GMap: Visualizing Graphs and Clusters as Maps

Emden Gansner*
AT&T Labs - Research, 180 Park Ave, Florham Park, NJ 07932

Yifan Hu†
AT&T Labs - Research, 180 Park Ave, Florham Park, NJ 07932.

Stephen Kobourov‡
University of Arizona, 1040 E 4th Street, Tucson, AZ 85721.

ABSTRACT

Information visualization is essential in making sense out of large data sets. Often, high-dimensional data are visualized as a collection of points in 2-dimensional space through dimensionality reduction techniques. However, these traditional methods often do not capture well the underlying structural information, clustering, and neighborhoods. In this paper, we describe GMap, a practical tool for visualizing relational data with geographic-like maps. We illustrate the effectiveness of this approach with examples from several domains. All the maps referenced in this paper can be found in `www.research.att.com/~yifanhu/GMap`.

Index Terms: G.2 [Discrete Mathematic]: Graph TheoryXX—[H.3]: Information Storage and Retrieval—Clustering

1 INTRODUCTION

Graphs capture relationships between objects and graph drawing allows us to visualize such relationships. Typically vertices are represented by points in two or three dimensional space, and edges are represented by lines between the corresponding vertices. The layout optimizes some aesthetic criteria, such as, showing underlying symmetries, or minimizing the number of edge crossings. While such point-and-line representation are most commonly used, other representations have also been considered. For example, treemaps [30] use a recursive space filling approach to represent trees. There is also a large body of work on representing planar graphs as contact graphs [8, 14, 22], where vertices are embodied by geometrical objects and edges are shown by two objects touching in some specified fashion. Koebe's theorem [18] shows that all planar graphs can be represented by touching disks. A similar representation is possible with triangles, where two adjacent vertices correspond to a vertex-to-side touching pair of triangles, as shown by de Fraysseix *et al.* [8]. If vertices are represented by rectilinear regions and edges correspond to side-to-side contact between paired regions, He [14] has shown that all planar graphs have such drawings. Graph representations of side-to-side touching regions tend to be visually appealing and have the added advantage that they suggest the familiar metaphor of a geographical map.

In this paper we describe GMap, an algorithm to represent general graphs as maps. Clearly, there are theoretical limitations to what graphs can be represented exactly by touching polygons, namely, subclasses of planar graphs. However, our aim here is practical rather than graph theoretical. We do not insist that the created map be an exact representation of the graph but that it captures the underlying relationships well. With this in mind, we do not insist that all vertices are represented by individual polygons either. In fact, we group closely connected vertices into regions. If we would like to show all of the relationships, we can superimpose a graph drawing on top of the map. Our overall goal is to create a representation which makes the underlying data understandable and visually appealing. Our map representation is especially effective when the underlying graph contains structural information such as clusters and/or hierarchy.

Given an input a graph (e.g., similarity between books as determined by purchase behavior at Amazon.com) we produce a map with a "natural" map-like look, outer boundaries that follow the outline of the vertex set, and inner boundaries having the twists and turns found in real maps; see Fig. 1. Our maps also can have lakes, islands, and peninsulas, similar to those found in real geographic maps.[1]

2 RELATED WORK

There is a large body of work on contact polygon representation of planar graphs, and in floor planning, as discussed in Section 1. While some theoretical problems from this area are related to our work, the emphasis in this paper is not on the strict preservation of adjacency information, but on a practical approach to visualizing general non-planar graphs.

There is little prior work on representing general (non-planar) graphs as geographic maps. In geography there are many papers about accurately and appealingly representing a given geographic region, or on re-drawing an existing map subject to additional constraints. Examples of the first kind of problem are found in traditional cartography, e.g., the 1569 Mercator projection of the sphere onto 2D Euclidean space. Examples of the second kind of problem are found in cartograms, where the goal is to redraw a map so that the country areas are proportional to some metric, an idea which dates back to 1934 [28] and is still popular today (e.g., the New York Times red-blue maps of the US, showing the presidential election results in 2000 and 2004 with states drawn proportional to population).

The map of science [5] uses vertex coloring in a graph drawing to provide an overview of the scientific landscape, based on citations of journal articles. Treemaps [30], squarified treemaps [7] and the more recent newsmaps [33] represent hierarchical information by means of space-filling tilings, allocating area proportional to some metric.

Representing imagined places on a map as if they were real countries also has a long history, e.g., the 1930's Map of Middle Earth by Tolkien [32] and the Bücherlandes map by Woelfle from the same period [1]. More recent popular maps include xkcd's Map of Online Communities [4]. While most such maps are generated in an *ad hoc* manner and are not strictly based on underlying data, they are often visually appealing.

In self-organizing maps (SOM) [19] an unsupervised learning algorithm places objects on a two-dimensional grid such that similar objects are close to each other. Similar to our GMap algorithm, SOM also uses a map metaphor to visualize the resulting embedding. Unlike GMap, in SOM the "map" is created by coloring cells

*e-mail: erg@research.att.com

†e-mail:yifanhu@research.att.com

‡Work done while this author was at AT&T Labs. e-mail:kobourov@cs.arizona.edu

[1]This paper contains zoomable high resolution images; all the images are also available at `www.research.att.com/~yifanhu/MAPS`.

