Analysis and Efficiency of Jeffrey Giansiracusa's Algorithm for Partial Recovery

Michael Burr

Abstract

I present the description of an algorithm, as devised by Giansiracusa, to compute the partial recovery of a conductivity network. The algorithm is a description of how this could be implemented on a computer.

1 Definitions

Definition 1. Given a medial graph m, let the set of lenses and their interiors be given by $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$. Then a lens, ℓ_i is said to be a *lensless lens* if $\ell_i \cup \ell_j = \ell_i \Rightarrow \ i = j$

Definition 2. Given a lensless lense, a *strand* is a portion of a geodesic which connects one side of the lens to the other. Note that if a geodesic enters and exits a lens twice, the two portions of the geodesic interior to the lens form two strands, not one.

Definition 3. The *inversion number* of a permutation is the number of pairs, (i, j), where $i < j$ and i lies to the right of j. For example, the permutation $(1, 4, 3, 2)$ has inversion number 3 because both 3 and 2 lie to the right of 4 and 2 lies to the right of 3. The inversion number is equivalent to the minimum number of inversions of adjacent elements needed to change the permutation into the permutation $(1, \ldots, n)$.

Definition 4. A *loop* is a path along a geodesic which intersects itself.

Definition 5. A *bubble* is a geodesic which is homeomorphic to a circle. It is a geodesic with no ends

Definition 6. A *vertex of a lens* is a crossing of the geodesic(s) which form(s) the lens. A lens may have zero, one, or two verticies.

2 Data Structures

2.1 List node

Data: Stores data, and next and previous fields.
Access Functions: Set functions for data, next, and previous, and get functions for data, next, and previous.

2.2 Circularly Doubly linked list

Data: Stores data for the first, last, and current values of a list node.
Access Functions: Append function which adds a new node with the specified data in the node. Delete function, which deletes the current node. Current function, which returns the data in node specified as current. Previous and next functions which change the current node to the previous or next node and returns the data in the previous or next node, respectively. First and last functions, which return the data in the first or last node and sets current to first or last, respectively. Set-first and a set-last function to set the first and last elements respectively. Finally, a set function to set the current node to a specific node. If this last feature is used effectively, the algorithm will run much faster.

2.3 Edges

Data: Stores its two endpoints in no particular order, and the geodesics which intersect on this edge
Access Functions: Set functions for each of the endpoints and geodesics and get functions for each of the endpoints and geodesics.

2.4 Points

Data: Stores a circularly linked list which has the list of edges incident to the point in clockwise order, as well as if the node is a boundary node or an interior node.
Access Functions: Set function, which sets a point's list to a certain list, and a get function, which returns a point's list of edges, in addition to set and get functions for whether the node is a boundary node or an interior node.

2.5 Geodesics

Data: For a each segment of a geodesic, this value stores the edge which the geodesic intersects and the other geodesic at the intersection. Also included is a field which will be used to color the geodesic later.
Access Functions: Set and get functions for the edge of intersection and the intersection geodesic. Set and get functions for the coloring as well.

For the remainder of this paper, these data functions will not be discussed, but these along with arrays and other typical data structures will be enough to program the algorithm described in following sections.

3 Input

The input is given in the following form:

$$
\begin{aligned}
&n, k \\
&1 : e_{1,1}(a_{1,1}), \ldots, e_{1,j_1}(a_{1,j_1}) \\
&\vdots \\
&n : e_{n,1}(a_{n,1}), \ldots, e_{n,j_n}(a_{n,j_n})
\end{aligned}
$$

Where n is the number of nodes, k is the number of boundary nodes ($k \leq n$), "#:" corresponds to the list of edges out of node "#", $e_{r,s}$ corresponds to and edge between nodes r and r_s in clockwise order. Finally, if there are multiple edges in parallel between two nodes, then the $(a_{t,u})$ term differentiates between the two edges. The $a_{t,u}$ can be omitted when there is only one edge between two nodes.

