An Automated Spatial Flow Layout Algorithm using Triangulation, Approximate Steiner Tree, and Path Smoothing

Shipeng Sun

ABSTRACT: Flow data visualization is an important yet challenging topic in cartography as well as in scientific and information visualization. In recent years, significant progress has been achieved for spatial flow data visualization using techniques like spatial hierarchical clustering, edge biding, edge routing, directed forces, and spiral trees. This paper introduces a new algorithm to map the geographic flow data from one origin to many destinations, i.e., one-to-many flow data. The algorithm simulates the flow formation of the natural water systems and can capture the spatial distribution pattern of the destinations. It methodologically combines the Delaunay triangulation and approximate Steiner tree, followed by a path simplification and smoothing procedure. The spatial flow layout produced by the algorithm features smooth edges, natural spatial cluster, and fluent transition along the flow tree routes. The paper illustrates the usability of the algorithm with two exemplar maps.

KEYWORDS: Flow data, automated map layout, flow visualization

Introduction

Tracking movement across space and visualizing movement data have been a central piece in numerous aspects of the study of the natural world and the human society (e.g., Tobler 1987; Sun and Manson 2012; MacDonald et al. 2015). Flow data visualization is an important yet challenging topic in cartography as well as in scientific and information visualization (Andrienko et al. 2008; Andrienko and Andrienko 2013). In recent years, significant progress has been made for the spatial flow data visualization in general and one-to-many, i.e., from one origin to many destinations, in particular (Buchin, Speckmann, and Verbeek 2011b; Debiasi et al. 2014). New techniques applied in flow visualization include spatial hierarchical clustering, edge binding, edge routing, directed forces, and spiral trees (Zhou et al. 2013; Guo and Zhu 2014; Zhu and Guo 2014; Landesberger et al. 2016). Despite these advancements, visualizing spatial flow data remains a challenge for cartographers and computer scientists, particularly because many of these methods still need considerable manual calibration, optimization, and revision during and after applying computer algorithms.

Among many designs of the one-to-many flow visualization, the tree structure-based flow layout is one of the latest and most appealing (Phan et al. 2005; Buchin, Speckmann, and Verbeek 2011b). A flow tree layout is a single-origin, multiple-destination, and directed tree structure. It reaches out from the origin to all destinations with smooth trunks that split into branches at certain locations and terminate at the destinations. By binding edges into different levels of trunks and branches, the flow tree is simple, straightforward, and effective in terms of delineating the network flow direction and volume. While the tree-based flow map layout is visually appealing, it is difficult to construct compared with other types of layouts such as the one using multiple one-to-one direct links. More particularly, it is especially challenging to create and position the intermediate nodes between the origin and destinations that help produce smooth flow routes but simultaneously avoid line crossing and swamps of points or edges.

This paper proposes a new, automated algorithm to map the geographic flow data from one origin to many destinations. The algorithm simulates the flow formation in the natural water systems and can capture the spatial distribution pattern of the destination nodes. The automated flow map layout algorithm combines Delaunay triangulation and approximate Steiner tree and can largely keep edges from crossing. Based on cartographic generalization and network regionalization, a path simplification and smoothing subroutine is designed to improve the appearance of the final flow layout by deleting and moving some intermediate nodes and streamlining transitions from the origin, to trunks, to branches, and to destinations in the tree. As such, the spatial flow layout produced by the algorithm features smooth edges, natural spatial clusters, and fluent transitions along the flow tree paths. The paper illustrates the algorithm with two exemplar maps.

Related Works

Among the rich set of flow data visualization methods, one relatively small subcategory—spatial one-to-many flow layout—has regained attention in recent years (Buchin, Speckmann, and Verbeek 2011b; Debiasi et al. 2014). Such one-to-many flow data describe real-world flow of substances from one location to many other locations on the Earth with physical coordinates. Note that the case of "many-to-one" is technically the same as "one-to-many" in terms of layout construction. One-to-many flow data examples include maize exports from U.S. to other countries or regions, soybean imports of China, and population migration to the State of California from all other states in the U.S. This type of data is best represented with tree-style flow maps, where the root node of the tree is the origin location, the leaf nodes are destinations, the intermediate nodes define the shape of the paths, and all nodes are connected using different levels of trunks and braches with gradually varying colors and girths that are proportional to the flow volume.