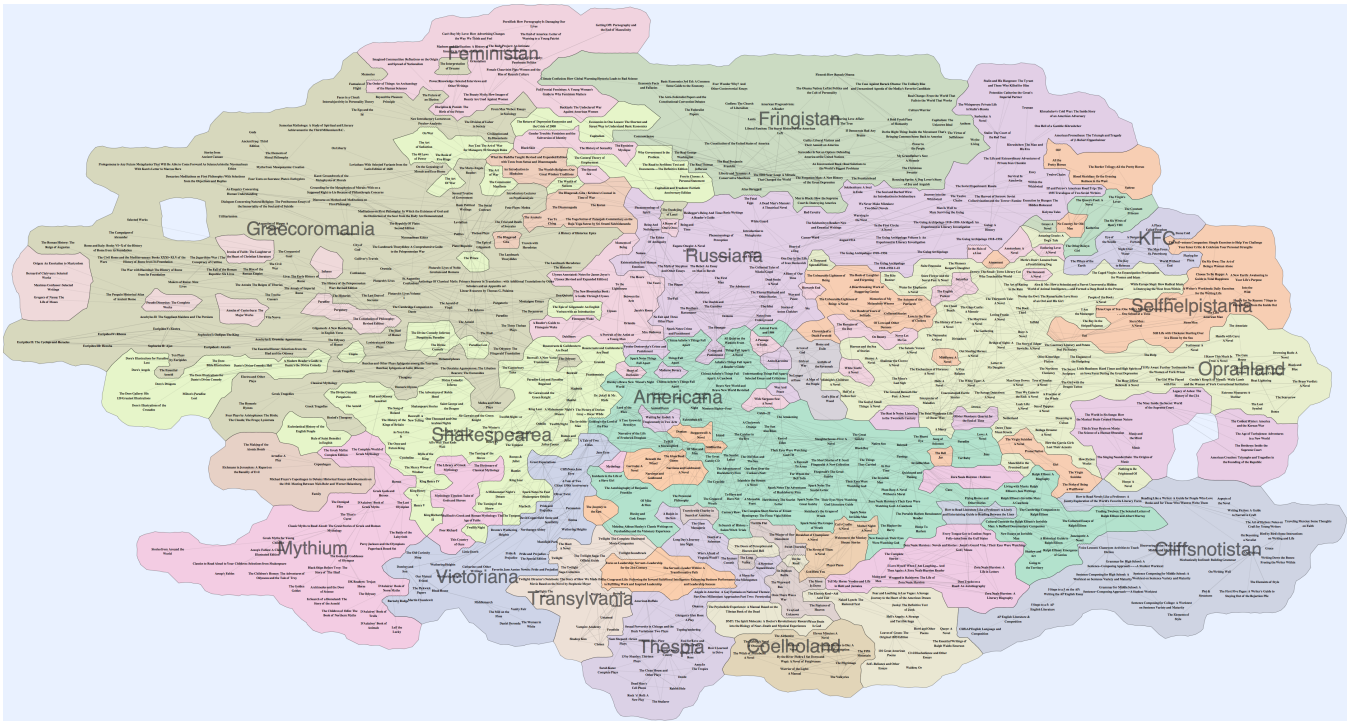

Figure 1: A map of books related to “1984” from Amazon.com

of the grid based on a feature value, therefore operating on a discrete grid space, without a clear inner boundary between “countries”. Furthermore, the grid tends to fit a rectangular box, therefore overall outline of the point set often follows that shape.

Generating synthetic geography has a large literature, connected to its use in computer games and movies. Most of the work (e.g., [23, 25]) relies on variations of a fractal model. These techniques could provide additional photorealism, and may be used in future extention of our work.

3 THE GMAP ALGORITHM

The input to our algorithm is a relational data set from which we extract a graph $G = (V,E)$. The set of vertices V corresponds to the objects in the data, e.g., authors in the graph drawing community, and the set of edges E corresponds to the relationship between pairs of objects, e.g., co-authoring a paper. In its full generality, the graph is vertex-weighted and edge-weighted, with vertex weights corresponding to some notion of the importance of a vertex and edge weights corresponding to some notion of the distance between a pair of vertices.

The first step in our GMap algorithm is to embed the graph in the plane. Possible embedding algorithms include principal component analysis [17], multidimensional scaling (MDS) [20], a force-directed algorithm [12], or non-linear dimensionality reduction such as LLE [29] and Isomap [31].

The second step is a cluster analysis of the underlying graph or the embedded pointset from step one. In this step, it is important to match the clustering algorithm to the embedding algorithm. For example, a geometric clustering algorithm such as k-means [24] may be suitable for an embedding derived from MDS, as the latter tends to place similar vertices in the same geometric region with good separation between clusters. On the other hand, with an embedding derived from a force-directed algorithm [12], a modularity based clustering [26] could be a better fit. The two algorithms are strongly related [27] and therefore we can expect vertices that are in the same cluster to also be physically close to each other in the embedding.

In the third step the two-dimensional embedding together with the clustering are used to create the map. In the final step, the countres are colored using a coloring algorithm to maximize color differences between neighboring countries. In the next two subsection we discuss the last two steps in more detail.

