Determining the Shape of a Tree Network From Boundary Measurements

K. Lorentz Johnson *

June 19, 2003

Abstract

The main concern of this paper is the recovery of the shape of a resistor network from boundary measurements. In particular, we concern ourselves with the special case where the graph is a tree. Algorithms are given for recovering not only the shape of the graph, but also the conductivity, γ, of each resistor. Applying the shape recovery algorithm to non-tree graphs, it turns out, has the effect of recovering any tree-like "branches" on the outer part of the network.

1 Introduction

Formally, a resistor network Γ is a collection of nodes and edges– each edge connecting two nodes. A good physical interpretation is an electric circuit. Each node, p, has a number associated with it u(p) which can be thought of as the voltage at that node. Each edge also has a number associated with it $\gamma(pq)$ which can be thought of as the conductivity of the resistor(edge). The number of edges incident to a node is the degree or valence of the node, which can be any number. The nodes are divided into two groups: interior nodes and boundary nodes. We will denote the set of boundary nodes $\partial(\Gamma)$. At the boundary nodes we are allowed to make measurements. A typical measurement consists of applying voltages to the boundary nodes and measuring the resulting currents at each boundary node.

*e-mail: johnsonk@veblen.carleton.edu

This is the forward problem: Given a potential function for the boundary, determine the current flow. The problem can be solved by making use of Kirchoff's law, which states that the net current flow out of an interior node is 0, and some linear algebra.

If we order the boundary nodes, and consider a voltage vector, ϕ and a current vector, ψ, then there is a matrix, Λ, which satisfies the equation $\Lambda\phi = \psi$. Λ can be found by performing row reduction on a matrix constructed from the conductivity function $\gamma(pq)$. This matrix is also straightforward to obtain from boundary measurements. To determine the columns of Λ, place a voltage of 1 at boundary node n. The resulting current vector corresponds to the n^{th} column of Λ. In [1], it is shown that Λ will be a symmetric matrix.

A typical inverse problem is: Given the shape of the network, and the Λ-matrix, find $\gamma(pq)$ for each resistor in the network. The inverse problem that we consider here is more challenging: Given the Λ-matrix, find not only $\gamma(pq)$, but also the shape of the network. "Shape", here, is meant in the topological sense. We want to find the number of nodes and edges and find all the connections between nodes and edges. We will begin by limiting our recovery attempts to finding graphs of a special type–"tree" graphs.

2 Tree Networks

The tree networks considered can have nodes of any degree greater than or equal to 3. Each node at the end of a branch (nodes of degree 1) is considered a boundary node, while all the branch points are interior nodes. The precise definition to be used in this paper is:

Definition 1. *A tree is any finite network which could be constructed as follows: Start with an interior node and attach at least three edges to it. At the other end of these edges, add either:*

(a) a boundary node

or

(b) a new interior node.

If a new interior node is added, at least two more edges must be attached to this node. Again, at the end of these two new edges put either (a) or (b). If a boundary node is added, then do nothing more to this branch. Repeat until all the branches have reached boundary nodes.

Theorem 2. *A network is a tree network if and only if the following conditions are met:*

(1) There are no closed loops inside the network. That is, there exists only one path which connects any two nodes.

(2) All boundary nodes are boundary spikes. That is, all boundary nodes are only connected to one edge.

(3) The degree of any interior node is at least three.

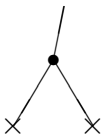
(4) Every boundary node is connected through the interior to every other boundary node. (1-connectedness)

NOTE: Condition (3) ensures us that we will not have any two resistors in series (a non-recoverable feature: See section 4).

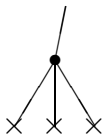
Proof. Constructing the network as in the definition ensures that we will have no closed loops (condition 1). Because we keep adding new nodes at the end of each new edge, the network can not close up on itself. Boundary nodes can only be added at the ends of edges so they will always be spikes (condition 2). Whenever we add a new interior node, we also add at least two new edges, so the degree must always be three (condition 3). Finally, the network was constructed so that every boundary node is connected to every other boundary node through the interior (condition 4).