Although geographers and cartographers, among others, had been making such flow maps for many years, only in recent years had computer scientist started to automate this process and produced esthetically pleasing flow layout maps (Long and Nelson 2013). This thread of research works was pioneered by Phan et. al (Phan et al. 2005). Unlike traditional flow maps that use many one-to-one direct links between the origin and destinations, either straight or curved, Phan's flow map uses only a single one-to-many tree to link all nodes together. Such a map essentially emulates flow systems in the natural world such as rivers. Using spiral trees, Buchin et. al. greatly improved the flow map layout (Buchin, Speckmann, and Verbeek 2011b, 2011a). Their works explored the application of Spiral and Steiner tree in one-to-many flow layouts. Because they designed their algorithm using the mathematical properties of these trees, the Spiral tree-based flow maps present a significant improvement in terms of map aesthetic quality. A supervised, force-directed algorithm was also developed to generate similar flow map layout using a strategy widely adopted in general graph plotting, where nodes attract or repulse each other based on physical laws that define the forces produced by nodes according to the differences between expected and actual distances (Debiasi et al. 2014).

Although automated map-making is generally appealing, cartography is a combination of science and art (Moenius 2012; Xiao and Armstrong 2012; Kent 2013). On one hand, an automatic cartographic algorithm should produce geometrically accurate maps with as high as possible aesthetic qualities. It should require little manual input or intervention from map-makers other than specifying spatial data and program parameters. On the other hand, an automatic algorithm should not deprive cartographers the option of manually improving the map design and its aesthetic quality. The algorithm should allow users editing the map with reasonable workload but still guarantee accuracy. In other words, manual intervention or revision is best provided as a not-required option to map-makers in cartographic algorithms. Existing one-to-many flow map methods vary significantly in terms of these two aspects. Some cannot produce high quality maps (Phan et al. 2005). Some requires considerable manual intervention during the map producing process (Debiasi et al. 2014). And most do not provide an intuitive and easy-to-use flow layout editing option.

Specific to the flow map quality, a few basic geometrical and aesthetic criteria might be instrumental when evaluating computer algorithms and the quality of their map products. First, all destination nodes should be grouped into tree-resembling hierarchy. The hierarchy should be able to reflect the spatial pattern of the nodes. For example, nodes that are physically close to each other should be allocated into the same sub-tree in the hierarchy. The origin is the root node, destinations are leaf nodes, and leaf nodes should be away from tree paths as far as possible for clarity. Second, the hierarchy of destination nodes must be routed towards the single origin without crossing. Edge bounding and edge rerouting are commonly necessary to achieve crossing-free tree maps. Third, the number and location of intermediate nodes, which are between the origin and destinations, should render smooth and natural transitions between adjacent tree trunks and branches. Trunks and branches are as clear and straightforward as possible. Map-makers should be able to change the number and/or the location of intermediate nodes so that they could apply their own design and aesthetic principles to the flow maps.

Algorithm

The proposed algorithm in this paper contains five key steps with the goal of automatically producing high-quality flow layout maps while offering the option of manually refining the position of intermediate nodes and the shape of flow routes. The five main steps of the algorithm are 1) add origin, destinations, and candidate intermediate points as possible nodes in the approximate Steiner tree; 2) triangulate all points, construct a network from the triangles, and conduct a series of stylized shortest path analyses from the origin to destinations in the network; 4) refine the paths using simplification, repositioning, and smoothing; and 5) render the paths to show the flow volume.

Step 1: Create Candidate Nodes for Approximate Steiner Tree

The algorithm first adds a set of points as candidate nodes for an approximate Steiner tree (CNAST). Only part of the points would be chosen, through a series of shortest-path finding procedures, as the extra vertices in the final approximate Steiner tree. Steiner tree, named after Swiss Mathematician Jacob Steiner, is a minimum-weight connected subgraph that connect all nodes in the network (Hwang and Richards 1992; Robins and Zelikovsky 2000). Steiner tree is conceptually similar to the minimum spanning tree (MST). MST connects network nodes with given network edges and has the shortest total path length. The Steiner tree also connects nodes with minimum total path length but it utilizes extra vertices that are not the original network nodes. Finding the Steiner tree from a set of points is a NP-complete problem and only has approximate solutions for most application cases (Garey, Graham, and Johnson 1976; Kou, Markowsky, and Berman 1981). Using external candidate points, this flow layout algorithm would create an approximate Steiner tree that connects the origin and destinations through extra intermediate nodes.