3.1 The mapping algorithm

Given the placement of the vertices from the first step, we want to create a map, with inner boundaries separating vertices not in the same cluster and outer boundaries perferably following the general outline of the point set. A naive approach for creating the map is to form the Voronoi diagram of the vertices based on the embedding information, together with four points on the corners of the bounding box; see Fig. 2(a). Such maps often have sharp corners, and angular outer boundaries. We can generate more natural outer boundaries by adding random points to the current embedding. A random point is only accepted if its distance from any of the real points is more than a preset threshold r away. This leads to boundaries that follow the shape of the point set. The randomness of the points on the outskirts also gives rise to some randomness of the outer boundaries, thus making them more map-like and natural; see Fig. 2(b). Furthermore, depending on the value of r, this step can also result in the creation of lakes (e.g., Fig. 1) in areas where vertices are far apart from each other. Nevertheless, some inner boundaries remain artificially straight.

Another undesirable feature is that the three “countries” all have roughly the same area, whereas we might often prefer some areas to be larger than others (e.g., due to the importance of the entities they represent). As an illustration, in Fig. 2, we assume that “node 1” is more important than the other two nodes, and use a larger label for that area. To make areas follow the shape of the labels, we first generate artificial points along the bounding boxes of the

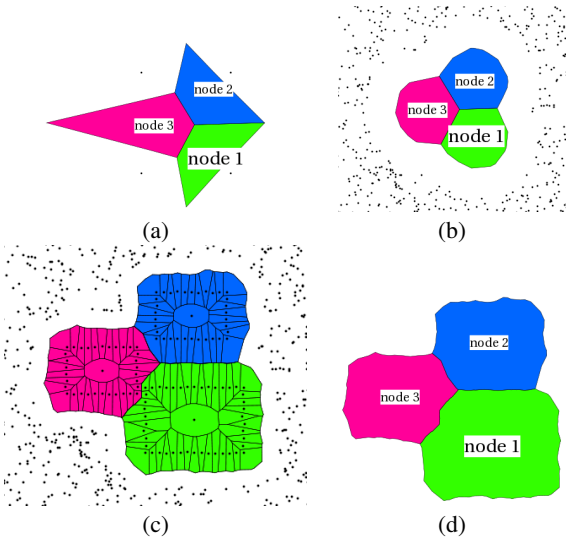

Figure 2: (a) Voronoi diagram of vertices and corners of bounding box; (b) better construction of outer boundaries through placement of random points; (c) Voronoi diagram of vertices and points inserted around the bounding boxes of the labels; (d) the final map.

labels; see Fig. 2(c)[2]. To make the inner boundaries less uniform and more map-like, we perturb these points randomly instead of running strictly along the boxes. Here Voronoi cells that belong to the same vertex are colored in the same color, and cells that correspond to the random points on the outskirt are not shown. Cells of the same color are then merged to give the final map in Fig. 2(d). Note that instead of the bounding boxes of labels, we could use any 2D shapes, e.g., the outlines of real countries, in order to obtain a desired look and proportion of area, as long as these shapes do not overlap.

We note that not all real maps have complicated boundaries. For example, the state boundaries in a map of the United States are mostly straight. We believe that maps with irregular boundaries, such as those in Europe, lead to more map-like results. The same underlying graph can lead to very different looking maps, depending on the way inner and outer boundaries are created. We find the European-style map of Figure 6 more appealing than the masp obtained with straight boundaries in Figures 8-7). But this is a matter of personal taste and our algorithm can generate maps of both styles.

When mapping vertices that contain cluster information, in addition to merging cells that belong to the same vertex, we also merge cells that belong to the same cluster, thus forming regions of complicated shapes, with multiple vertices and labels in each region. At this point we can add more geographic components to strengthen the map metaphor. For instance, in places where there is significant space between vertices in neighboring clusters, we can add lakes, rivers, or mountain ranges to the map to indicate the distance. These structures can all be formed by a similar insertion of random points in places where vertices are far away from each other.

In terms of complexity, the algorithm is scalable and has a time complexity of $O(|V|\log|V|)$. We first add n_a artificial points along the bounding boxes of the labels; typically, $n_a = 40|V|$. We then insert n_r random points of distance r away from any vertices and artificial points; usually n_r is set to between $|V|$ to $40|V|$, depending on the size of the graph. This step is carried out by first forming a quadtree of the vertices and artificial points, which takes time $O(|V|\log|V|)$, then testing whether a random point is within distance r of the set of vertices and artificial points. Each test takes $O(\log|V|)$ time, for $O(|V|\log|V|)$ time overall. We next compute a Delaunay triangulation of the points, which can be done in time $O(|V|\log|V|)$ [11]. Next we create the corresponding Voronoi diagram of all points and merge Voronoi cells that belong to the same cluster. This step requires $O(|V|)$ and thus the overall complexity of GMap is $O(|V|\log|V|)$, with a relatively large coefficient due to the large number of artificial and random points.

[2]Weighted Voronoi diagrams can be used to make the areas Voronoi cells proportional to their weight. However, we do not use this approach because we want the Voronoi cell to also contain a specific shape, e.g., the bounding box of a label.

3.2 The map coloring algorithm

In this subsection we consider the problem of assigning good colors to the countries in our maps. The Four Color Theorem states that only four colors are needed to color any map so that no neighboring countries share the same color. It is implicitly assumed that each country forms a contiguous region. However, this result is of limited use to us because countries in our maps are often not contiguous. For instance, in Fig. 6, a group of North American researchers are placed in a cluster made from three disjoint regions in light orange color in the southwest corner of the main continent. In cases where one cluster is represented by several disjoint regions we must use the same color for all regions to avoid ambiguity. Thus, four colors (or even five or six) are not enough.