Now, we start with a network which satisfies the 4 conditions. Choose an interior node as our starting point. There must be at least three edges attached by condition 3. By condition 1, these branches must always lead to either new interior nodes or boundary nodes. By condition 2, if we reach a boundary node, there can be no further edges attached. If we lead to a new interior node there must always be at least two new edges attached by condition 3. Finally, by condition 4, we must be able to construct the entire network in this fashion. □

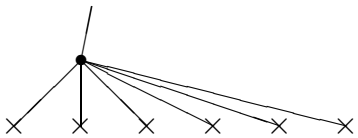
The definition of a tree network gives rise to the concept of pairings. If we reached a point in our construction where we added only two new edges to an interior node, and placed boundary nodes at the ends of these edges, we would have a special relationship between these nodes nodes. Precisely, two boundary nodes of a tree network are said to be *paired* if they are both connected to an interior node of valence 3 (as shown below).



There is also a notion analogous to pairing for an interior node of higher degree. If an interior node has degree (n+1), and n of its edges connect to boundary nodes, then this set of boundary nodes is said to be n-tupled.



3-tupled
boundary nodes



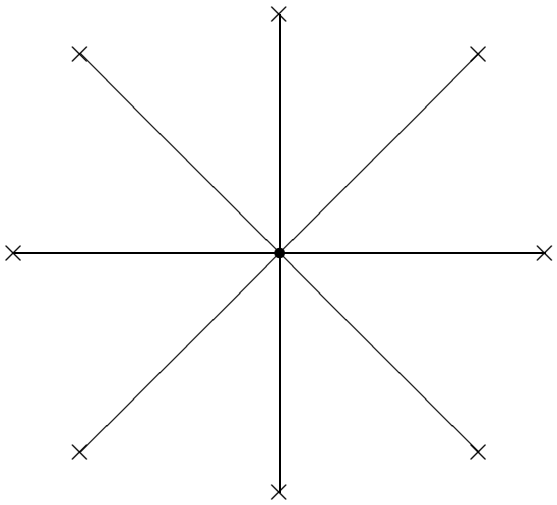
Set of 6-tupled
boundary nodes

NOTE: A pairing is an n-tupling with n=2.

Lemma 3. *If a network is a tree, then there is an $n \geq 2$ for which there exists a set of n boundary nodes that is n-tupled.*

Proof. Due to the way a tree can be constructed, when we add a boundary node, if all the other nodes added to the same interior node are either all boundary nodes, or else at least one is an interior node. If they are all boundary nodes, then we have an n-tupling. If one of the other nodes is an interior node, we repeat our process of looking at the nodes added to this interior node. Since our network is (assumed to be) finite, there must be some point at which no new interior nodes are added, and we add only boundary nodes. Therefore, there will always be an n-tuple. □

Definition 4. *A spider is a tree graph with only one interior node. All of the boundary nodes, then, are connected directly to this interior node.*



Spider with 8 boundary nodes

Note that a spider must have at least 3 boundary nodes. Note also that when constructing a tree, if we choose to add all boundary nodes at the first step, we will have constructed a spider. So, a spider is a very basic type of tree graph.

3 Connectivity

An important tool in recovering the shape of a network is that of connectivity. Two boundary nodes are connected through the interior if they can be connected by a path which doesn't go through any other boundary nodes. We can extend this concept to a set of n nodes being connected to another set of n nodes. In the case of n=2 for example, two boundary nodes p_1, p_2 are said to be connected through the interior to two other boundary nodes q_1, q_2 if and only if there is a path between p_1 and q_1 disjoint from a path between p_2 and q_2. The existence of this connection is denoted $(p_1, p_2; q_1, q_2)$. For a resistor network, Γ, the set $\pi(\Gamma)$ is defined as the set of all (P;Q) which are connected where P and Q are sets of any size. (P has the same size as Q)

It is shown in [1], that if and only if $(p_1, p_2; q_1, q_2) \notin \pi(\Gamma)$, then the 2×2

determinant of Λ (of rows p_1 and p_2, columns q_1 and q_2) is 0. This condition holds for n $\times$ n determinants for any n. So, by checking determinants of Λ, we can find $\pi(\Gamma)$. When recovering the shape of a tree network, it turns out that all we need to know is the set $\pi(\Gamma)$. The rest of the information in Λ concerns conductivity values.

4 Equivalency

One inherent limitation to accurate recovery of a network from boundary measurements is due to equivalency. Certain combinations of resistors will respond to boundary measurements the same way as other resistor setups. The problem is that two different networks may have the same set $\pi(\Gamma)$. These networks are then said to be in the same equivalence class. A simple example is that two resistors in series, each with conductivity 1, will act the same as a single resistor with conductivity 1/2. So, we could transform a network by replacing two resistors in series with a single resistor. If we choose the right conductivity for our new resistor, there will be no effect on boundary measurements. We can not tell the two networks apart.