The candidate Steiner tree node set can be created from a few different sources. The principle here is to cover areas where the optimal Steiner tree nodes are most likely to locate while simultaneously controlling the total number of the candidates. First, the origin and destinations would be added as "real" nodes in the tree (other nodes are also called "dummy" nodes in this paper). Second, a line segment is plotted from the origin to each individual destination and points are evenly sampled on the line. These lines help capture the directions from the origin to the destinations. Third, a group of points are created around each destination. This is to create multiple possible routes connecting the tree to the destinations in the case of path conflicts. Fourth, optional trunk routes could be added to the set in order to direct the flow. This is particularly useful when there are well known migration routes, for instance. Last, a systematic or random sample points could be created within the extent of the map to fill critical gaps. These points become necessary when points from other sources cannot produce quality results. Note that this option should be used with caution because it will generate a large number of points and would reduce the efficiency of the algorithm.

When adding new nodes to the candidate Steiner tree node set, if they are too close to existing nodes, they should be ignored in order to avoid overcrowded node clusters. Furthermore, all intermediate nodes or dummy nodes should not be too close to the "real" nodes for the purpose of making rooms for visually attractive curves that end at the destinations. This step is summarized as follows (Figure 1).

- Add candidate Steiner tree nodes or "dummy" nodes
  - Add points on the line segments from origin to destinations: get the "big" direction
  - Add points around the destinations: help make destinations leaf nodes in the tree
  - Add external points (optional): explicitly highlight important routes
  - Add auxiliary random or systematic points (optional): fill in the gaps
  - Keep nodes away from each other with a minimum distance: avoid arbitrary swamps of nodes
  - Remove all "dummy" nodes that are too close to "real" nodes: make room for curves.

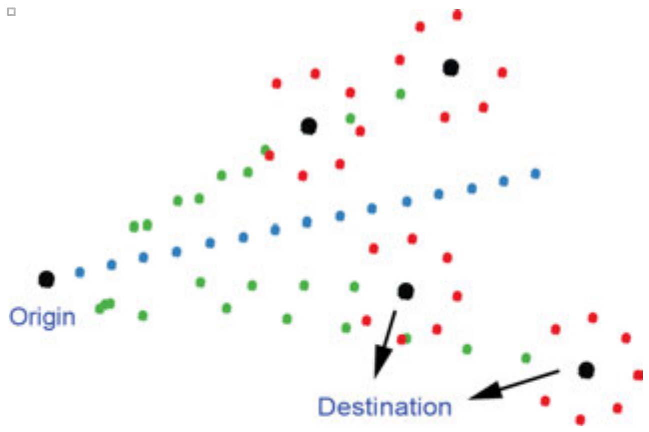

Figure 1 Candidate Nodes for Approximate Steiner Tree (red: points around destinations, green: points on the line segments from origin to destination, blue: external points)

Step 2: Triangulate points, build network, and find paths with constraints

Using all points in the candidate Steiner tree node set, a Delaunay triangulation could be conducted to find how the mapped area can be decomposed into connected triangles. The triangulation could be a regular or a constrained one. When using the constrained, the connectivity between the origin to destinations must be preserved. Then, a network would be constructed with the vertices of those triangles being the graph nodes and their edges being the graph edges (Figure 2). In the network, vertices and edges are categorized in different groups. Vertices could be "real" nodes, that is the origin and destinations, or "dummy" nodes, that is the intermediate nodes created for the approximate Steiner tree. From the two types of nodes, three types of edges are defined: edges that connect two "real" nodes, two "dummy" nodes, or one "real" and one "dummy" node.

