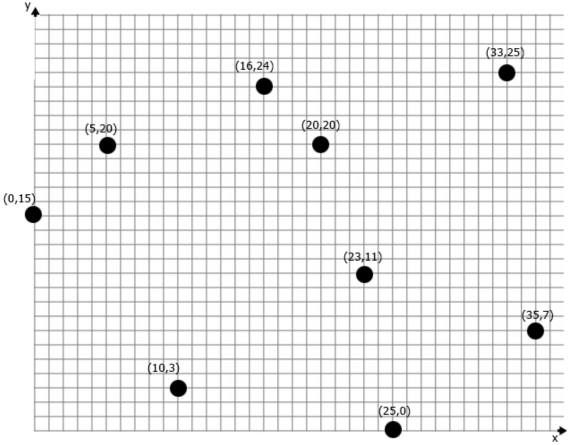


Figure: Hanan grid generated for 5 points, by Jeffrey Sharkey, on Wikipedia.

Hanan's Theorem

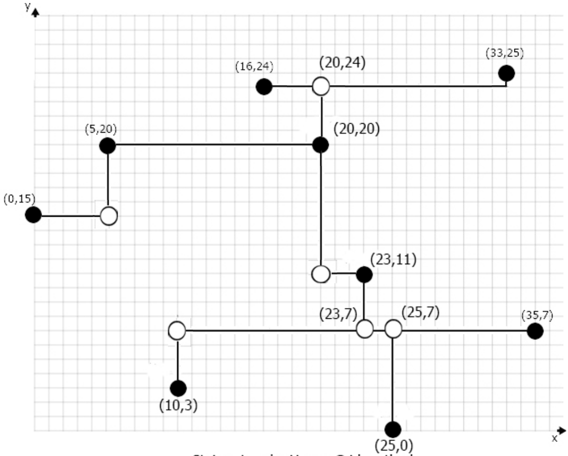
In 1966, Maurice Hanan proved: any MRST of a set of points *must* have vertices on its Hanan grid.

Manual MRST Construction via Hanan Grid



Desmonstrate Given Set of Points

Manual MRST Construction via Hanan Grid



Steiner tree by Hanan Grid method

Prim's Algorithm

Generates the minimum spanning tree of a connected graph.

Prim's Algorithm

Generates the minimum spanning tree of a connected graph.

Algorithm

- Step 1: Create a new tree, adding an arbitrary vertex from the graph.

Prim's Algorithm

Generates the minimum spanning tree of a connected graph.

Algorithm

- Step 1: Create a new tree, adding an arbitrary vertex from the graph.
- Step 2: Of all edges that connect this tree to vertices not yet in the tree, find the smallest-weight edge and add that to the tree.

Prim's Algorithm

Generates the minimum spanning tree of a connected graph.

Algorithm

- Step 1: Create a new tree, adding an arbitrary vertex from the graph.
- Step 2: Of all edges that connect this tree to vertices not yet in the tree, find the smallest-weight edge and add that to the tree.
- Step 3: Repeat step 2 until all vertices are in the tree.

Using MSTs to Find the Minimum Steiner Tree

(Naïve) Hanan/Prim Hybridization

- Step 1: Construct the Hanan Grid for the given points.

Using MSTs to Find the Minimum Steiner Tree

(Naïve) Hanan/Prim Hybridization

- Step 1: Construct the Hanan Grid for the given points.
- Step 2: Repeated remove all 1- and 2-order points not adjacent to a starting vertex.

Using MSTs to Find the Minimum Steiner Tree

(Naïve) Hanan/Prim Hybridization

- Step 1: Construct the Hanan Grid for the given points.
- Step 2: Repeated remove all 1- and 2-order points not adjacent to a starting vertex.
  (This reduces the Hanan Grid by removing "outside" points.)

Using MSTs to Find the Minimum Steiner Tree

(Naïve) Hanan/Prim Hybridization

- Step 1: Construct the Hanan Grid for the given points.
- Step 2: Repeated remove all 1- and 2-order points not adjacent to a starting vertex.
  (This reduces the Hanan Grid by removing "outside" points.)
- Step 3: Using Prim's Algorithm, find the MST of this resulting graph.

