Flow Map Layout

Doantam Phan[1], Ling Xiao[1], Ron Yeh[1], Pat Hanrahan[2], and Terry Winograd[2]

Stanford University

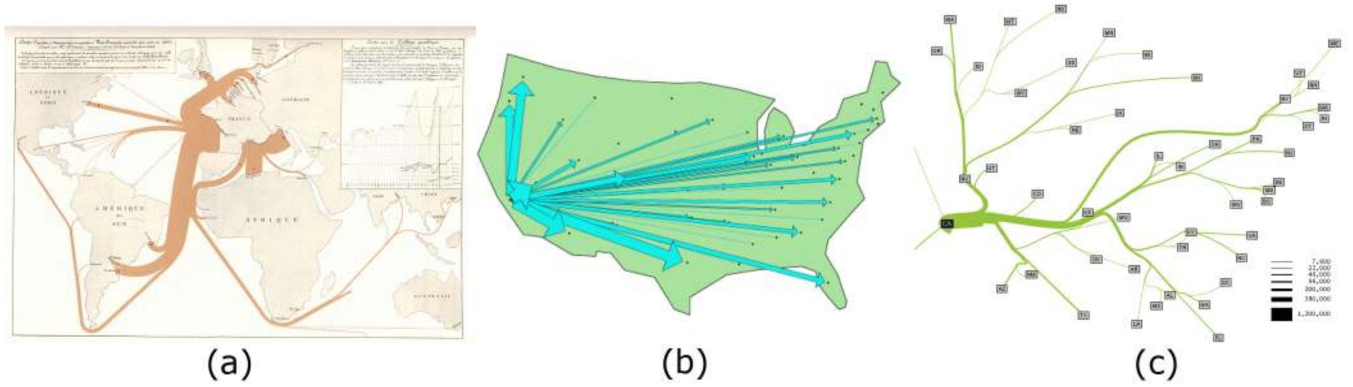

Figure 1. Flow Maps. (a) Minard's 1864 flow map of wine exports from France [20] (b) Tobler's computer generated flow map of migration from California from 1995 - 2000. [18; 19] (c) A flow map produced by our system that shows the same migration data.

Abstract

Cartographers have long used flow maps to show the movement of objects from one location to another, such as the number of people in a migration, the amount of goods being traded, or the number of packets in a network. The advantage of flow maps is that they reduce visual clutter by merging edges. Most flow maps are drawn by hand and there are few computer algorithms available. We present a method for generating flow maps using hierarchical clustering given a set of nodes, positions, and flow data between the nodes. Our techniques are inspired by graph layout algorithms that minimize edge crossings and distort node positions while maintaining their relative position to one another. We demonstrate our technique by producing flow maps for network traffic, census data, and trade data.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Viewing Algorithms
Additional Keywords: flow maps, GIS, hierarchical clustering

1 Introduction

Visualizing network flow and topology is challenging because displaying a large number of connections with lines results in visual clutter. SeeNet [3] let users adjust the visualization parameters of the map to manually reduce clutter and provided alternative designs to link maps. Node maps eliminated links and displayed connection data as node size. Matrix displays removed the geographic layout and encoded data by color. Other work tried to minimize clutter by using an extra dimension. SeeNet 3D [4;5] and work by Munzner [14] on Mbone visualization, drew links as variable-height arcs over a 2D map, where the height was encoded as the traffic volume. Unfortunately, these techniques still produce cluttered maps when encoding geographic detail, connectivity, and traffic volume on one map.

Cartographers have solved this problem with flow maps, which illustrate the movement of objects among locations. A flow map shows the spatial distribution of univariate geographic phenomena [17]. Lines of varying width which represent the number of objects being transferred are overlaid on the map. Visual clutter is reduced by merging edges that share destinations. The first flow maps illustrated rail ridership in Ireland and since then, cartographers have used flow maps to depict migrations, trade, and any data set with a from-to relationship [6].

Our goal is to produce flow maps to visualize networks and other kinds of flow data. A well-drawn flow map allows a user to see the differences in magnitude among the flows with a minimum of clutter. Figure 1a is a hand drawn map by Minard illustrating the export of wine from France. Figure 1b shows a computer generated flow map by Tobler [18; 19], which is still cluttered because it does not take advantage of the techniques of hand-drawn maps.

Our contribution is a technique that automatically generates flow maps. Figure 1c illustrates a flow map generated automatically by our system. Our approach uses hierarchical clustering to create a flow tree that connects a source (the root) to a set of destinations (the leaves). Our algorithm attempts to minimize edge crossings and supports the layering of single-source flow maps to create multiple-source flow maps. We do this by preserving branching substructure across flow maps with different roots that share a common set of nodes. We have produced flow maps for trade data, network traffic, and migration data.