In [2], it is shown that there are 6 such transformations that can be made to networks. Most of these, such as the example of resistors in series, arise from the existence of "superfluous edges" in a graph. A critical network is a network in which the contraction (elimination) of any edge will affect the set of connections. Therefore, if we assume that our network is critical, then every edge is important and there are no "superfluous" edges.

The one equivalence we can not eliminate by limiting ourselves to critical graphs is the Y - Δ equivalence. A Δ is a set of three nodes and three edges. Each node is at a vertice of a triangle, and the edges correspond to the sides of the triangle. There are no other nodes or edges inside the triangle. A Y is a set of four nodes and three edges. One node is in the center and has degree 3. Each of its edges connects to each of the other three nodes. These two formations will produce the same connections so we can not tell them apart from boundary measurements.

When attempting to recover a network, then, the best we can do is to find something in the same equivalence class as the original network. This does not present any ambiguity in the case of trees due to the following theorem.

Theorem 5. *A tree can not be transformed by Y-Δ and Δ-Y transformations*

into a different tree.

Proof. If we start with a tree, we have no closed loops, so there can not exist any Δ. The only transformation we can make initially is a Y-Δ. In order to end up with another tree, we have to somehow remove the Δ we have just added to the graph. We can either:

(a) Make another Y-Δ transformation to one of the corner nodes (a vertice of the Δ).

(b) Make a Δ-Y to undo what we did and return to the original graph

It will now be shown that (a) is impossible. Therefore, we can not come up with a new tree, by making Y-Δ and Δ-Y transformations.

The effect of the Y-Δ transformation on a corner node is to increase the degree by 1. Since we started with a tree, these nodes must have had degree at least 3, and now they have at least degree 4. In order to make a Y-Δ transformation, the center node must have degree 3, so the corner nodes can not be the center of another Y-Δ transformation. □

So, for tree graphs, recovering the equivalence class is as good as recovering the shape of the graph itself.

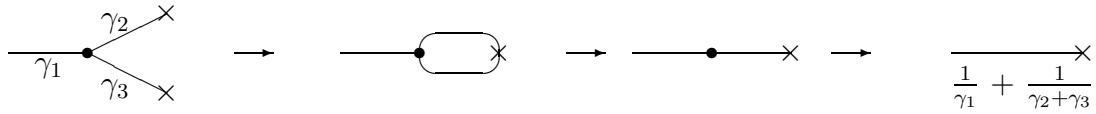
5 Recovery Algorithm

The method for recovery hinges on being able to find sets of boundary nodes that are n-tupled. A method for finding whether or not a set of nodes is n-tupled from $\pi(\Gamma)$ will be presented in the next section. For now, we will assume that this is possible.

When we find two nodes that are n-tupled, we will remove these n nodes along with the n edges connected to them. The resulting graph will still be a tree. Going back to our definition of a tree, we could have constructed the network the same way, but stopped a step earlier and placed a boundary node instead of an interior node and the n edges. Therefore, we still have a tree.

Fortunately, if we have an actual physical network, we do not have to rebuild the network or remove any resistors in order to apply the algorithm. When we find n-tupled nodes, we simply ground these nodes together so that from now on, the voltage at these nodes is always the same. We would then

have n resistors in parallel followed by one resistor in series. This can be treated as a single resistor with a conductivity which is a function of the original n+1 conductivities. The transformation will look like (for n=2):



Theorem 6. *In the* Λ*-matrix, if we add together the rows and columns corresponding to n n-tupled nodes, then use this new row and column to correspond to our new boundary node as described above, we can eliminate the n rows and columns corresponding to the deleted boundary nodes, and the resulting* Λ*-matrix will be correspond to the new network.*

Proof. We saw in section 1 that the nth column in the Λ-matrix corresponds to the current produced by a voltage of 1 placed at node n. If we placed voltages of 1 at n boundary nodes, the resulting current vector would just be the sum of these n columns in the Λ-matrix.