Using MSTs to Find the Minimum Steiner Tree

(Smarter) Modifying Prim's Algorithm

Rather than construct a full Hanan Grid, we can apply Prim's algorithm directly; with a few changes:

Using MSTs to Find the Minimum Steiner Tree

(Smarter) Modifying Prim's Algorithm

Rather than construct a full Hanan Grid, we can apply Prim's algorithm directly; with a few changes:

- In step 2: Keep a running list of distances from each disconnected vertex to its closest neighbor in the growing tree. (Needed for variable cardinality of the growing tree.)

Using MSTs to Find the Minimum Steiner Tree

(Smarter) Modifying Prim's Algorithm

Rather than construct a full Hanan Grid, we can apply Prim's algorithm directly; with a few changes:

- In step 2: Keep a running list of distances from each disconnected vertex to its closest neighbor in the growing tree. (Needed for variable cardinality of the growing tree.)
- In step 2: When adding the edge, augment the growing tree with a Steiner point, if needed.

Using MSTs to Find the Minimum Steiner Tree

(Smarter) Modifying Prim's Algorithm

Rather than construct a full Hanan Grid, we can apply Prim's algorithm directly; with a few changes:

- In step 2: Keep a running list of distances from each disconnected vertex to its closest neighbor in the growing tree. (Needed for variable cardinality of the growing tree.)
- In step 2: When adding the edge, augment the growing tree with a Steiner point, if needed.
  - If near an existing path, insert Steiner point on that path.

Using MSTs to Find the Minimum Steiner Tree

(Smarter) Modifying Prim's Algorithm

Rather than construct a full Hanan Grid, we can apply Prim's algorithm directly; with a few changes:

- In step 2: Keep a running list of distances from each disconnected vertex to its closest neighbor in the growing tree. (Needed for variable cardinality of the growing tree.)
- In step 2: When adding the edge, augment the growing tree with a Steiner point, if needed.
  - If near an existing path, insert Steiner point on that path.
  - Otherwise, choose the more "central" Steiner point.