4 Output

The output will be the final graph in the exact same form as the input form along with a list of transformations from the original graph to the final graph describing the type of transformation and the edges involved.

5 Algorithm

1. The first step is the construction of the graph itself using the data structures described above. There should be n points, where the circularly linked list comprising its data function contains the edges which are incident to the given point in clockwise order. The edges, similarly contain their endpoints. Additionally, an aditional spacer will be placed into the list of the edges around a boundary vertex, after the last edge and before the first edge as a spacer for creating geodesics. It would also be helpful to incude a spacer after the last element and before the first element of the lists for a boundary node.

2. Next, we need to construct the medial graph. To do this, it will be easiest to keep three sets of edges: the set of edges which have not contributed to the medial graph, the set of edges which have contributed once to the medial graph, and the set of edges which have contributed twice to the medial graph. For each boundary node, v_i, consider the geodesics which begin at this node. Construct the geodesic which intersects the first edge in the clockwise edge list of v_i. Let the current node be v_i and then the first edge of the geodesic is the first edge in the list of edges around v_i, let this edge be e_{i_1}. Now, move this edge from the set it is currently in to the set where the edge is used by an additional geodesic. Set the new current node to be the other endpoint of edge e_{i_1}, let this node be w_1. Now find the previous edge in the clockwise edge list from e_{i_1}, let this edge be e_{i_2}, move this edge to the set where it is used once more. Now, let the other endpoint of the edge e_{i_2} be w_2, and continue this algorithm, alternating clockwise and counterclockwise rotations. Once this has been done for all boundary nodes, both for their first and last edges, if there are any edges which have not been used in two geodesics, they are part of bubbles. Starting at one of the edges which has this property and continuing from there, we can get the bubbles (the vertex to start at is the one where there are two consecutive edges which are not part of two geodesics.

3. After we've created the graph itself, we need to find a lens with the minimum number of interior regions. This takes 3 steps, first finding the lenses themselves, restricting this set to a set of possible minimal lenses, and actually finding the minimal lens.

   (a) The first task is to find the lenses. The easiest way to do this is to keep a set of lenses. First, all the bubbles are in this set, as they form lenses. Secondly, all loops should be in this set as well, and finally, all typical geodesics consisting of two intersections of two geodesics should be in this set. Now, comes the task of finding them. First, the bubbles are easily found, as they are constructed in step 2. Secondly, loops are easily found, as

they are self intersections with the geodesics and can be found by walking the length of a geodesic, finding the pairs of self intersections. Some care must be taken in case a geodesic intersects itself in more than one place, but that can be taken care of by coloring the self intersections. Finally, the intersections of two geodesics twice can be found by walking the length of a geodesic, while keeping track of the most recent intersections with each of the other geodesics, and whenever a double intersection occurs, that is a lens.

(b) As the algorithm depends on finding a lens with the minimum number of cells, by Proposition 2, we can restrict this search to finding lensless lenses. To do this, we will, for each lens, color its boundary and we can determine if a lens is lensless by examining all the geodesics interior to a lens. A lens is lensless iff at every node, either there is no coloring, or if it if colored, then the coloring also crosses the boundary of the lens at some point, by Proposition 3. From this information, we can extract the set of lensless lenses.

(c) Now, we need to determine which of the lensless lenses has a minimum number of interior regions. To do this, we walk along one edge of a lens and for each strand, we color each endpoint of the strand the same color. Now, we just need to consider the order of the colors on both ends and calculate the inversion number. Then the number of cells in a lens is $k + 1 + I(\ell_i)$, where I is the inversion number of a specific lens, ℓ_i, and k is the number of strands across the lens.

