Journal of Combinatorial Mathematics and Combinatorial Computing, 121: 67–70
DOI:10.61091/jcmcc121-07
http://www.combinatorialpress.com/jcmcc
Published 30 June 2024

Article

Finding Optimal Spanning Trees For Damaged Networks

John Hamilton[1,*], **and Hossein Shahmohamad**[2]

[1] School of Mathematical Sciences
[2] Rochester Institute of Technology, Rochester, NY 14623

* **Correspondence:** jfhsms@rit.edu

Abstract: We use a representation for the spanning tree where a parent function maps non-root vertices to vertices. Two spanning trees are defined to be adjacent if their function representations differ at exactly one vertex. Given a graph G, we show that the graph H with all spanning trees of G as vertices and any two vertices being adjacent iff their parent functions differ at exactly one vertex is connected.

Keywords: Graph, Spanning tree

---

1. History

The history of the minimum spanning trees can be found in many papers. In [1] details of all the classical algorithms related to the minimal spanning tree problem are analayzed. In [2] the asymptotical complexity of the methods used to solve different types of optimum undirected tree problem are discussed. In [3] an insight to the algorithmic technique for solving the minimal spanning tree problem is provided. In [4] various computational methods for MST algorithms are discussed. Non-greedy approaches for MST are found in [5]. In [6] a survey of different computational experiments is given. A nice recent survey is [7].

In this paper, we follow the notation in [8]. Many of the ideas discussed here can be found in the two excellent sources [8] and [9]. Let G be a simple, connected, and rooted graph with a vertex set $V(G)$, an edge set $E(G)$, and a root vertex v^*. We let $|V|$ and $|E|$ denote the number of vertices and edges, respectively, and we let V* denote the set of non-root vertices.

2. Motivation

The motivation to study this came from working on computer networks for the first author. He was developing a network protocol that would recover very quickly if a link or node were to fail. The idea was that a sub-optimal spanning tree might be found quickly and then migrate, one step at a time, to an optimal spanning tree. At every step the network would still have a functional spanning tree. In order for the migration to be possible, the collection of spanning tree for the damaged network had to be connected.

A spanning tree T of G is a subgraph of G containing no cycles and having $|V| - 1$ edges. For more on spanning trees see [10]. We will let T denote both the spanning tree itself and its edge set. While G and T are undirected graphs, it is useful to think the edges of T as having an orientation which points in the direction of v^* along a path in T.

For any vertex v in $V(G)$, we let $N(v) = \{w \in V | (v, w)$ is an edge in $E\}$. We say $N(v)$ is the set of nearest neighbors of v. Each edge e in $E(G)$ has an associated positive cost. Any path in G has an associated cost which is equal to the sum of its edge costs. For any vertex v in $V(G)$, there is a shortest path cost, denoted by $c(v)$, which is the minimum path cost over all paths in G between v and v^*. It follows that $c(v^*)$ is zero and that, for any other vertex v, $c(v)$ is positive. A shortest path spanning tree T' has the property that for any vertex v, the cost $c(v)$ is equal to the cost of the (only) path in T' between v and v^*. It follows that if w is another vertex on the path in T' between v and v^*, then $c(w) < c(v)$. A graph G always has a shortest path spanning tree and in general it is not unique.

3. A Few Observations

For any spanning tree T, and for any vertex v in V^*, there is a unique path in T, having at least one edge, between v and v^*. It follows that there is a unique vertex $P(v)$, called the parent of v, which is a nearest neighbor of v and is closer to v^* along the same path. The root vertex v^* has no parent, but it may be the parent of another vertex. The spanning tree T is completely described by its parent function $P_T : V^* \to V$. Note that an arbitrarily defined parent function may not correspond to a valid spanning tree (e.g. arbitrary $P(v)$ might not be a nearest neighbor. For future reference, we make the following observations here:

1. Let T be a spanning tree of G with parent function P.
2. Let T' be a shortest path spanning tree of G with parent function P'. Since path-cost on T' is identical to path-cost on G, then it follows that $c(P'(v)) < c(v)$ for any v in V^*.
3. Let the set V of all vertices be sorted in order of increasing path cost. Thus, we have: $V = \{v_k | 0 \leq k \leq |V| - 1$ and if $i < j$ then $c(v_i) \leq c(v_j)\}$. The lowest cost vertex is $v_0 = v^*$ and $c(v_0) = 0$.
4. Let $A(k) = \{v_j | 1 \leq j < k\}$ and $B(k) = \{v_j | k \leq j < |V|\}$ for $1 \leq k \leq |V|$. Now $A(1) = \emptyset, A(|V|) = V^*, B(1) = V^*, B(|V|) = \emptyset$, and $A(k) \cap B(k) = \emptyset$ for any k.
5. Let $\{P''_k | 1 \leq k \leq |V|\}$ be a sequence of parent functions such that $P''_k = P'$ on $A(k)$ and $P''_k = P$ on $B(k)$.
6. Let $A^*(k) = \{v^*\} \cup A(k)$

We want to show that any parent function P corresponding to spanning tree T can be modified, one step at a time, to reach parent function P' corresponding to shortest path spanning tree T'. The modified parent functions are denoted by P'' and it is important to show that each of them corresponds to a spanning tree T'' (and doesn't generate a loop such as the example above). The machinery of the proof is to write the vertices of the graph in order of path cost; that means that v_0 is the root (cost=0), v_1 is the next low-cost vertex, etc until we come to $v_{|V^*|}$ which is the most path-costly vertex.

