Math 443/543 Graph Theory Notes 4: Connector Problems

David Glickenstein

October 9, 2014

1 Trees and the Minimal Connector Problem

Here is the problem: Suppose we have a collection of cities which we want to connect by a system of railway lines. Also suppose we know the cost of constructing lines between pairs of cities. How do we construct the system so that the cost is minimal, regardless of the inconvenience to the passengers.

Definition 1 *A* network *(or* weighted graph*) is a graph G together with a map $\phi : E \to \mathbb{R}$. The function ϕ may represent the length of an edge, or conductivity, or cross-sectional area or many other things.*

Given a network (G, ϕ), we can define the weight of a subgraph $H \subset G$ to be

$$\phi(H) = \sum_{e \in E(H)} \phi(e).$$

The problem is then: given two vertices $u_0, v_0 \in V(G)$, find a u_0v_0-path of smallest weight (where we consider the path as a subgraph).

NOTE: We will assume that $\phi(e) > 0$ for the remainder of the section, as this simplifies exposition.

We are trying to construct a network of minimal cost. Here are some observations:

- The solution must be connected.
- The solution should contain no cycles, for if there were any cycles, we could remove one of the edges to get a smaller cost, and passengers could still get from one place to the other. Thus we want every edge to be a bridge.

Such a graph is called a tree.

Definition 2 *A graph with no cycles is called a* forest*. A connected graph with no cycles is called a* tree*.*

Note that each component of a forest is a tree. Trees kind of look like branches of a tree, which is where the name comes from.

Here are some properties of trees.

Theorem 3 *Let u and v be two vertices of a tree G. Then there is exactly one uv-path in G.*

Proof. Since G is connected, there is at least one uv-path. Suppose there were two uv-paths, $P = v_0, v_1, v_2, \ldots, v_k$ and $P' = v'_0, v'_1, v'_2, \ldots, v'_{k'}$ (note that k need not equal k'). If $P \neq P'$, then we can define s and f as

$$s = \min \left\{ i : v_{i+1} \neq v'_{i+1} \right\}$$
$$f = \max \left\{ i : v_{i-1} \neq v'_{i-1} \right\}.$$

We can then construct two disjoint paths between v_s and v_f, producing a cycle. But there are no cycles in G, so we must have $P = P'$. ■

Also, recall the following:

Theorem 4 *If a (p, q)-graph G is a tree, then $q = p - 1$.*

We come back to the minimal connector problem.

Definition 5 *If G is a connected graph, a subgraph T which is a tree and contains every vertex of G is called a* spanning tree.

We wish to find a spanning tree of minimal value. Let's find a particular tree called an *economy tree*. We construct the subgraph T_E starting with all of the vertices of G and then add edges one at a time:

1. First add the edge of minimal weight.
2. Continue to add edges of minimal weight unless they would form a cycle with the previously added edges.
3. Stop when the graph is connected.

Since no cycles are produced and the graph ends up connected and reaching every vertex, we produce a spanning tree. One might ask whether we can, in fact do this; that is, is it possible that no edges can be added that would not result in a cycle, yet the graph is not connected. If the subgraph T_E at some stage is not connected, then consider a path between two different components (which exists since G was connected). If all edges on the path not already in T_E (there must be some) create a cycle when added, then the two components were already connected (by the other part of the cycle), a contradiction.

Note that the economy tree is not necessarily unique. We may have many of them.

The theorem is that the economy tree solves the minimal connector problem.

Theorem 6 *Let (G, ϕ) be a network. The economy tree T_E has minimal weight among all spanning trees.*

Proof. Let T_0 be a spanning tree of minimal weight. We will show that $\phi(T_E) \leq \phi(T_0)$, which implies equality since $\phi(T_0)$ is minimal among all spanning trees. We can order the edges in T_E by the weights, i.e., $E(T_E) = \{e_1, e_2, \ldots, e_{p-1}\}$ where

$$\phi(e_i) \leq \phi(e_{i+1}).$$

Now let e_j be the first edge in T_E which is not in T_0. Let $G_0 = T_0 + e_j$. Since T_0 is a spanning tree, G_0 must have a cycle C. Since T_E is a tree, there must be an edge e_0 in C which is not in T_E. In particular, $e_0 \in T_0$. We can consider the graph $T_0' = G_0 - e_0$, which has no cycles and is connected, so it is also a spanning tree. We notice that

$$\phi(T_0') = \phi(T_0) + \phi(e_j) - \phi(e_0).$$

Since T_0 is minimal, we have

$$\phi(T_0) \leq \phi(T_0') = \phi(T_0) + \phi(e_j) - \phi(e_0)$$

and hence

$$\phi(e_0) \leq \phi(e_j).$$

However, $\phi(e_j)$ was chosen to have minimal weight among edges in G not in T_E in the construction, which means we must have that

$$\phi(e_0) = \phi(e_j).$$

Thus

$$\phi(T_0') = \phi(T_0).$$

So T_0' is also minimal. We now consider T_0' instead of T_0. We may do the same construction as before, finding the first edge $e_{j'}$ which is in T_E but not T_0'. We note that $j' \geq j + 1$. We may continue to do this until we construct a minimal spanning tree which is equal to T_E. ■

See Example in Figure 4.11 in C.