2 System Design

Our analysis of good, hand-drawn flow maps reveals three common characteristics: intelligent distortion of positions, merging of edges that share destinations, and intelligent edge routing. Our technique attempts to produce flow maps with the same characteristics as these hand-drawn exemplars. Minard's map of wine exports (Figure 1a) illustrates these characteristics. The Strait of Gibraltar has been widened and the UK has been

[1] dphan, lingxiao, ronyeh @ graphics.stanford.edu
[2] hanrahan, winograd @ cs.stanford.edu

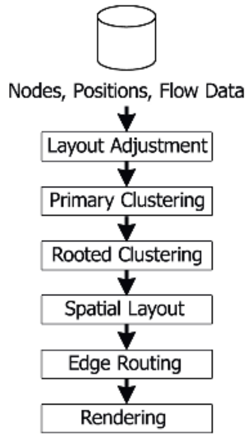

Figure 2. System Diagram

shifted away from France to make room for the flow lines. However, these distortions preserve the relative positions of countries with respect to one another. Edges going to different regions of the world are merged and in this map, edge crossings are minimized.

Other characteristics of flow maps described by cartographers concern their visual appearance [6]. Flow lines should be the dominant visual element and should be easily distinguishable from other map symbols. A linear mapping should be used to transform data values to line thickness. The map legend must be clear and provide key values for line widths. If edges cross, smaller lines should rest on top of larger lines. To our knowledge, there are no published guidelines describing how to layout a hand-drawn flow map, which is the focus of this paper.

We achieve intelligent distortion of positions by using a layout adjustment algorithm that ensures that the separation distance between nodes is greater than the maximum thickness of the flow lines. This algorithm guarantees that nodes maintain their left-right and up-down positions relative to one another [12].

Merging edges that share destinations is done by using hierarchical clustering to find nodes that are close together. The tree formed by the hierarchical clustering is the basis for the branching structure of the flow map. The clustering is generated such that the root of the flow map is the root of the tree. We use a binary hierarchical clustering because it allows us to formulate the layout in a simple and recursive manner.

Our edge drawing and routing also uses hierarchical clustering. The recursive procedure to layout node positions uses the fact that all branches are binary. Clusters are modeled as rectangular bounding boxes enclosing their nodes. The edges check for intersections with the bounding boxes of their siblings. If an intersection exists, we place the node such that edges connected to that node are routed around the bounding box.

To visualize more general networks using flow maps, we want to support maps with more than one root. We can create multiple-source maps by layering single-source maps that have been generated by our algorithm. To make layering more effective, we preserve node position and branching structure across flow maps, which makes it easier to compare and contrast flow patterns across different layers. This is done by generating a primary hierarchical clustering that relies on the positions of the input set of nodes. The rooted clustering tree used to lay out each flow map is a modification of the primary hierarchical clustering tree.

Figure 2 presents an overview of our system. The input to our system consists of a set of nodes, positions, and flow data among the nodes. There are layout phases and a rendering phase.

Layout phases ensure nodes are adequately spaced, generate a topological layout tree, choose the positions of the branching points, and route edges around obstacles. The rendering phase converts the given spatial layout tree into a set of thick splines for each edge and generates a legend for the flow map.

3 Layout

Sections 3.1 to 3.5 describe the steps of the algorithm for generating a single-source flow map. In 3.6 we describe the additional considerations that arise from having multiple layers in the system.

3.1 Layout Adjustment

The layout adjustment step enforces a minimum separation distance among the nodes in the horizontal and vertical directions and preserves their relative positions to one another. We specify the minimum separation distance to be the maximum width of a flow line in pixels. We use Misue et al.'s force scan algorithm [12] which runs in $O(n^2)$. We have observed that good flow maps contain a moderate number of nodes (less than 100), so the efficiency of the algorithm is not an issue. Nodes are sorted into two lists, one ordered by x-coordinate and the other ordered by y-coordinate. The algorithm consists of a horizontal scan followed by a vertical scan.