We define $A(k)$ and $B(k)$ wherein all the vertices in $A(k)$ are less path-costly than all the vertices in $B(k)$. As the value of k goes from 1 to $|V|$, we create the sequence of parent functions P''_k such that the sequence starts with function P and ends with function P'. We argue that each P''_k corresponds to a spanning tree T''_k. In this sequence of modified parent functions, P''_k is followed by P''_{k+1}. We show that these functions differ only at vertex v_k which means they are distance apart and hence adjacent and connected. The corresponding spanning trees are T''_k and T''_{k+1} , respectively.

The last parent function in the migration sequence is $P''_{|V|}$ which equals P', the desired goal. The value k acts like a slider from 1 to $|V|$: $A(k)$ and $B(k)$ form a partition of the vertices. At $k = 1$, $A(1)$ is empty and $B(1)$ is the entire list of vertices and P''_1 is P. At $k = |V|$, $A(|V|)$ is the entire list of vertices and $B(|V|)$ is empty and $P''_{|V|}$ is P', the goal.

Lemma 1. *If parent function P''_k corresponds to a spanning tree T''_k, then parent function P''_{k+1} also corresponds to a spanning tree T''_{k+1} .*

Proof. Consider (1), (2), (3), (4), (5), and (6). Vertex v_k is in $B(k)$, therefore its parent is $P''_k(v_k) = P(v_k)$. Now consider the edge that connects vertices v_k and $P(v_k)$. If this edge is deleted from the spanning tree T''_k, then T''_k is disconnected into two components. One component, denoted by L_k, is the "lower" subtree having vertex v_k as its root. The other component, denoted by U_k, is the "upper" subtree, $T''_k - L_k$, which has v^* as its root. We now shift to parent function P''_{k+1} by changing the parent of vertex v_k to $P'(v_k)$. We have to show that, by including the new edge $(v_k, P'(v_k))$, the resulting structure T''_{k+1} is a spanning tree.

Because of (2), it follows that $c(P'(v)) < c(v)$. However, the vertices are sorted by the path-cost and v_k has the smallest path-cost of nay vertex in $B(k)$. Thus, it must be that $P'(v_k)$ is in $A^*(k)$. Also, because of (2) and (3), repeated applications of P' to any vertex in $A^*(k)$ must terminate at v^*. Therefore, all the vertices in $A^*(k)$ are connected to V^* and $A^*(k)$ is in the upper subtree U_k.

Clearly, v_k is in the lower subtree L_k, so this means that by adding the new edge, the disconnected components U_k and L_k are reconnected. Regarding cycles, there is no way that a single edge connecting two disjoint sets can be part of a cycle. The two components were already trees in their own right so neither had internal cycles. Thus, T''_{k+1} is a (connected) tree containing all vertices, i.e. a spanning tree. □

4. Main result

Now imagine that spanning trees are abstracted as vertices themselves in a graph H of the spanning trees of G. Then we could say that two spanning trees T and T' are adjacent if and only if their parent functions P and P' differ at exactly one vertex. We would then say that T and T' are distance one apart. Notice that from (5), distance $(P''_i, P''_j) \leq |i - j|$. Now we ask whether H is also connected or not.

Theorem 1. *The graph H is connected.*

Proof. Consider (1), (2), (3), (4), (5), and (6). The first function in the sequence of parent functions is P''_1. Because $A(1) = \emptyset$ and $B(1) = V^*$, this function is identical to P which is given as corresponding to a spanning tree T. Therefore, by repeated application of Lemma 3.1, every parent function in the sequence corresponds to a spanning tree. The last function in the sequence of parent functions is $P''_{|V|}$. Because $A(|V|) = V^*$ and $B(|V|) = \emptyset$, this function is identical to P' which corresponds to the shortest path spanning tree T'. Thus, spanning trees T and T' are connected. But, T is arbitrary, so any two arbitrary spanning trees are connected to the same shortest path tree T' which means they are connected to each other. Therefore, the graph H is connected. This completes the proof. □

If we now consider H', the subgraph of H containing only shortest path spanning trees of G, we can have the following immediate consequence:

Corollary 1. *The graph H' is connected.*

As a future work, we pose the following question: Can arbitrary spanning trees of G be path connected without using shortest path spanning tree, i.e., is the graph $H^* = H - H'$ connected?

References

1.Pierce, A. R., 1975. Bibliography on algorithms for shortest path, shortest spanning tree, and related circuit routing problems (1956-1974). *Networks, 5*, pp.129-149.

2.Maffioli, F., 1981. Complexity of optimum undirected tree problems: A survey of recent results. In *Analysis and design of algorithms in combinatorial optimization* (pp. 107-128). Springer.

3.Graham, R. L. and Hell, P., 1985. On the history of the minimum spanning tree problem. *Annals of the History of Computing, 7*, pp.43-57.

4.Haymond, R. E., Jarvis, J. P. and Shier, D. R., 1984. Computational methods for minimum spanning tree algorithms. *SIAM Journal on Scientific and Statistical Computing, 5*(1), pp.157-174.

5.Glover, F., Klingman, D., Krishnan, R. and Padman, A., 1992. An in-depth empirical investigation of non-greedy approaches for the minimum spanning tree problem. *European Journal of Operational Research, 56*, pp.343-356.

6.Moret, B. M. E. and Shapiro, D., 1991. How to find a minimum spanning tree in practice. In *New results and new trends in computer science: Proceedings, Graz, Austria, June 1991* (pp. 192-203). Springer.

7.Biswas, P., Goel, M., Negi, H. and Datta, M., 2016. An efficient greedy minimum spanning tree algorithm based on vertex associative cycle detection method. *Procedia Computer Science, 92*, pp.513-519.

8.Chartrand, G., Lesniak, L. and Zhang, P., 2016. *Graphs & digraphs* (6th ed.). CRC Press.

9.West, D. B., 2001. *Introduction to graph theory* (2nd ed.). Prentice Hall.

10.Biggs, N. L., 1974. *Algebraic graph theory* (2nd ed.). Cambridge University Press.