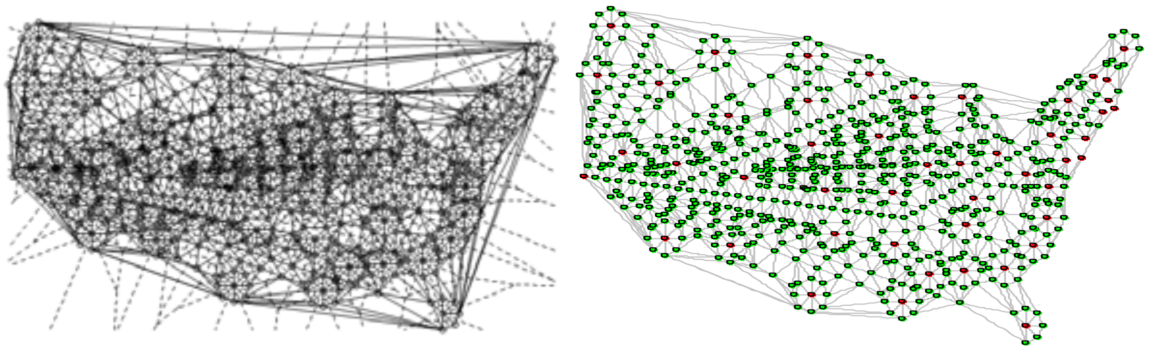

Figure 2 Triangulation and Network Construction

With the network, it is critical to find the paths that form the approximate Steiner tree and "optimally" connect the origin and destination nodes with the intermediate nodes in between. This is the core of the proposed algorithm and is based on the weighted betweenness centrality of nodes and edges. Betweenness centrality is a network statistic that measures the number of occasions in which a graph node or edge is included in the shortest path between any two nodes in the network. The edges and intermediate nodes that are never on the shortest paths from origin to any destination would be removed from the network as they contribute little or nothing to the optimal routes (Figure 3). The weight of each edge left in the network is also accumulated when it is repeatedly included in different shortest paths. In the end, the edges that are more often being chosen in the shortest paths would have more weights and are more likely to be included in the shortest path to the next destination. In other words, the shortest distance path finding has positive feedback and edges are able to accumulate their advantages in the calculation. This is similar to the evolution of river channels in the natural world, where the riverbeds of main streams experience more erosion because of greater water volume, lower attitude, and would likely gather more water over time and become the main trunk.

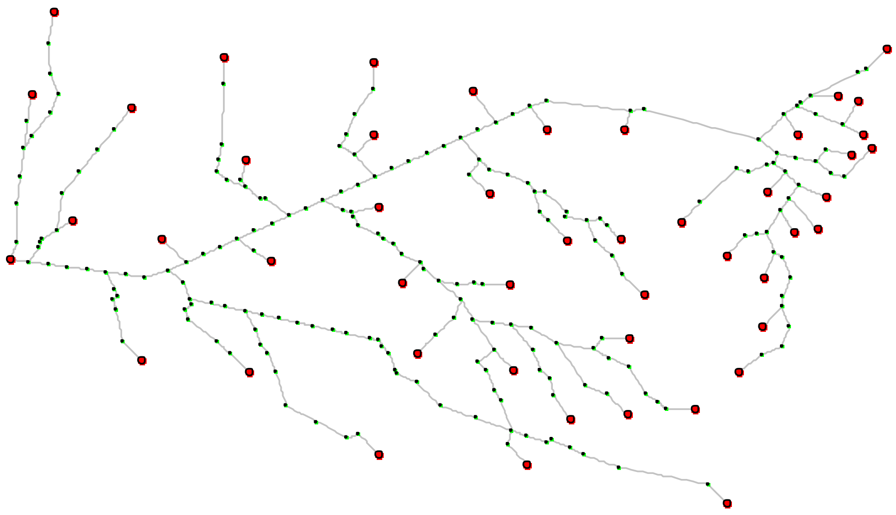
Figure 3 Path Finding in the Network for Approximate Steiner Tree

The edges that are not on any shortest paths in the Steiner tree will be removed. The intermediate nodes that are solely associated with those edges will be accordingly removed. During the shortest path finding and edge removing process, the following constraints must be considered.

- Constraints when finding shortest paths and removing edges (similar to a constrained MST)
  - Leaf node: all destination nodes must be leaf nodes and all leaf nodes must be destination nodes.
  - Root node: the origin node shall be the only root node.
  - "Real" nodes: only "dummy" nodes or edges connecting two "dummy" nodes can be removed.
  - Connectivity: all destination nodes must be connected to the origin node.
  - Degree: all nodes must have a degree of three or smaller. It means the tree must be binary.