We also experimented with a force directed scheme from the

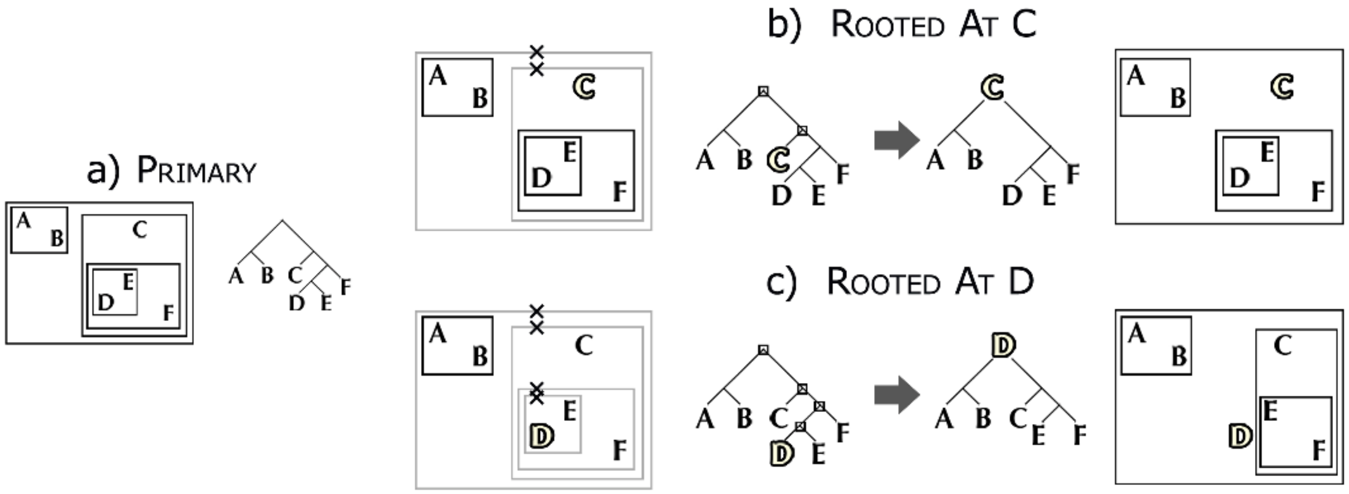

Figure 3. Hierarchical Clustering. A flow map tree is generated by clustering a set of nodes. (a) We show a spatial representation of the primary hierarchical clustering (PHC) and its equivalent tree. Rooted hierarchical clustering (RHC) modifies the PHC to produce a flow map for a particular root. The x's on the bounding boxes indicate the clusters that are not reused. (b) The RHC for a flow map from C. The (A,B) and the ((D,E),F) clusters are kept. (c) The RHC for a flow map from D. Only the (A,B) cluster is preserved.

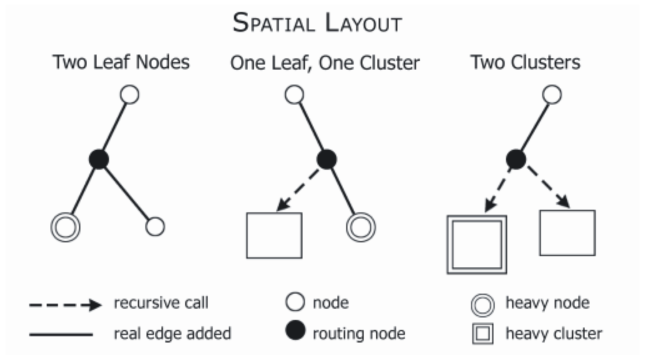

Figure 4. Spatial Layout. The binary structure of the rooted clustering allows to us generate the layout recursively. Branching points are always placed on the line between the start node and the destination that has more weight (or flow).

graph drawing literature [2]. The problem we encountered was that force directed methods often included randomness in the system to avoid local minima in layout. However, we were interested in generating flow maps with stable node positions for a given set of nodes. The force directed technique resulted in slightly different layouts each time, so we chose to use the algorithm of Misue et al. [12] which remains stable. Note that keeping node positions stable allows us to layer flow maps on top of one another. However, without this constraint, there might have been a better way to create a flow map by adjusting positions.

3.2 Primary Hierarchical Clustering

The primary hierarchical clustering captures information about the spatial distribution of the input nodes. We use an agglomerative hierarchical clustering [9] that creates a binary tree from the input set of nodes. The input nodes are at the leaves of the primary hierarchical clustering. The distance between two clusters is the Euclidean distance between the centers of their rectangular bounding boxes. Figure 3 gives an example of how a primary clustering is generated.

3.3 Rooted Hierarchical Clustering

We want to generate a flow map with a given node *r* at the root. The tree produced by the primary hierarchical clustering is not suitable because its root is a combination of clusters of input nodes. To generate a tree where node *r* is the root and connects to the other input nodes, we modify the tree produced by the primary clustering. For a given root node *r*, we accumulate nodes to be reclustered by following parent pointers from node *r* to the root of the primary clustering tree. We add clusters hanging off this path to our set to be reclustered, including the cluster with node *r*. Figure 3 demonstrates how rooted clustering modifies a primary clustering.