In GMap we start with a coloring scheme from ColorBrewer [6], which typically has 5 easy to differentiate base colors, and generate as many colors as the number of countries by blending the base colors. As a result our color space is linear and discrete. Because of the blending, any two consecutive colors in the linear array of colors are similar to each other. When applying these colors to the map, we want to avoid coloring neighboring countries with such pairs of colors. With this in mind, we define the *country graph*, $G_c = \{V_c, E_c\}$, to be the undirected graph where countries are vertices, and two countries are connected by an edge if they share a non-trivial boundary. We then consider the problem of assigning colors to nodes of G_c so that the color distance between nodes that share an edge is maximized.

More formally, let C be the color space, i.e., a set of colors; let $c : V_c \to C$ be a function that assigns a color to every vertex; and let $w_{ij} \geq 0$ be weights associated with edges $\{i,j\} \in E_c$. Let $d : C \times C \to R$ be a color distance function. Define the vector of color distances along edges to be

$$v(c) = \{w_{i,j}\, d(c(i), c(j)) \mid \{i,j\} \in E_c\}.$$

Then we are looking for a color function that maximizes this vector with respect to some cost function. Two natural cost functions are:

$$\max_{c \in C} \sum_{\{i,j\} \in E_c} w_{i,j}\, d(c(i), c(j))^2 \quad \text{(2-norm)}$$

or

$$\max_{c \in C} \min_{\{i,j\} \in E_c} w_{i,j}\, d(c(i), c(j)) \quad \text{(MaxMin)}$$

The weights along the edges can be used to model the undesirable effect of two nearby but not connected countries having very similar colors by making the country graph a complete graph, and assigning edge weights to be the inverse of the distance between two countries.

Dillencourt et al. [9] investigated the case where all colors in the color spectrum are available. They proposed a force-directed model aimed at selecting $|V_c|$ colors as far apart as possible in the color space. However in our map coloring problem, for aesthetic reasons, we are limited to "map-like" colors, and our color space is discrete.

Therefore we model our coloring problem as one of vertex labeling, where our color space is $C = \{1, 2, \ldots, |V_c|\}$, and the color function we are looking for is a permutation that maximizes the labeling differences along the edges. The cost functions we consider are

$$\max_{c} \sum_{\{i,j\} \in E_c} w_{i,j}(c_i - c_j)^2, \quad \text{(2-norm)} \tag{1}$$

and

$$\max_{c} \min_{\{i,j\} \in E_c} w_{i,j}|c_i - c_j|, \quad \text{(MaxMin)}$$

where c_i is the i-th element of the vector c, and c is a permutation of $\{1, 2, \ldots, |V_c|\}$.

While we are not aware of prior work on this particular vertex labeling problem, the complementary problem of finding a permutation that *minimizes* the labeling differences along the edges is well-studied. For example, in the context of minimum bandwidth or wavefront reduction ordering for sparse matrices, it is known that the problem is NP-hard, and a number of heuristics [16, 21] were proposed. One such heuristic is to order vertices using the Fiedler vector. Motivated by this approach, we approximate (1) by

$$\max \sum_{\{i,j\} \in E_c} w_{i,j}(c_i - c_j)^2, \text{ subject to } \sum_{k \in V_c} c_k^2 = 1 \tag{2}$$

where $c \in R^{|V_c|}$. This continuous problem is solved when c is the eigenvector corresponding to the largest eigenvalue of the weighted Laplacian of the country graph, while the Fiedler vector (the eigenvector corresponding to the second smallest eigenvalue) minimizes the objective function above. Once (2) is solved, we use the ordering of the eigenvector as an approximate solution for (1). We call this algorithm SPECTRAL.

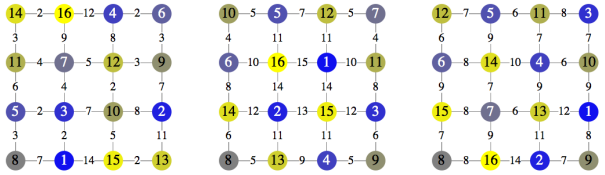

Figure 3: Coloring schemes RANDOM, SPECTRAL, and SPECTRAL+GREEDY. Each node is colored by the color index shown as the node label. Edge labels are the absolute difference of the endpoint labels.

Fig. 3 illustrates three coloring schemes on a 4×4 unweighted grid graph given 16 colors in the Blue-Yellow spectrum. A random assignment of colors, RANDOM, does reasonably well, but has one edge with a color difference of 2. SPECTRAL performs better, with the minimum color difference of 4. However there are still 2 edges with a color difference of only 4. It is easy to see that SPECTRAL can be improved (e.g., swapping colors 6 and 2 would improve the measurements according to both cost functions). With this in mind we propose GREEDY, a greedy refinement algorithm based on repeatedly swapping pairs of vertices, provided that the swap improves the coloring scheme according to one of the two cost functions. Starting from a coloring scheme obtained by SPECTRAL and applying GREEDY often leads to significant improvements.

So far we have been using a simple grid graph to illustrate the algorithms. The actual country graphs are usually more complex. Fig. 4 (left) gives the country graph corresponding to the map in Fig. 6, with color assignment given by SPECTRAL. There are two edges of color difference 1. Applying GREEDY algorithm guided by the MaxMin cost function to the result of SPECTRAL gives Fig. 4 (right). Now the minimum color difference along any edges is 4, a large improvement. This is indeed the coloring scheme used to create Fig. 6.