Step 3: Simplify and smooth the paths

After getting all the nodes and edges that define the "optimal" paths in the tree, these paths are still not smooth enough because the intermediate nodes were created from many different sources and do not align with each other. In order to derive smooth paths, some unimportant intermediate nodes must be removed. A process similar to the Douglas-Peucker generalization is then applied to remove those two-degree intermediate nodes that have minimal impacts on the shape of the path. Two-degree nodes only influence the shape of the paths and removing them would not change the network topology. One important rule when deleting intermediate nodes is to keep network edges from intersecting. The simple line segment intersecting routine can be used to check this constraint. Any redundant edges that link to four or more degree nodes are also removed.

Once the network paths are simplified, a smoothing procedure can be applied to create visually more appealing routes between the origin and destinations. The smoothing process goes through each individual path from the origin to the destination. Every step of the smoothing involves three nodes. The smoothing is based on the quadratic form of a Bézier curve and the detail of the method is described below (Figure 4). This method is a local operator and only influences the shape between two nodes. However, changing the position of a single intermediate node would impact two segments in the path.

Inputs: Previous controlling node (N_p), From node (N_f), and To node (N_t).
Outputs: A Bézier curve between N_f and N_t
Process: Draw an extension line L_{pf} from N_p to N_f. If $\angle N_p N_f N_t$ is not close to 180 degree, i.e., N_t is not on L_{pf} and far from it. Add a point $P_{control}$ on L_{pf} and make $dist(N_f, P_{control}) = dist(N_f, N_t)$. Use the quadratic form of Bézier curve to generate a smooth line BC_{ft} connecting N_f and N_t. If BC_{ft} intersects with another curve, move the point $P_{control}$ along the extension line L_{pf} until BC_{ft} only goes through open area or the program reaches maximum number of tries.

If N_t is on the extension line L_{pf} or very close to it, at least two controlling points $P^1_{control}$ and $P^2_{control}$ are needed to generate a smooth curve. $P^1_{control}$ should be along the extension line L_{pf}, between N_f and N_t and locates two thirds of line segment LS_{ft} from N_f. $P^2_{control}$ should be on the line going through N_t and be perpendicular to L_{pf}, with $dist(N_t, P^2_{control}) = dist(N_f, N_t)$. Similarly, move these two controlling points along these two lines to avoid collision with other curves.

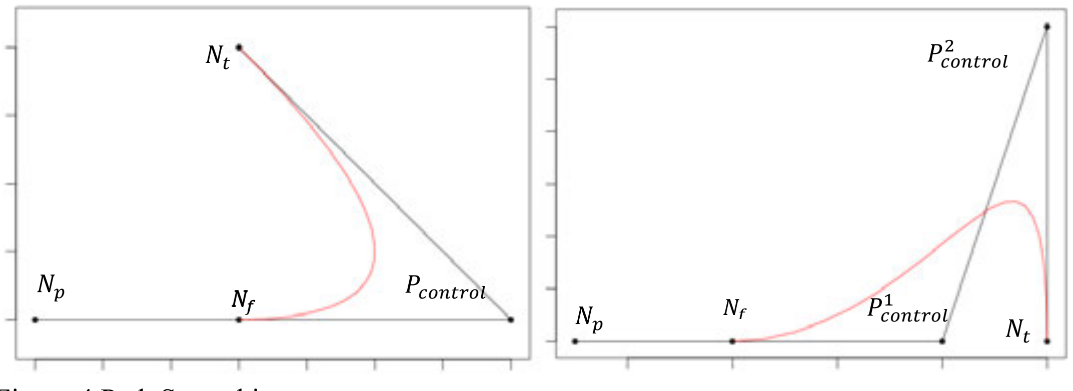

Figure 4 Path Smoothing

Step 4: Refine paths automatically or manually (optional)

This is an optional step that allows map-makers to change the position of intermediate nodes. Contrast to the simplification that removes unnecessary intermediate nodes, path refinement changes the position of the intermediate nodes instead of deleting them. Because the origin and destination nodes are “real” nodes with geographic coordinates, only the intermediate nodes can be moved during the process. The automatic method would try to move “dummy” nodes so the angles between path segments can be maximized with the constraints of not intersecting other paths. The effect is similar to line straightening with curvy, smooth transition. The algorithm offers an interactive node selection and relocation interface in the R® and RStudio® environments. It automatically redraws the smoothed paths of the flow layout and gives users instant feedback. This process would continue until users terminate it. The edited network paths can then be rendered using the actual flow volume.