Our theoretical analysis indicates that following these parent pointers takes $O(\lg m)$ time, and clustering the accumulated set takes $O(\lg m)^2$ time, where *m* is the total number of nodes in the system, including nodes generated by the primary clustering as well as the input nodes. We note that this is $O(n)$ nodes, where *n* is the number of input nodes in the system. Without re-use of substructure from the primary clustering, our rooted clustering would take $O(m^2)$. Currently the system implements the simplified version of our algorithm that takes $O(m^2)$.

Rooted hierarchical clustering merges nodes differently from standard hierarchical clustering. Suppose the algorithm wants to merge clusters C1 and C2 because they are the closest in the system. Before it does so, it checks to see if C1 or C2 contains the root node *r*. Suppose C1 contains *r*. C2 is marked for attempting to merge with the root. The algorithm looks for another cluster C3 to merge with C2. C3 must be unmarked and distance(C2,C3) ≤ distance(C2, root). If we only marked the C2 cluster without looking for C3, we might miss an opportunity to merge clusters on the same side of the root. The distance check ensures that we do not merge nodes on opposite sides of the root. The algorithm terminates if it cannot find an unmarked cluster C3.

When the reclustering terminates due to the algorithm being unable to find any unmarked clusters, the remaining clusters in the system become the children of the source cluster. Note that all the nodes except for the root of a rooted clustering have two children. Since reclustering may terminate with more than two marked clusters, a rooted hierarchical clustering is not always a binary tree, in contrast to a primary clustering, which is always binary.

The last step in the rooted clustering computes the weights of the clusters. Given the flow data among the nodes, we know how much data flows from the root *r* to our destination nodes. This is recorded as the weight of a node. We compute the weight for each cluster, bottom-up, in $O(m)$ time.

3.4 Spatial Layout

The next step is to recursively lay out the tree of flows rooted at the source. Each node creates a branching structure from the parent to its two children. The challenge is to minimize edge crossings. We assume that the bounding boxes of the children do not overlap. If the layout of the subtrees corresponding to each child is contained within its bounding box, then the branch between the parent and the child will not intersect the tree corresponding to the child.

We create a branching node between the root and each of its immediate child clusters. We choose the position of that node by drawing a line *l* from the root's position *a*, to the center of the cluster *c*. The closest intersection of *l* with *c*'s bounding box is called *b*. The position of the branching node is the average position of *a* and *b*, or $(a+b)/2$. We recurse on the grand children of the root nodes.

Figure 4 illustrates the placement of a binary branching point for the three cases: a split between two leaf nodes, one leaf node

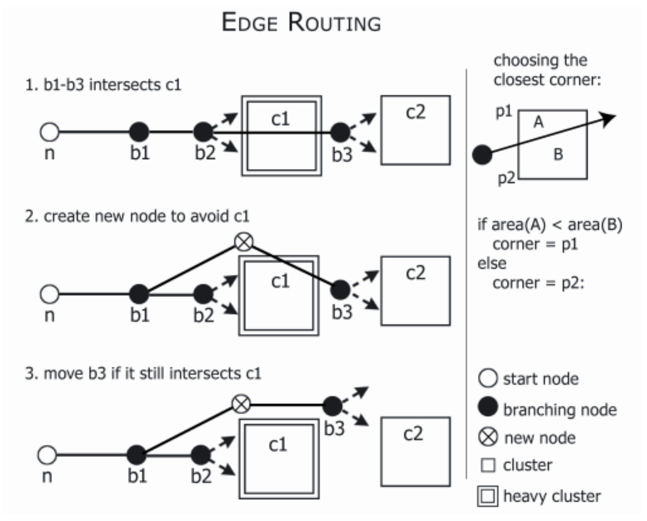

Figure 5. Edge Routing. Spatial layout may cause an intersection by placing b3 in a way that intersects c1. The algorithm finds the intersection of b1-b3 with c1, and adds a new node and adjusts the position of b3 to avoid c1 if necessary.