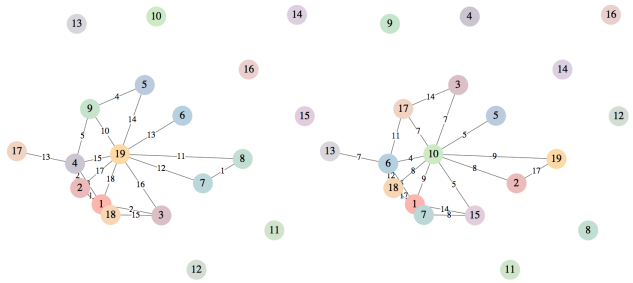

Figure 4: Applying coloring schemes for the country graph corresponding to the map in Fig. 6. Left: SPECTRAL. There are two edges of color difference 1. Right: SPECTRAL+GREEDY, the smallest color difference along any edges is now 4. Node labels are the color index given to the node, and edge label are the absolute difference of the node color index. Nodes are positioned at the center of the polygons in Fig. 6.

The GREEDY algorithm has a high computational complexity as we consider all possible $O(|V_c|^2)$ pairs of vertices for potential swapping. Since recomputing the cost functions can be done in time proportional to the sum of degrees of the pair on nodes considered for swapping, the overall complexity of GREEDY is $O(|V_c|^2 + |E_c|^2)$. Because the country graph G_C is typically much smaller than the underlying graph G, GREEDY is still quite fast and all maps in this paper were colored using SPECTRAL+GREEDY.

We note in passing that GREEDY is flexible enough to be used with any other cost function. For example, the MaxMin cost function could be modified to measure the distance between two colors in terms of their Euclidean distance in the RGB or Lab color space, instead of the index difference.

4 Results

In this section we apply the GMap algorithm to two applications. In the first application, we use GMap to visualizing collaboration relations. In the second application, we visualizing the "landscape" of books as implied by user purchase behavior at Amazon.

4.1 Collaboration Graph

This graph has authors as vertices and collaborations as edges. That is, there is an edge between two authors if they have collaborated on a paper. The graph has 509 vertices and 1517 edges. The largest component has 275 vertices and 784 edges, and thus contains about 54% of all authors. The data comes from the first 10 years of the Symposium on Graph Drawing, 1994-2004 [10]. We look at the first eight largest connected components. This graph is cumulative, in the sense that two authors are connected with an edge if they have written at least one joint paper in the first ten years of the symposium. Even when drawn with a high-quality scalable force-directed algorithm [15] and after applying a node-overlap removal step, the resulting graph looks more like a hairball than anything else; see Fig. 5.

On the other hand, the corresponding map, as shown in Fig. 6, seems much more "readable". The map shows one continent corresponding to the largest connected component and seven islands, corresponding to the seven largest remaining connected components. The continent contains about a dozen "countries" determined by the collaboration patterns. The size of each label is determined

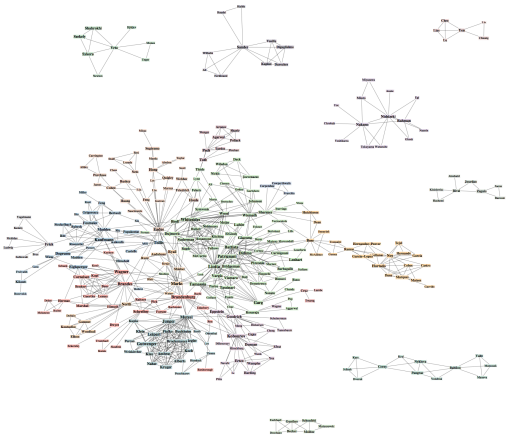

Figure 5: Graph Drawing author collaboration, 1994-2004.

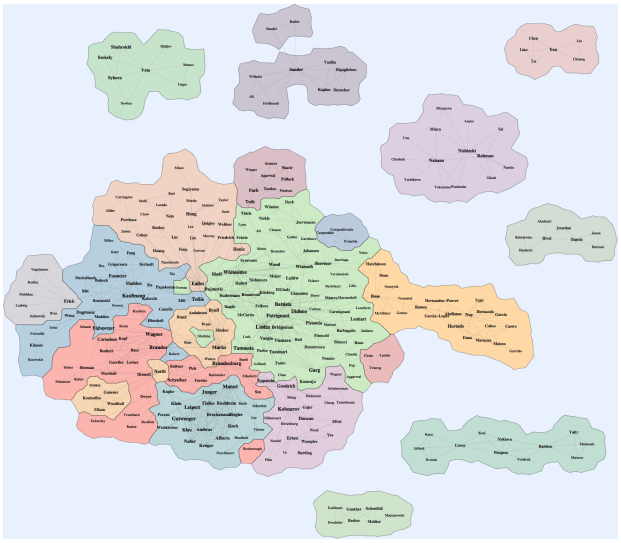

Figure 6: Collaboration graph drawn by GMap.