Due to the way in which we collapsed the n-tupled boundary nodes, applying a voltage of 1 at the new boundary in the new graph has the same effect as placing voltages of 1 at all n nodes in the old graph. So this is the same as placing voltages of 1 at the n boundary nodes, and the resulting current vector is the sum of these n columns in the Λ-matrix. The rows can also be added due to the symmetry of the Λ-matrix. □

After modifying the network, we repeat the process of searching for n-tuples until we cannot find any more n-tuples for n less than or equal to the number of boundary nodes minus two. We shall see that we will need at least two more boundary nodes in order to make any checks about a set being n-tupled(see next section).

6 Finding n-tuples

We will begin by showing how to determine whether or not two nodes are paired given $\pi(\Gamma)$.

Theorem 7. *Two nodes p_1, p_2 are paired nodes of a tree network, Γ, if and only if the following two conditions are met:*
(1) $(p_1, x; p_2, y) \in \pi(\Gamma)\ \forall \{x, y\} \in \{\partial(\Gamma) \setminus \{p_1, p_2\}\}, x \neq y$
(2) $(p_1, p_2; x, y) \notin \pi(\Gamma)\ \forall \{x, y\} \in \{\partial(\Gamma) \setminus \{p_1, p_2\}\}$

NOTE: Both conditions are necessary if we are dealing with a graph that may or may not be a tree. If we know the graph is a tree, either one of the conditions will suffice.

Proof. Given 2 paired nodes in a tree network p_1, p_2 we can connect p_1 to p_2 through their common interior node. This path, then, involves no other interior nodes. The common interior node has valence 3, so there is only 1 more edge incident to it. Therefore, there can not be a path between x and y that goes through this node. There must be some path between x and y by property 4 of Theorem 2.2. Therefore, this path must be disjoint from the path between p_1 and p_2.

Condition 2 is true because if we connect p_1 to some other node x, we have to go through the common interior node. Now, if we wish to connect p_2 to anything, we must also go through the common interior node, so there can not exist a disjoint path.

If condition 1 is true, then we must have a path between p_1 and p_2 that does not interfere with any other connections between boundary nodes. If we can show that there is one common node along this path from which all other paths branch, then we will know that p_1 and p_2 are paired.

Condition 2 shows that there must be such a common node. If p_1 is connected to some other node, then we can not simultaneously connect p_2 to any other nodes. This can only happen if p_1 and p_2 are paired. □

In section 3, we saw that we can determine the connections in Γ by taking determinants of Λ, so we can determine whether or not two nodes are paired from these determinants. Now, if we wish to determine whether a set of n nodes are n-tupled, we make a slight modification to our test for pairings.

Definition 8. *Two nodes p_1, p_2 are paired without respect to a set Q of boundary nodes, $\{q_1, q_2...\}$ if upon removal of Q and the edges connected to Q, p_1 and p_2 would be paired.*

The test for pairings without respect to Q then uses $\{x, y\}$ chosen from ∂ $(\Gamma) \setminus \{p_1, p_2, q_1, q_2...\}$. The following theorem gives us a way to find n-tuples.

Theorem 9. *In a tree network, a set P of n nodes $\{p_1, p_2...p_n\}$ is n-tupled if and only each node p_k is paired with the next node p_{k+1} for all $k \leq n$ without respect to the rest of the set($P \setminus \{p_k, p_{k+1}\}$.*

Proof. If we have n n-tupled nodes, it is straightforward to see that each node is paired with the next node without respect to the rest of the n-tupled nodes.

Going the other direction, we start with p_1 paired with p_2 in a network which doesn't contain $\{p_3, p_4...p_n\}$. In a network without $\{p_1, p_3, p_4...p_n\}$, p_2 and p_3 are paired. This can only occur if p_1, p_2, and p_3 are a 3-tuple in the network without $\{p_4, p_5...p_n\}$. Continuing in this fashion, we will find that $\{p_1, p_2...p_n\}$ is an n-tuple in the entire network. □

Now that we have the ability to test for n-tuplings, we are ready to state the recovery algorithm.

Algorithm 10. *We begin by checking for an n-tupling using Theorems 6.1 and 6.3. If we find n boundary nodes that are n-tupled, we replace these nodes with one boundary node. In the Λ-matrix we add together the rows and columns to produce a new row and column. If we do not know the Λ-matrix but are instead given the set $\pi(\Gamma)$, then we simply choose one of the old node's connection properties to represent the new node.*

After we find an n-tuple and have collapsed it into a single boundary spike, we continue to search for n-tuples. The algorithm proceeds until we can no longer find n-tuples. This means that we cannot find an n-tuple for n less than or equal to the number of boundary nodes minus 2.