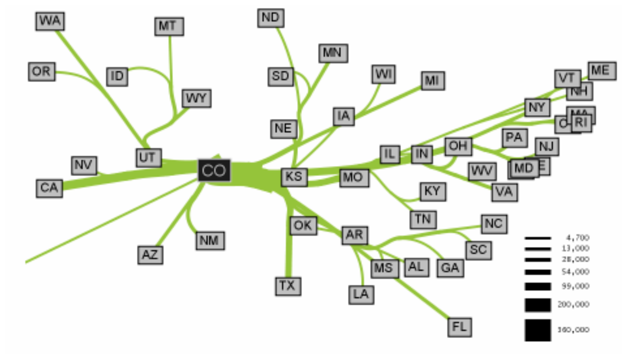

Figure 6. Outgoing migration map from Colorado from 1995-2000, generated by our algorithm without layout adjustment or edge routing. Note how the spatial structure imposed by our hierarchical clustering still merges edges in a way that produces a clean map despite the lack of edge routing.

and a cluster, or two clusters. Branching nodes *n* are placed halfway between the position of the start node, *a,* and another point *b*. For two leaf clusters, *b* is the location of the heavier node. For the other cases, let *l* be a line between *a* and the center of the heavier cluster. *b* is the closest point to *a*, on the line *l*, that intersects the bounding box of either child cluster. In both cases the position of the branching node *n* is set to be *(a+b)/2*.

We always add an edge from our starting node to the branching node *n*, and an edge from *n* to the leaf nodes. We always recurse on clusters. Note the branching point may be placed where an outgoing edge intersects the bounding box of one of its children. We describe how we avoid these intersections in the next section.

3.5 Edge Routing

We route edges around the bounding boxes within the same hierarchical cluster. A consequence of this is that the edges from the root node to its immediate children are not routed, because they are not in a cluster with one another. We will discuss this issue in more detail in Section 5. We illustrate how we route edges around sibling clusters for the case of two clusters in Figure 5. Step 1 illustrates how for each branching point *b1*, we check if the edge *b1-b2* or the edge *b1-b3* intersects its sibling's bounding box. We see that *b1-b3* intersects the bounding box of cluster *c1*, so we know that *c1* occludes *c2*. Step 2 shows how we find the closest corner on the bounding box *c1* to the line *b1-b3* and create a new node at that corner to route the edge around the bounding box. In Step 3, we check the line from the new node to *b3* and see if it intersects *c1*. If it does, we move the point *b3* to just outside the corner of the bounding box of *c1*.

Figure 5 also illustrates how we decide which corner to position nodes. If an edge that intersects a bounding box, it splits it into two parts. We compute the areas of each part of the box that was split by the edge. We choose the corner on the side of the edge with less area.

3.6 Multiple-Layer Issues

Previous sections described how the layout works for a single layer, which is created to depict data from an initial query. Here we describe the modifications to the algorithm when users generate further queries to get a combined visualization.

Each additional query might introduce new nodes that did not exist in the previous query. As a result, the layout adjustment step must be re-run for each new query, which may take *O*(m^2) where *m* is the total number of unique nodes accumulated over all the queries. We reset the positions of all the nodes to their initial locations, and then run the layout adjustment step as described in 3.1. When we get new nodes, we also run the primary hierarchical clustering step again, which also takes *O*(m^2).

The hierarchical clustering steps in our system must be modified to account for the fact that not every cluster belongs to every layer. Every cluster has a list of the layers with which it is associated. Clusters also store a mapping from layers to a list of their subclusters which appear in that layer. Our current implementation of rooted clustering runs in *O*(m^2). This is because we do not propagate the layer information up the tree when constructing the primary hierarchical clustering, so we have to inspect the subtree to find which subclusters appear in which layer. However, by storing subclusters' information at each level, this would improve the running time of rooted clustering *O(lg m)* to accumulate nodes and *O*($(\lg m)^2$) to recluster the set.

We note that the edge routing is done on a per-layer basis, which will cause many edge crossings if too many maps are layered on top of one another.

4 Rendering

To make the flow map visually appealing, each edge is rendered as a catmull-rom spline which interpolates between the nodes of the spatial layout tree that we produced. The splines are rendered such that they have width proportional to the weight of the edge. We can set the widths of the splines to lie between a minimum and a maximum display width. We map the data to this interval using a linear or a logarithmic function. We adjust the splines so that edge widths add up visually. For each binary branch, we shift the starting point of the child splines by a distance proportional to the width of the child, in a direction perpendicular to the vector that represents the edge.

We generate a legend for the map by running k-means clustering [9] on all the edge widths that appear in the map. We chose to generate 7 categories for the legend. Since we do not enforce that a bin in our k-means should not contain 0, some legends generated by our system may have less than 7 categories. The location of the legend on the map is set manually. A unique color is set for each layer so to distinguish them better.