Step 5: Render paths with varying color and width

The last step of the flow visualization is to render the paths using varying colors and width. The color scheme is using two divergent colors for the origin and the destinations. The color gradually changes from the color of the origin to the color of the destinations proportional to the accumulative lengths of the network paths. It is mathematically determined by interpolating the two colors using the path distances to the origin and destinations. Similarly, the width of the approximate Steiner tree can be determined by scaling the edge width using the total flow volume that goes through that particular edge. Depending on the nature of the flow data, the width can be transformed with logarithm or negative exponential functions in order to achieve the best contrast.

Results

Two examples are provided to illustrate the algorithm. The first example is the migration flow to California from all other states in the U.S. in the 2000 census (Figure 5). It adopts the optional external migration paths to generate candidate Steiner tree nodes. No auxiliary random or systematic points are used. This is a many-to-one flow map, with the centroids of states as origins and the centroid of California as the only destination. The color and width of the migration paths are scaled according to the population that migrated to California in the census data. The transition between nodes and edges is smooth and the map also shows clear clustering patterns. For example, the cluster of Northeast, Southeast, and Midwest are visually identifiable in the map. At the same time, the flow map effectively arranges the flow routes to reach overcrowded destination nodes like those in the Northeast.

The second example is the soybean export from Brazil (Figure 6). The origin is the centroid of the Brazil and destinations are major sea ports in major importing countries. This example used the systematic auxiliary points, which are similar to the regular grid covered on the Earth. Taking advantage of the shortest path algorithm embedded in the method, different travel costs are assigned to network edges according to their locations. Edges on the land area have higher cost that those in the ocean. And edges that fall in the polar area have extreme high cost, so the paths would not go through the Arctic ocean. As a result, the flow routes look similar to the real-world shipping routes in many locations. As the algorithm makes no line crossing a priority, routes in some areas are not close to real navigation routes. In this flow map, the sizes of origin and destinations are also scaled according to the volume of soybean trade.