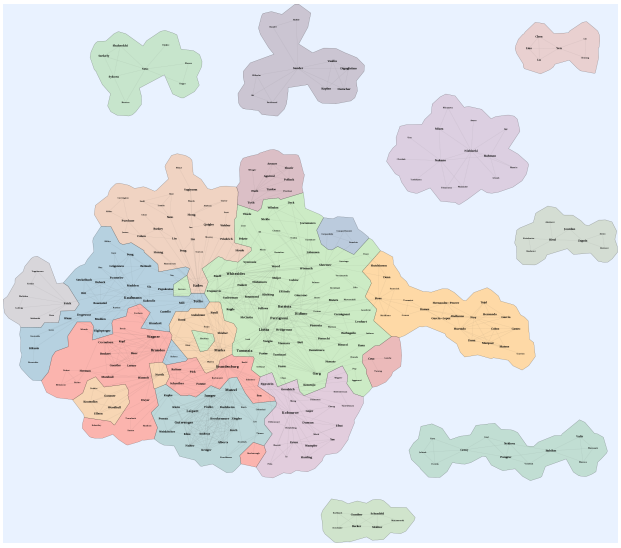

Figure 7: Map without interior artificial points.

by the logarithm of the number of publications and the edge thickness is similarly proportional to the number of collaborations. However, node weights and edge weights are not used in the layout calculations.

From Figure 6, it is easy to see that European authors dominate the main continent. Several well-defined German groups can be seen on the west and southwest coasts. A largely Italian cluster occupies the center, with an adjacent Spanish peninsula in the east. The northwest contains a mostly Australasian cluster. Two North American clusters are to be found in the southeast and in the southwest, the latter one made up of three distinct components. A combinatorial geometry cluster forms the northernmost point of the main continent. Most Canadian researchers can be found in the central Italian cluster and the Spanish peninsula. Northeast of the mainland lies a large Japanese island and southeast of the mainland there is a large Czech island. Northwest of the mainland is a Crossings Number island.

Figure 7 shows a map generated without adding artificial points around the labels, which results in more regular boundaries, when compared to the map in Figure 6. The size of the two maps in the paper makes it difficult to compare them, but the differences are easy to see on the screen when the images are zoomed in. We found that the map on Figure 6 with more European-style borders was more appealing, but our algorithm can generate maps of both styles. The map generated without adding random points to define the outer boundaries is even more noticably un-map-like; see Fig. 8.

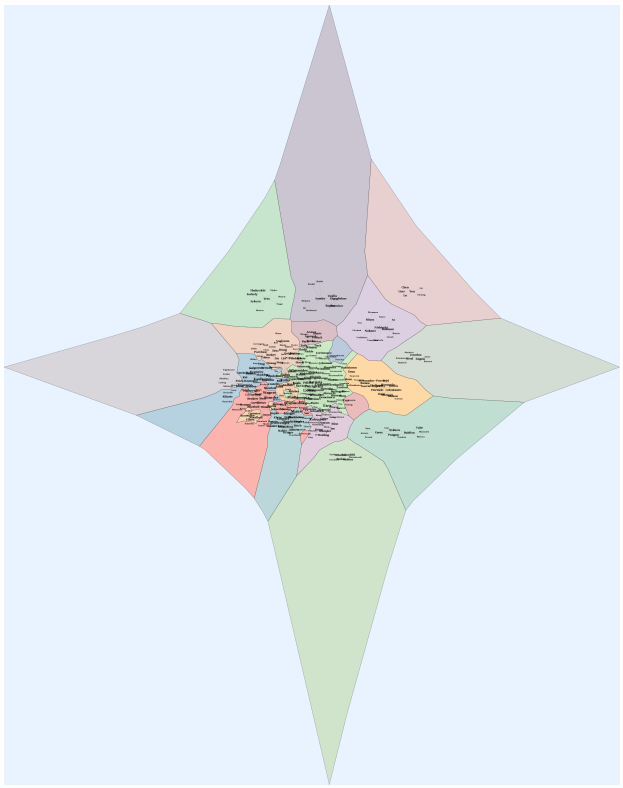

Figure 8: Map without outer artificial points.

4.2 BookLand Maps

Many e-commerce websites provide recommendations to allow for exploration of related items. Traditionally this is done in the form of a flat list. For example, Amazon typically lists around 5-6 books