5 Results and Discussion

We used several data sets to test our method. The US Census data for 1995-2000 [21] records all of the county to county migrations. Network traffic data was recorded from our lab's router over a period of several months. The ecological footprint data is courtesy of the Footprint Network [13]. All the node positions were specified in latitude and longitude, and converted to screen space via a Mercator projection. The maps produced by

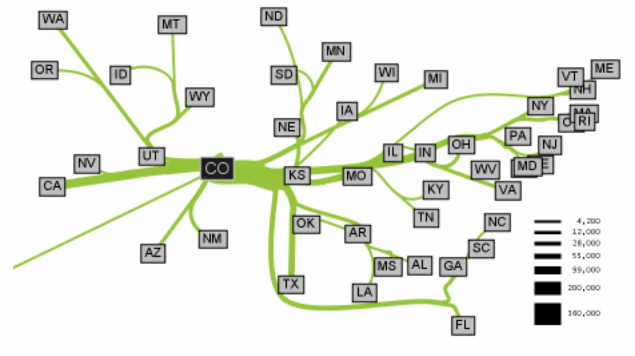

Figure 7. Outgoing migration map from Colorado for 1995-2000 generated using edge routing but no layout adjustment.

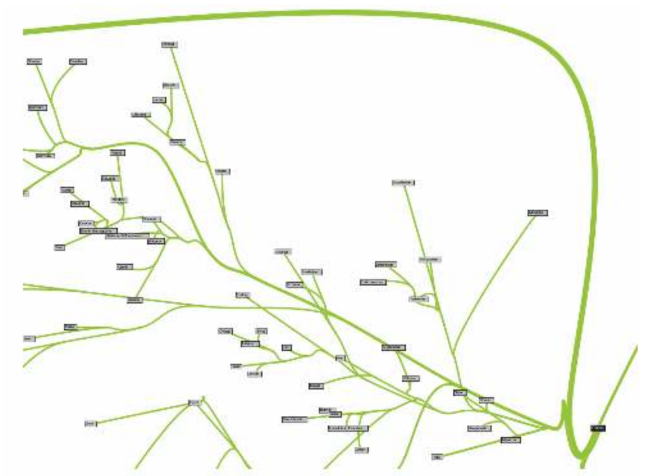

Figure 8. Imports to China. A part of a flow map showing the top 200 countries from which China (bottom right) receives imports. The thick line curving around the top are the imports from the USA (not shown), which without edge routing would have gone straight through the middle of the image. Edge crossings still occur in some parts of the map, illustrating cases where our routing does not work. For more information see the discussion in Section 5.

our system were produced in a few seconds of time on a 1.4GHz laptop. The system itself is written in Java using the prefuse visualization toolkit developed by Heer [8].

Our technique produces aesthetically pleasing flow maps for a variety of data sets. However, there are still cases where the flow maps do not look as good as expected. To understand these failure cases, we analyze our technique in terms of the three characteristics of the good flow maps.

Intelligent Distortion. The force scan algorithm [12] used for layout adjustment separates the nodes and guarantees that we are not changing the relative positions of the nodes to one another. The problem with this method can be seen by comparing Figure 1c with Figure 6. While Figure 6 is clearly recognizable as the outline of the United States, Figure 1c has been vertically distorted so that the states on the East Coast are spread apart. This stretches out the West Coast too much.

Merging of edges that share destinations. Our simple version of hierarchical clustering is able to find spatial clusters in the data sets we use. The biggest drawback to our algorithm in this regard is that all splits are binary. If there are too many destination nodes in a small area, forcing binary splits introduces too many extra routing nodes and leads to clutter. In addition, with too many nodes, continuity constraints for splines cause them to loop.

Intelligent edge routing. Our heuristic works well for simpler cases (Figure 7) and sometimes does not work well for larger cases (Figure 8). One problem is that we do not perform any routing on the edges leaving the root. Branches in these separate binary trees don't know about each other and may overlap. In Figure 8, the reason why there are so many edge crossings is that our method assumes bounding boxes created by our spatial clustering are disjoint. However, our clustering does not always produce disjoint boxes, so we cannot always apply our heuristic.

Layering and branching structure. We have found that the method for sharing substructure across flow maps that share a common set of nodes works quite well. Figure 9 has an example where two flow maps layered together share branching structure to several nodes.

Linear or logarithmic display widths. In our initial maps we used logarithms to map our data to a display width, because we were working with network data where there were many orders of magnitude in the data that we were plotting. However, for data sets not dominated by a few high magnitude values, we find that linear mappings draw the eye more. All the maps we demonstrated in this paper use linear mappings.