4. Now, if the lens is not empth the next thing to do is to start emptying the lens with the minimum number of cells. To do this we need to do two things. If the strand which intersects the boundary of the lens the closest to a vertex does not make any intersections with other strands, then do a $Y - \Delta$ transformation with the three edges at the intersection of the strand with the boundary of the lens and the vertex closest to the strand. Then there is one less region in the lens we are considering, and as this is a local change, in all lenses, at most one region has been added or subtracted from the interior. Thus the lens under consideration is still a minimal lens. So that we can continue from there. If the strand closest to the vertex does intersect another strand, then walk along the edge of the lens until you find a pair of neighboring geodesics which intersect each other, and then moving their crossing outside of the lens by a $Y - \Delta$ transformation, similarilly to above. Repeat this step until the lens has been emptied. Report the changes which are made at each step.

5. If a lens has only one region inside of it, then open up a vertex of the lens, by uncrossing the geodesic(s) which form(s) the vertex. Thus reducing the number of lenses in the graph by one. Report the changes made each time a lens is opened up, which corresponds to removing a series or parellel connection.

6. Repeat steps 3-5 until there are no lenses.

7. Report the final graph.

6 Analysis, Correctness, and Termination

6.1 Analysis

1. The first step takes a maximum of $O(n * m \log m)$ time and $O(m)$ space, where m is the total number of edges in the graph. Let m_i be the degree of a vertex, so $\sum_{i=1}^{n} m_i = 2m$,

and $\forall m_i \leq m$. It takes $O(1)$ time to create each edge and each edge takes $O(1)$ space to store. As each set of edges may need to be sorted for each node, this takes $O(m_i \log m_i)$ for each node, as there are n nodes, so this takes $O(\sum_{i=1}^{n} m_i \log m_i) \leq O(\sum_{i=1}^{n} m \log m) = O((m \log m) \sum_{i=1}^{n} 1) = O(n * m \log m)$.

2. The second step takes a maximum of $O(m)$ time and $O(m)$ space as each edge is visited only once, and each edge is part of two portions of geodesics.

3. (a) This part of step 3 takes $O(m)$ time and space as it requires walking down the length of each geodesic (which adds up to visiting each edge of the original graph twice) and keeping track of all other geodesics, of which there are at most $O(m)$.
   (b) This step takes a bit longer to complete, as there are at most $O(m)$ lenses and there are at most $O(m)$ intersections within a lens. Thus this step takes $O(m^2)$ time and $O(m)$ space. This bound could most likely be tighter.
   (c) Finally, this step takes $O(m^2)$ as the colorings of the endpoints of the strands can be done in the previous step and once that is done, all that needs to be done is to walk down the boundary of a lens, which takes $O(m)$ time for $O(m)$ lenses. This bound as well could most likely be tighter.

4. This step takes a well $O(m)$ time, if one keeps track of the strands inside a lens carefully, and uses previous information to make each step amortized $O(1)$.

5. This step takes $O(1)$ time as we are just opening a lens.

6. These steps need to be run through $O(m)$ times, as there are at most $O(m)$ lenses.

7. This takes $O(m)$ as each edge is reported twice.

6.2 Correctness

Observation 1. Given a medial graph, let the set of lenses (with their interiors) of this medial graphs be $\{\ell_1, \ldots, \ell_n\}$. Let $1 \leq i \leq n$. If $\exists j \neq i$ where $\ell_j \subseteq \ell_i$, then ℓ_i is not a lensless lens and ℓ_j has strictly fewer regions in its bounded interior than ℓ_i.

Proposition 1. At no point during the process will there be an empty bubble

Proof. An empty bubble corresponds to a disconnected graph, and as $Y - \Delta$ transforms and removind series and parellel connections do not change the connectivity of the graph, an empty bubble will not occur. □

Note that this means that, by the Jordan curve theorem, every bubble will form a lens with another geodesic.

Proposition 2. The number of regions interior to a lensless lens, ℓ_i, is $k+1+I(\ell_i)$, where I is the inversion number, and k is the number of strands which cross the lens.