Results: Hanan/Prim Hybridization

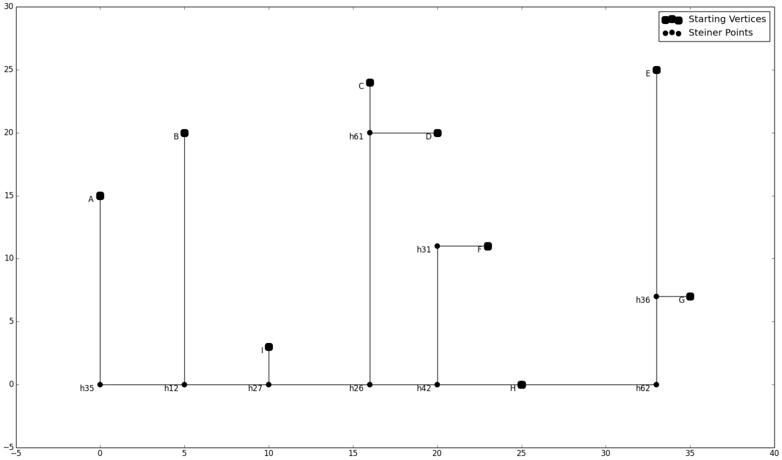


Figure: Result MRST – Total Weight: 140 & 187.59

Results: Modified Prim's Algorithm

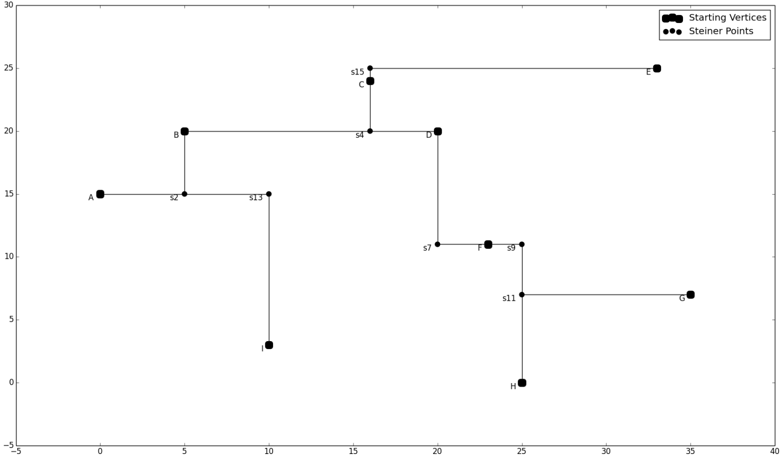


Figure: Result MRST – Total Weight: 140 & 187.59

Discussion: Comparison of Methods

Naïve Hanan/Prim Hybridization

Modified Prim's Algorithm

Discussion: Comparison of Methods

Naïve Hanan/Prim Hybridization

- (+) Simpler code!

Modified Prim's Algorithm

- (–) More code complexity (needs to maintain state, e.g. minimum-distance list)

Discussion: Comparison of Methods

Naïve Hanan/Prim Hybridization

- (+) Simpler code!
- (+) Holistic algorithm: MRST constructed with full knowledge of graph

Modified Prim's Algorithm

- (–) More code complexity (needs to maintain state, e.g. minimum-distance list)
- (–) Greedy algorithm: MRST constructed by taking the best choice at each iteration

Discussion: Comparison of Methods

Naïve Hanan/Prim Hybridization

- (+) Simpler code!
- (+) Holistic algorithm: MRST constructed with full knowledge of graph
- (−) Constructs entire Hanan Grid: **very slow** for large data sets!
  e.g.: Several minutes running time for $n \approx 50$, hours for $n \gtrapprox 200$

Modified Prim's Algorithm

- (−) More code complexity (needs to maintain state, e.g. minimum-distance list)
- (−) Greedy algorithm: MRST constructed by taking the best choice at each iteration
- (+) Only adds Steiner points on an as-needed basis: **much faster**
  e.g.: < 1 second for $n \approx 50$, minutes for $n \gtrapprox 1000$

Discussion: Comparison of Methods

Naïve Hanan/Prim Hybridization

- (+) Simpler code!
- (+) Holistic algorithm: MRST constructed with full knowledge of graph
- (−) Constructs entire Hanan Grid: **very slow** for large data sets!
  e.g.: Several minutes running time for $n \approx 50$, hours for $n \gtrsim 200$
- (−) Likely to use more Steiner points for larger sets.

Modified Prim's Algorithm

- (−) More code complexity (needs to maintain state, e.g. minimum-distance list)
- (−) Greedy algorithm: MRST constructed by taking the best choice at each iteration
- (+) Only adds Steiner points on an as-needed basis: **much faster**
  e.g.: < 1 second for $n \approx 50$, minutes for $n \gtrsim 1000$
- (+) Attempts to minimize the number of Steiner points to add.

Model Assumptions

- No obstacles in laying cables (such as gas mains, water pipes, terrain);

Model Assumptions

- No obstacles in laying cables (such as gas mains, water pipes, terrain);
- Cables have uniform cost (i.e., copper coaxial, some fiber optic, etc.)

Model Assumptions

- No obstacles in laying cables (such as gas mains, water pipes, terrain);
- Cables have uniform cost (i.e., copper coaxial, some fiber optic, etc.)
- Cables can only go in straight horizontal/vertical lines. (i.e., Steiner point not necessary only to change direction)

Works Cited

- Dreyer,D., and Overton,M. (1998). *Two heuristics for the Euclidean Steiner tree problem*. Journal of Global Optimization, 13(1), 95-106.
- Greenbaum, A. (2006). Discrete Mathematical Model [Lecture Note]. Retrieved from `https://www.math.washington.edu/greenbau/Math_381/notes/381notes.pdf`
- Hanan, M., *On Steiner's problem with rectilinear distance*. SIAM J. Applied Math., 14:255 – 265, 1966.
- Image: Hanan grid generated for a 5-terminal case. (C) 2007 Jeffrey Sharkey, retrieved from `https://en.wikipedia.org/wiki/File:Hanan5.svg`; licensed under the Creative Commons Attribution-ShareAlike license.

Thank You

Questions?