6 Related Work

A survey of American flow mapping history and techniques is described in [15]. Flow map design is discussed in various geography textbooks [6; 17]. Guidelines are given on legend design and the use of a visual hierarchy to emphasize flow lines over region boundaries. Ruggles and Armstrong [16] discuss a framework for the cartographic visualization of networks. More generally, map design is discussed in [11].

An alternative method of layout adjustment is described by Lyons [10], who improves the distribution of nodes in a graph and uses measures for graph similarity and distribution to decide if too much distortion has occurred. Since this method tends to more evenly distribute the nodes, it tends to blur out geographic features, which is not as useful for flow maps.

Our edge routing technique is related to the ones found in the graph drawing literature. Dobkin et al. [7] describes how to route individual edges through a set of obstacles represented by polygons. Unfortunately, they leave the question how to do this for multiple edges open.

Our work is related to Agrawala and Stolte [1], who studied hand-drawn route maps to understand the principles that made them effective. They made use of intelligent distortion to produce computer generated route maps.

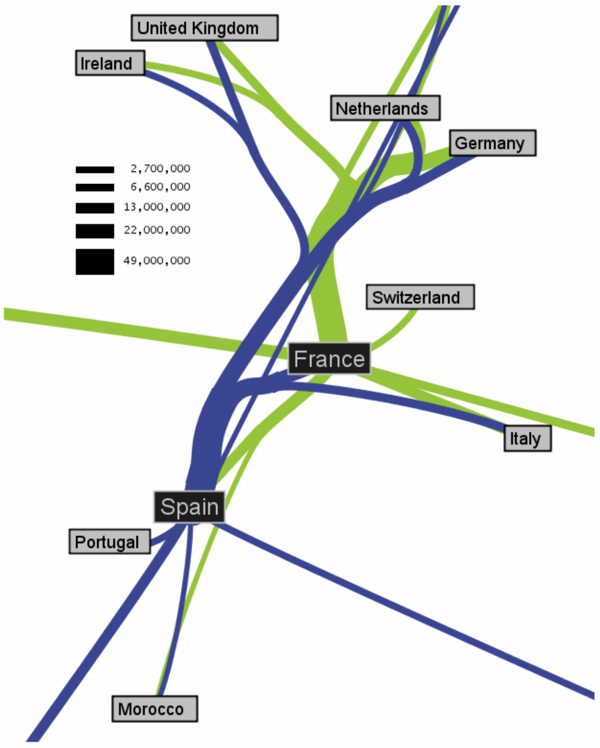

Figure 9. Branching Structure. A close-up of top 15 imports to Spain and France. Notice the branching structure is shared across different nodes, for example Spain, and France branch to the Netherlands, Germany and the UK in the same way.

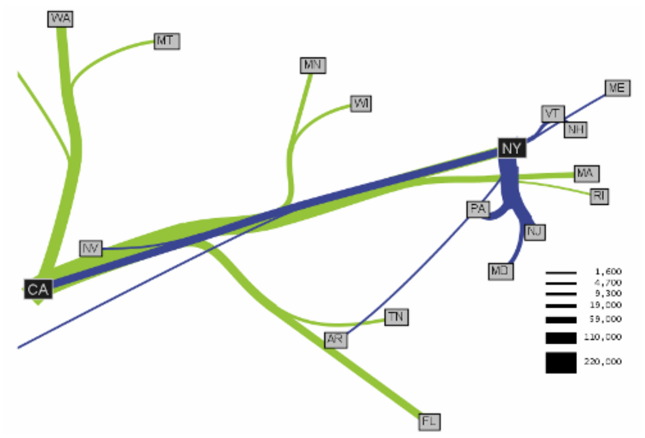

Figure 10. California and New York migration. Another example of how layering can be used in our system. The map shows the top 10 states that migrate to California and New York. Flow maps make it easy to spot an interesting spatial pattern, namely that New York tends to attract people from the East Coast, while California residents come from more geographic regions in the United States.

7 Conclusion and Future Work

There are many areas of future work suggested by this project. Global enforcement of horizontal and vertical ordering seems too strict for flow maps. A better layout adjustment algorithm might only enforce horizontal and vertical ordering within some 2D local window of each node, so that things may be globally distorted but locally consistent.

Although using binary clustering makes our spatial layout step simpler, having non-binary splits could eliminate some clutter and some loopy edges. Also, instead of our heuristic-based method for edge routing, it may be possible to use simulated annealing to adjust the edge positions.

We are also interested in the layout and display of data that does not explicitly have a location. Although all of our work has been on data sets with geographic locations, it may be interesting to generate flow maps for more abstract sets of objects that have no predefined positions but that can be grouped in other ways.