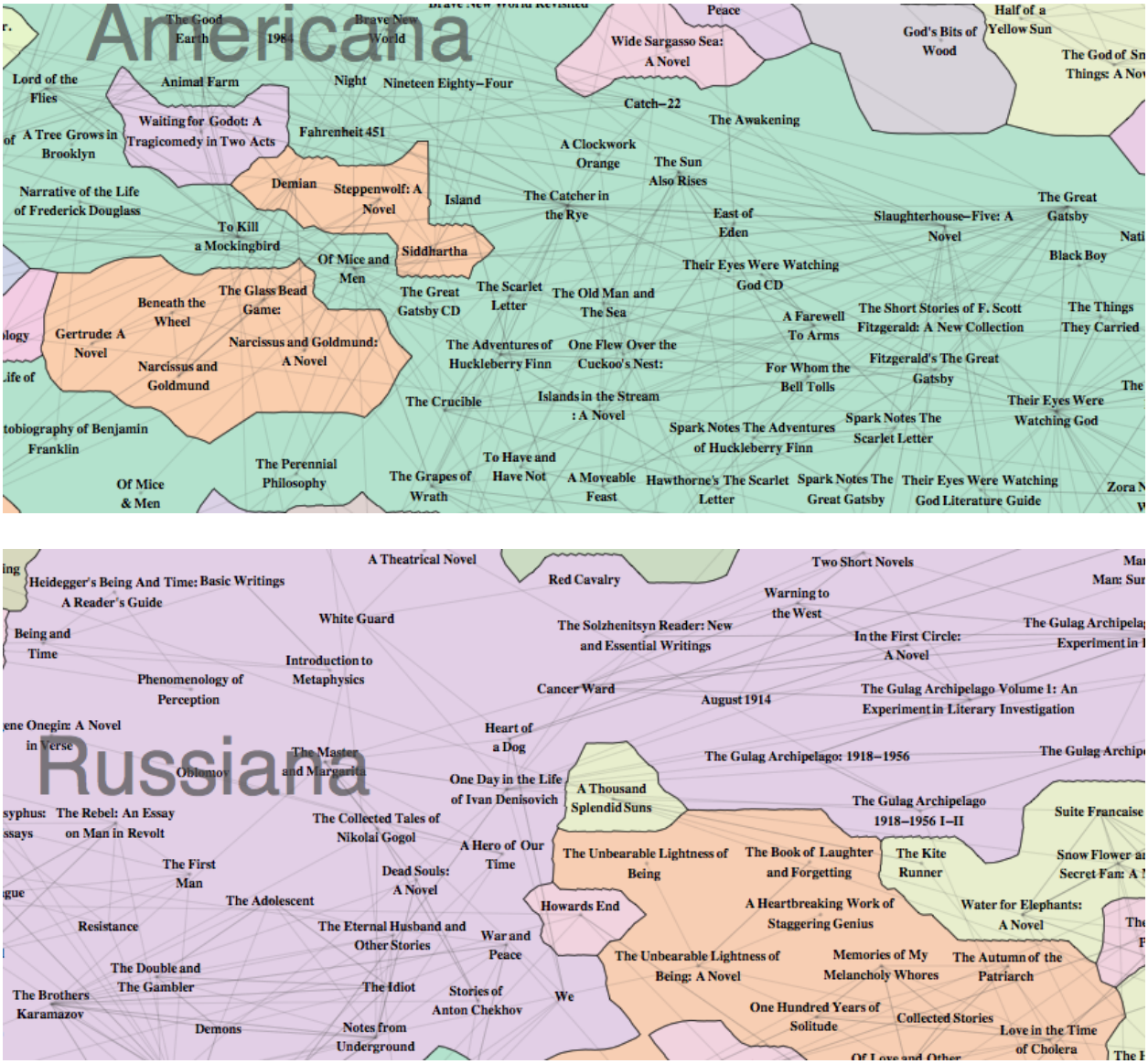

Figure 9: Two of the central clusters in BookLand.

under "Customers Who Bought This Item Also Bought", with a clickable arrow to allow a customer to see further related items.

Instead of a flat list, which provides a very limited view of the neighborhood, there have been attempts to convey the underlining connectivity of the products through graph visualization. For example, *TouchGraph* [3], has an Amazon browser which shows a graph defined on a small neighborhood surrounding the book of interest. None of the existing approaches, however, gives a comprehensive view of the relationship and the clustering structures.

Using our GMap algorithm, we obtained the map in Fig. 1. The underlying data is obtained with a breadth-first traversal following Amazon's "Customers Who Bought This Item Also Bought" links, starting from the root node, George Orwell's *1984*. Links are followed up to a distance of 12 from the root node. We then trim the graph by keeping only vertices of distance 9 or less from the root vertex. We further merge nodes that represent the same book, but with different publishers or different bindings, by matching books with the same title. As can be seen by the 5 versions of Chinua Achebe's *Things Fall Apart* in the central cluster, we are not always successful because the titles can vary slightly. The underlying graph for this map contains 913 vertices and 3410 edges. With an average degree of nearly eight, peripheral vertices in this map have only a handful of edges while central vertices have more than 20 immediate neighbors, reflecting the directed nature of this graph. We next examine several of the "countries" in the map in more detail.

Americana: Somewhat surprisingly, George Orwell's *1984* along with *Animal Farm* ended up in the west corner of a region populated mostly by American writers. Britain is also represented by William Golding's *The Lord of the Flies* and Aldous Huxley's *Brave New World* along with Anthony Burgess's *Clockwork Orange*, which connect the British corner of the region to the main part dominated by 20th century American classics. Ray Bradbury's *Fahrenheit 451* and Salinger's *Catcher in the Rye* provide a transition to a variety of well-known novels: Steinbeck's *Grapes of Wrath* and *Of Mice and Men*, Ernest Hemingway's *For Whom the Bell Tolls* and *The Old Man and the Sea*, F. Scot Fitzgerald's *Great*

Table 1: CPU time (in seconds) for the GMap algorithm.

Graph	|V|	ncluster	npoly	CPU
Fig. 6	340	19	27	0.26
Fig. 1	913	26	95	0.95
Movie map [2]	1981	41	108	2.65

Gatsby, Harper Lee's *To Kill a Mockingbird* and Ralph Ellison's *Invisible Man*, Joseph Heller's *Catch 22*, Kurt Vonnegut's *Slaughterhouse Five*, Ken Kesey's *One Flew Over a Cuckoo's Nest*. Some 19th century novels can also be found here: Nathaniel Hawthorne's *Scarlet Letter* and Mark Twain's *Adventures of Huckleberry Finn*.

Victoriana: To the southwest of Americana is a region dominated by Dickens, Austen and Bronte novels. Starting with *A Tale of Two Cities*, *Great Expectations* and *Oliver Twist* in the north and going through *Jane Eyre*, *Pride and Prejudice*, *Sense and Sensibility* and *Wuthering Heights* in the middle, the region ends with more Dickens' books in the southwest (*The Pickwick Papers*) and George Elliot novels in the southeast (*Middlemarch*).