Note, the algorithm of constructing the economy tree is called Kruskal's algorithm. Kruskal's algorithm can give a minimal spanning forest as well.

2 Shortest path problems and Dijkstra's algorithm

We consider the shortest path problem: Given a railway network connecting various towns, determine the shortest route between a given pair of towns.

Often, we will refer to the the weight of an edge as a length and the value of the smallest weight as the distance. We will present the algorithm of Dijkstra and Whiting-Hillier (found independently). In the sequel, we will assume that ϕ is defined on all pairs of vertices and $\phi(uv) = \infty$ if $uv \notin E(G)$.

Definition 7 *The* distance *between two vertices $u, v \in V(G)$ is equal to*

$$d(u,v) = d_G(u,v) = \min\{\phi(P) : P \text{ is a path from } u \text{ to } v\}.$$

A path P which attains the minimum is called a shortest path.

We then have the following algorithm, known as Dijkstra's algorithm:

1. Let $\ell(u_0) = 0$ and let $\ell(v) = \infty$ for all $v \neq u_0$. Let $S_0 = \{u_0\}$ and let $i = 0$.

2. For each $v \in S_i^c$, replace $\ell(v)$ with

$$\min_{u \in S_i} \{\ell(v), \ell(u) + \phi(uv)\}.$$

3. Compute M to be

$$M = \min_{v \in S_i^c} \{\ell(v)\}$$

and let u_{i+1} be the vertex which attains M.

4. Let $S_{i+1} = S_i \cup \{u_{i+1}\}$.

5. If $i = p - 1$, stop. If $i < p - 1$, then replace i with $i + 1$ and goto step 2.

Lemma 8 *If $v_0, v_1, \ldots, v_k$ is a shortest path, then $v_0, v_1, \ldots, v_j$ is a shortest path for any $j \leq k$.*

Proof. If there were a shorter path from v_0 to v_j, then we could replace the current path with a shorter beginning and get a shorter path to v_k. ■

Let's prove that at the termination of the algorithm, $\ell(u) = d(u, u_0)$. We will induct on i. Clearly, this is true for $i = 0$. We will make the following inductive hypothesis:

- For every $u \in S_i$, $\ell(u) = d(u, u_0)$.

We have the base case, so we need only prove the inductive step. Suppose it is true for S_i. We must show that

$$d(u_0, u_{i+1}) = \ell(u_{i+1}).$$

Let $P = v_0, v_1, v_2, \cdots, v_k$, where $v_0 = u_0$ and $v_k = u_{i+1}$, be a u_0u_{i+1}-path such that

$$d(u_0, u_{i+1}) = \phi(P).$$

If $v_{k-1} \in S_i$, then the path $P' = v_0, v_1, v_2, \cdots, v_{k-1}$ is a shortest path and by the inductive hypothesis $\phi(P') = \ell(v_{k-1})$. Thus

$$d(u_0, u_{i+1}) = \phi(P) = \ell(v_{k-1}) + \phi(v_{k-1}u_{i+1}) \geq \ell(u_{i+1})$$

but since $d(u_0, u_{i+1})$ is the minimum length path and $\ell(u_{i+1})$ is the length of some path, then we must have equality. Thus the inductive step is proven if $v_{k-1} \in S_i$.

We now show $v_{k-1} \in S_i$. Take the smallest j such that $v_j \notin S_i$. Then since $P_j = v_0, v_1, \ldots, v_j$ is a shortest path, we have, since $v_{j-1} \in S_i$, that

$$\ell(v_j) \leq \ell(v_{j-1}) + \phi(v_{j-1}v_j) = \phi(P_j) \leq \phi(P) \leq \ell(u_{i+1}).$$

since $\ell(u_{i+1}) = \min\{\ell(u) : u \in S_i^c\}$, that means that all of the inequalities are equalities and $j = k$ (since $P_j = P$) and $v_{k-1} \in S_i$. By the previous argument, we are done.

We do not discuss the complexity of this algorithm. It turns out to be a good algorithm.