Now that we have an algorithm to generate flow maps programmatically, we are interested in developing techniques to interact with the data. We are interested in working with people who analyze flow data to see how they use an interactive tool.

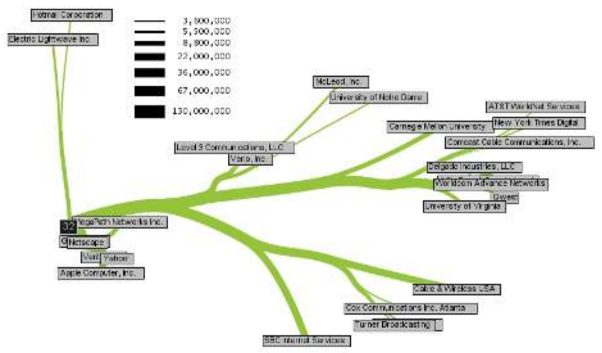

Figure 11. An example of the top ASN (Autonomous Systems) that a computer from our lab communicated with in one day. The latitudes and longitudes for the ASNs were obtained by manually looking for the city or state in which the ASN was registered.

In this paper, we have presented a method to automatically generate flow maps. We use distortion to ensure that our nodes are well spaced but still preserve their relative positions to one another. We merge edges based on their destinations using hierarchical clustering. This allows flow maps with the same input nodes to share branching structures. Finally we use the spatial information given to us by the hierarchical clustering to do edge routing to avoid edge crossings.

Acknowledgements

This work was supported by the Stanford Regional Visual Analytics Center and the Department of Energy.

References

[1] Maneesh Agrawala and Chris Stolte. "Rendering Effective Route Maps: Improving Usability Through Generalization". SIGGRAPH 2001. p. 241-250. 2001.

[2] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tolis. Graph Drawing. Prentice Hall, New Jersey, 1999.

[3] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. Visualizing Network Data. IEEE Transactions on Visualization and Computer Graphics, 1995.

[4] Kenneth C. Cox and Stephen G. Eick. Case Study: 3D Displays of Internet Traffic. Proceedings of Infovis 1995.

[5] Kenneth C. Cox, Stephen G. Eick, and Taosong He. 3D geographic network displays. SIGMOD Rec., ACM Press, 25:50-54, 1996.

[6] Borden D. Dent. Cartography : Thematic map design. McGraw-Hill. New York. 1999.

[7] David P. Dobkin, Emden R. Gansner, and Eleftherios Koutsofios, and Stephen C North,. Implementing a General-Purpose Edge Router Springer-Verlag, 262-271. 1997.

[8] Jeffrey Heer, Stuart K. Card, and James A. Landay, prefuse: a toolkit for interactive information visualization. CHI 2005, 421-430. 2005.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn, Data clustering: a review ACM Comput. Surv., ACM Press, 31:264-323 , 1999.

[10] Kelly A. Lyons, Cluster busting in anchored graph drawing IBM Press, 7-17. 1992.

[11] Alan MacEachren. How maps work: Representation, Visualization, and Design. Guilford Press. New York. 1995.

[12] Kazuo Misue, Peter Eades, Wei Lai, Kozo Sugiyama. Layout adjustment and the Mental Map. Journal of Visual Languages and Computing. 1995.

[13] D. Moran, M. Wackernagel, S. Goldfinger, and M. Murray. *International Ecological Trade Flows*. Global Footprint Network. www.footprintnetwork.org. 2005.

[14] T. Munzner and E. Hoffman and K. Claffy and B. Fenner. Visualizing the global topology of the MBone. InfoVis 1996.

[15] M. Jody Parks. American Flow Mapping. Unpublished master's thesis. Atlanta: Georgia State University, Department of Geography.

[16] Amy J. Ruggles and Marc P. Armstrong, Toward a conceptual framework for the cartographic visualization of network information. *Cartographica*, 34:1-15. 1997.

[17] Terry A. Slocum. Thematic cartography and visualization. Prentice Hall. New Jersey. 1999.

[18] Waldo Tobler. Experiments in Migration Mapping by Computer. American Cartographer, 1987.

[19] Waldo Tobler. Movement Mapping. http://csiss.ncgia.ucsb.edu/clearinghouse/FlowMapper. 2004.

[20] Edward R. Tufte. The Visual Display of Quantitative Information. Graphics Press. Chesire, Conneticut. 2001.

[21] U.S. Census Bureau. 2003. County-to-County Migration Flow Files. http://www.census.gov/population/www/cen2000/ctytoctyflow.html