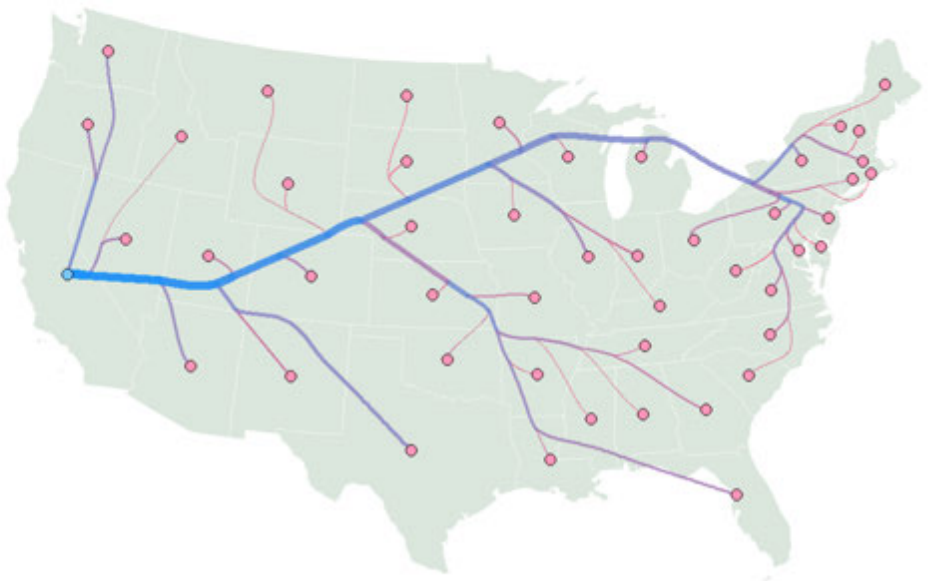

Figure 5 Migration to California in 2000 Census

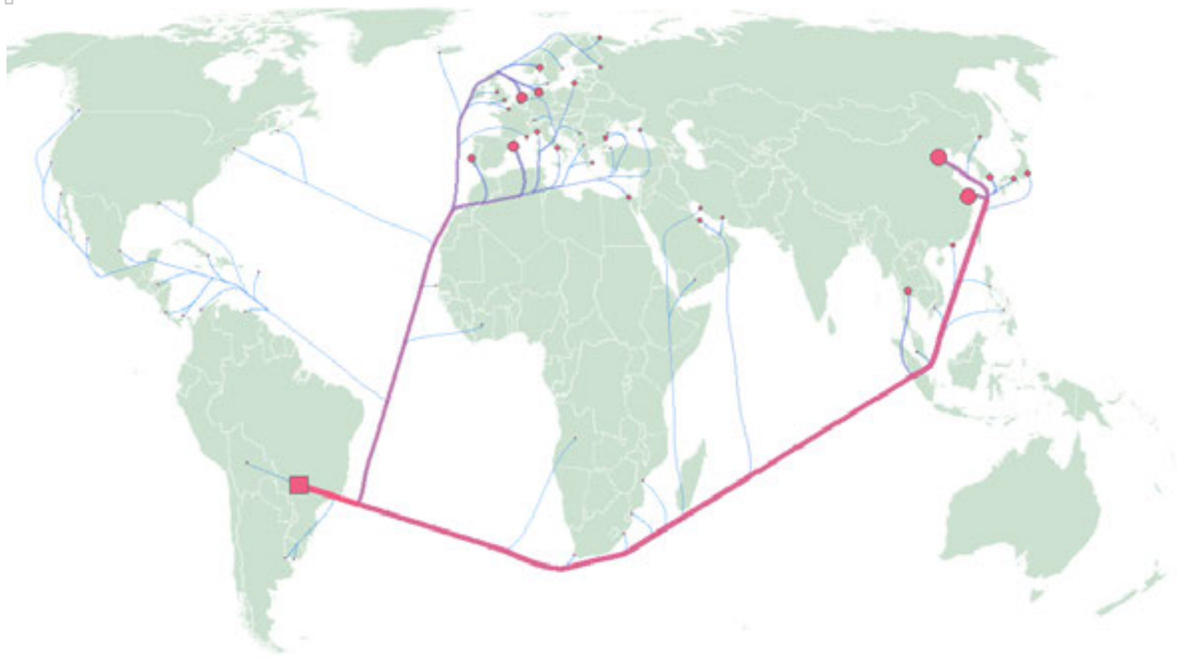

Figure 6 Brazil Soybean Exports

In both examples, no manual intervention was involved. Results were generated completely automatically by the algorithm implemented in R using packages *sp*, *spatial*, *maptools*, *deldir*, and *igraph*. All nodes are smoothly connected and transition from edge to node is natural with the Bézier curves. The background map and the path routes are rendered along with the origin and destination nodes. The colors of edges change according to the volume on each specific segment of the edges.

Discussion and Conclusion

This paper presents a new automatic flow map layout generation method. It simulates the natural flow formation in the physical world. The algorithm constructs a network using a set of arbitrary points with Delaunay triangulation. A series of styled shortest paths finding procedures are then applied to build an approximate Steiner tree that connects the origin to multiple destinations through intermediate nodes. The approximate Steiner tree can largely capture the spatial distribution of destination nodes. Then the approximate Steiner tree is simplified, smoothed, and rendered to product the final flow layout map. Importantly, the algorithm is designed with cartographic principles and has an optional node-editing tool that allows map-makers to customize flow maps while strictly following predefined topologic and aesthetic rules. By using specific auxiliary points and configuring the cost or weighted distances of edges in the network, this algorithm has the capability of guiding the flows in certain areas or directions. The two examples provided in the paper illustrate that the algorithm can effectively produce one-to-many flow layout maps with certain aesthetic standards.

While the current algorithm can automatically create quality flow maps, improvements are still possible and would be pursuit in the following directions. Further optimization could explicitly utilize the spatial clustering patterns of destination nodes (Zhu and Guo 2014; Tao and Thill 2016). The flow path could connect to a cluster first and then connect to nodes in the cluster. This is particularly useful when there are a large number of destinations and/or destinations are clustered in small areas. Second, alternative smoothing methods that can simultaneously consider multiple nodes and edges could be designed to improve the smoothness in the path transition. Third, method of automatically moving dummy nodes or optimizing dummy nodes position is needed as the initial possible points are created with little consideration of the network constructed later. A raster mask layer could be used to label the space as open or occupied. Dummy nodes can freely move in the open space without crossing the occupied space. A network node location optimization procedure could be constructed to create better locations for those three-degree intermediate nodes. Last, techniques in general graph visualization like force-directed algorithm and alpha-based edge bundling should be explored, which might help develop a solution for the ultimate many-to-many flow layout maps (Holten and Van Wijk 2009; Zhou et al. 2013; Debiasi et al. 2014).