7 Arbitrary Networks

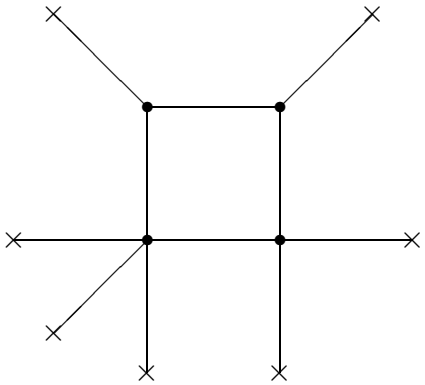
We now consider the algorithm as applied to arbitrarily shaped networks. The theorems in this section will show that we can use the theorem to determine whether or not a network is a tree. Also, in non-tree networks, we can recover any tree-like appendages on the outside part of the graph.

Theorem 11. *If our original network is a tree, then applying the algorithm to the network will produce a spider.*

Proof. Since the network is a tree, by Lemma 2.3, there will always be an n-tuple to remove until we reach a spider (which is also an n-tuple). A spider can not be removed because we do not have enough boundary nodes to continue to check. Also, considering a tree as it was defined, removing n-tuples will just reverse the construction process, until we get back to the point of having a spider. □

We can easily check if the resulting network is a spider. In a spider, each boundary node is connected through the interior to every other boundary node. However, there are no sets of 2 or more nodes connected through the interior. The Λ-matrix for a spider is rather special.

One complication that arises from applying the algorithm to non-tree graphs is that we must modify our concept of n-tupling. If n nodes are connected directly to an interior node which is part of a closed loop, then these nodes might be n-tupled. They are n-tupled if we make sure we are including all such boundary nodes.



We now allow 3-tuples as in the lower left corner,
and pairings as in the lower right corner

The algorithm will find these n-tuples, and we can remove them the same way as we did for tree graphs. We will not, however, remove any closed loops out of the graph. This, in essence, is the proof of the following theorem.

Theorem 12. *Applying the algorithm to a non-tree graph will not produce a spider.*

Proof. The algorithm will find n-tuples on the outer parts of the graph. It will also find n-tuples of the sort described above. So, eventually, we will have our network reduced to just a closed polygon with boundary nodes at the corners, or on the ends of edges attached to the corners. If we choose any two adjacent boundary nodes, we can always connect them simultaneously to two other boundary nodes by going different directions around the polygon. Therefore condition 2 of our test for pairings will always fail. No nodes will be removed and the closed loop will never be removed from the graph. We can not, then, produce a spider since a spider has no closed loop in it. □

8 Recovering Conductivities

Once we know the shape of a tree graph, we can recover the conductivities. If we have two boundary nodes that are paired (or two out of a set that is n-tupled), we can apply a voltage of 1 at one node, and a voltage of -α at the

other node. Apply voltages of 0 at all other boundary nodes. Choose α so that the current will be 0 at all boundary nodes besides the paired ones. All the current flow will be through the two boundary spikes. From the value of this current (which we are allowed to measure), and the value of α that we chose, we can find the conductivities of the two boundary spikes.

We can then ground the two boundary nodes together and treat the complex resistor as a single resistor as we did in section 5. We already know the values of two of the three conductors, so finding the value of the complex conductor will give us enough information to calculate the conductivity of the third resistor. We can thus recover the conductivity function over the entire network by proceeding in this manner.

9 Further Problems

Another way to approach the problem of tree recovery is through the use of dual graphs. The concept of a dual graph is precisely defined in [1]. The convenient thing about the dual graph of a tree is that it will have no interior nodes. Since there is a way of relating the Λ-matrix for a network to the Λ-matrix for it's dual, we could attempt to recover the shape of a network by recovering the shape of it's dual graph.

Other questions that arise involve finding other structures in a network that could be easily recovered from boundary measurements. Networks that are close to being trees might be recoverable by methods similar to those described in this paper. Hopefully, one would eventually be able to recover, within equivalence classes, the shape of any arbitrary network.

References

[1] E. B. Curtis, D. Ingerman, J. A. Morrow, *Circular planar graphs and resistor networks*.

[2] Y. Colin de Verdière, I. Gitler, D. Vertigan, *Rèseaux Èlectriques Planaires II*, Prèpublication de l'Institute fourier, no. 276, (1994).