Proof. First, if the lens we are considerig is a bubble, then no geodesics can enter or exit it because then the bubble would not be lensless, so a bubble has one interior region. Moreover, this also implies that $k = I(\ell_i) = 0$, so $k + 1 + I(\ell_i) = 1$. Secondly, If the lens we are considering is a loop, then by the same reasoning, the loop must also be empty, so there is only one region inside the loop. Additionally the lack of strands implies that $k = I(\ell_i) = 0$, so $k+1+I(\ell_i) = 1$. Finally, if the

lens we are considering consists of the intersection of two geodesics, there can be geodesics which enter the lens, but the must leave through the bounding geodesic which they did not enter from as otherwise they would form a lens and the lens we are considering would not me lensless. This implies that the regions interior to the lens are divided up by strands. Moreover, any two strands can only intersect once as otherwise they would form a lens interior to the lens being considered. Let $n = I(\ell_i)$, then if $n = 0$, then it is easy to see that none of the strands intersect so the interior of the lens is divided into $k + 1$ regions. Now, let the proposition be true for $n = h - 1$, let $n = h$. Then starting at one bounding geodesic of the lens and moving to the other bounding geodesic, there must be one crossing which occurs last. Then whithout this crossing, the lens is divided into $k + 1 + h - 1 = k + h$ regions. Now if we now cross these two remaining strands, they divide the region between them into two regions and thus partition the lens into $k+h+1$ different regions. □

Proposition 3. A lens ℓ_i is lensless iff every lens, ℓ_j where a proper subset of the interior of ℓ_j intersects a proper subset of the interior of ℓ_i, has a transverse crossing of the boundaries.

Proof. First, if there are no other lenses whose interiors intersect the interior of ℓ_i, then ℓ_i is clearly lensless. Now, assume that $\forall j \neq i$ such that the interior of ℓ_j intersects the interior of ℓ_i, there exists a transverse crossing of the boundary of ℓ_i, then ℓ_i is lensless once again, as a transverse crossing implies that some part of the lens ℓ_j is external to ℓ_i, and thus ℓ_i is lensless. Finally, if $\exists j \neq i$ such that the interior of ℓ_j intersects the interior of $'ell_i$, but there is no transverse crossing of the boundaries. Assume that ℓ_i is lensless. Note that $\ell_i \not\subset \ell_j$ as ℓ_i and ℓ_j intersect on a proper subset of the interior of ℓ_i. Moreover, this implies that part of the boundary of ℓ_j lies interior to ℓ_i. Additionally, as we are assuming that ℓ_i is lensless, then there must be some portion of ℓ_j which lies outside of ℓ_i. But then by the Jordan curve theorem, there must be a crossing of the boundaries, moreover, as two geodesics only intersect at a point, then this crossing must be a transverse crossing, which is a contradiction, so thus ℓ_i cannot be lensless. □

6.3 Termination

First, let us define an ordering on $\mathbb{Z}^2_{\geq 0}$, where $(a_1, a_2) > (b_1, b_2)$ iff $a_1 > b_1$ or $a_1 = b_1$ and $a_2 > b_2$. Let $\mathcal{M}$ be the set of all medial graphs, and let $\phi : \mathcal{M} \to \mathbb{Z}^2_{\geq 0}$, so that for a given medial graph, $m \in \mathcal{M}$, $\phi(m)_1$ is the number of lenses in the medial graph and $\phi(m)_2$ is the minimal number of regions interior to a lens.

Proposition 4. Assume the algorithm on steps 3-5 takes a medial graph m to m', then $\phi(m) > \phi(m')$

Proof. After each interation of step four, the number of regions interior to the minimal lens has been decreased by one, and thus $\phi(m) > \phi(m')$. After step five the new medial graph has at least one less lens, so $\phi(m) > \phi(m')$. □

It is important to note that the algorithm will terminate although step 5 requires that a lens has a crossing as there will never be an empty bubble and opening a lens of a bubble will destroy the bubble.

References

[1] Giansiracusa, Jeffrey. "The Map L and Partial Recovery in Circular Planar Non-Critical Networks." 2002.

[2] Curtis, B., and James A. Morrow. “Inverse Problems for Electrical Networks.” Series on applied mathematics – Vol. 13. World Scientific, ©2000.