References

Andrienko, G., N. Andrienko, J. Dykes, S. I. Fabrikant, and M. Wachowicz. 2008. Geovisualization of Dynamics, Movement and Change: Key Issues and Developing Approaches in Visualization Research. *Information Visualization* 7 (3-4):173.

Andrienko, N., and G. Andrienko. 2013. Visual Analytics of Movement: An Overview of Methods, Tools and Procedures. *Information Visualization* 12 (1):3-24.

Buchin, K., B. Speckmann, and K. Verbeek. 2011a. Angle-Restricted Steiner Arborescences for Flow Map Layout. In *Algorithms and Computation*: Springer.

———. 2011b. Flow Map Layout Via Spiral Trees. *Visualization and Computer Graphics, IEEE Transactions on* 17 (12):2536-2544.

Debiasi, A., F. Graphitech, B. Simões, and R. De Amicis. 2014. Supervised Force Directed Algorithm for the Generation of Flow Maps. Paper read at WSCG 2014 - 22nd International Conference on Computer Graphics, Visualization and Computer Vision at Plzen, Czech Republic.

Garey, M. R., R. L. Graham, and D. S. Johnson. 1976. Some Np-Complete Geometric Problems. Paper read at Proceedings of the eighth annual ACM symposium on Theory of computing.

Guo, D., and X. Zhu. 2014. Origin-Destination Flow Data Smoothing and Mapping. *IEEE Transactions on Visualization and Computer Graphics* 20 (12):2043-2052.

Holten, D., and J. J. Van Wijk. 2009. Force‐Directed Edge Bundling for Graph Visualization. Paper read at Computer graphics forum.

Hwang, F., and D. S. Richards. 1992. Steiner Tree Problems. *Networks* 22 (1):55-89.

Kent, A. J. 2013. Aesthetics: A Lost Cause in Cartographic Theory? *The Cartographic Journal* 42 (2):182-188.

Kou, L., G. Markowsky, and L. Berman. 1981. A Fast Algorithm for Steiner Trees. *Acta informatica* 15 (2):141-145.

Landesberger, T. v., F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, and A. Kerren. 2016. Mobilitygraphs: Visual Analysis of Mass Mobility Dynamics Via Spatio-Temporal Graphs and Clustering. *IEEE Transactions on Visualization and Computer Graphics* 22 (1):11-20.

Long, J. A., and T. A. Nelson. 2013. A Review of Quantitative Methods for Movement Data. *International Journal of Geographical Information Science* 27 (2):292-318.

MacDonald, G. K., K. A. Brauman, S. Sun, K. M. Carlson, E. S. Cassidy, J. S. Gerber, and P. C. West. 2015. Rethinking Agricultural Trade Relationships in an Era of Globalization. *BioScience* 65 (3):275-289.

Moenius, J. 2012. A Lay Mapmaker's Perspective on the Dilemma of Cartographic Design. *Cartographic Perspectives* (73):23-30.

Phan, D., L. Xiao, R. Yeh, and P. Hanrahan. 2005. Flow Map Layout. Paper read at IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.

Robins, G., and A. Zelikovsky. 2000. Improved Steiner Tree Approximation in Graphs. Paper read at Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms.

Sun, S., and S. Manson. 2012. Intraurban Migration, Neighborhoods, and City Structure. *Urban Geography* 33 (7):1008-1029.

Tao, R., and J. C. Thill. 2016. Spatial Cluster Detection in Spatial Flow Data. *Geographical Analysis* (online first).

Tobler, W. R. 1987. Experiments in Migration Mapping by Computer. *The American Cartographer* 14 (2):155-163.

Xiao, N., and M. P. Armstrong. 2012. Towards a Multiobjective View of Cartographic Design. *Cartography and Geographic Information Science* 39 (2):76-87.

Zhou, H., P. Xu, X. Yuan, and H. Qu. 2013. Edge Bundling in Information Visualization. *Tsinghua Science and Technology* 18 (2):145-156.

Zhu, X., and D. Guo. 2014. Mapping Large Spatial Flow Data with Hierarchical Clustering. *Transactions in GIS* 18 (3):421-35.

Shipeng Sun, Department of Environmental Studies, University of Illinois Springfield, Springfield, IL 62703