Russiana: To the north of Americana lies one of the largest countries in BookLand, dominated by Russian literature and history. The core contains classic novels by Dostoyevsky (*Crime and Punishment, The Brothers Karamazov*), Tolstoy (*War and Peace, Anna Karenina*), and Solzhenitsyn (*The Gulag Archipelago, Cancer Ward*). In the northern part of the region is a collection of books about Russia and Russian history: *Stalin: The Court of the Red Tsar, Khrushchev: The Man and His Era* and *Potemkin: Catherine the Great's Imperial Partner*. In the west there is a cluster of Albert Camus books (*The Stranger, The Plague, The Fall*), all well connected with the Russian classics.

Graecoromania: Another large region to the west of Americana contains a diverse collection of Graeco-Roman books. History books by Thucydides, Plutarch, Livy, Suetonius, Salust share the region with philosophy by St. Augustine, Plato, Socrates, and Aristotle. Greek theater is represented by Aristophanes, Aeschylus, Euripides, Sophocles and epic poetry by Homer and Virgil.

Shakespearea: Very centrally located, neighboring Victoriana, Americana, Russiana, and Graecoromania lies the land of Shakespeare. It is not surprising that nearly all tragedies, comedies and histories are present but it is interesting to observe what non-Shakespeare books are in this region: Chaucer's *Canterbury Tales*, Tennyson's *Idyls of the King*, Dante's *Divine Comedy*, *One Thousand and One Arabian Nights, Beowulf* and *The Adventures of Robin Hood*.

4.3 Running Time

In Table 1, we show the running times of GMap on a few input graphs. We only report the CPU time for mapping, not the coloring, since unlike the mapping algorithm which is implemented in C efficiently, the coloring part of our algorithm is implemented in Mathematica, and is not optimized. In addition, the amount of time spent on coloring is proportional to the number of clusters, which is much smaller than the size of the graph.

The two maps in this paper are both generated in less than one second. A larger map of the popular movies on NetFlix (not shown in this paper but is available at [2]), mapping takes 2.65 seconds. As a reference point, GMap generated a map for a graph with 440,000 nodes in four minutes. All experiments were carried out using a single thread on a Linux machine with 16 Intel Xeon processors, each with 4 cores running at 2.4 GHz, with 16 GB memory per processor.

4.4 Map defragmentation

While the force-directed algorithm tends to work well with modularity based clustering, in the sense that vertices in the same cluster often form a contiguous region, this is not always the case.

There are two cases for a possible mismatch of these two algorithms. In the first case, the connectivity information necessitates discontiguity if connected vertices are to be placed closeby. For example, in Figure 5, the light orange cluster of researchers from AT&T in the SW is separated from a region NE of the AT&T cluster, with one research "North" who is from the same organization. The reason is that North co-authored papers with many non-AT&T authors and if he is to be placed with the rest of the AT&T group this would lead to longer edges to the non-AT&T authors.

In the second case, the problem arises in the overlap removal stage. Often after force-directed embedding there are many overlapping vertex labels and an overlap-removal algorithm is used to remove then. As a result vertices can move sufficiently far from their original positions to break an initially contiguous region. In many cases, such as the maps in this paper, a small degree of discontiguity does not hinder the overall visualization. However, a high degree of discontiguity can make the map too fragmented, necessiating a defragmentation step.

Table 1 shows the number of clusters, and the number of polygons, for the two maps in Figures 6 and 1. If the number of polygons is the same as the number of clusters, then all vertices in a given cluster are also in the same polygon, and each country is contiguous. In general there are more polygons than the number of clusters and the ratio gives us a measure of how fragmented the maps are.

If defragmentation is needed, such a step can be performed after the clustering and before the embedding steps of GMap. Specifically, given the clustering, we can modify the underlying graph with the goal of generating the embedding in which vertices in the same cluster are placed closer together. Instead of a traditional force-directed embedding step on the input graph, we modify the edge weights as follows: edge lengths of intra-cluster edges are set to 1 and intra-cluster edges to a number $\alpha > 1$. The embedding is obtained using a stress majorization algorithm [13] to give an embedding that takes into account the clustering information.

Figure 10 shows the result of this defragmentation algorithm applied to Figure 5 with $\alpha = 50$. Compared with force directed embedding which gives 27 polygons (see Table 1), now there are only 19 polygons, and every country is contiguous. However this contiguity comes at the expense of some long edges.

In applying our GMap algorithm to many graphs from different domains, we found that for graphs of up to a few thousand vertices fragmentation is not a serious problem. All the maps in this paper, with the exception of Figure 10, are generated without defragmentation.

5 Conclusion and Future Work

In this paper we described GMap, an efficient algorithm for drawing graphs as geographic maps. Using a number of structurally different graphs and graphs of different sizes, we illustrated the aesthetic appeal of the map metaphor for displaying underlying structures and clustering information. While the approach of visualizing relational information with the aid of geographical maps is general, here we showed one particular implementation where a scalable force-directed layout algorithm was coupled with a modularity-based clustering algorithm. Exploring different combinations of layout and clustering algorithms is one clear direction for future work.

Informal studies of the way users interact with traditional graph drawings and maps of the same data seem to indicate different interaction patterns. Specifically, users are more likely to stop by and examine a poster of a map that the exact same data drawn as a graph.

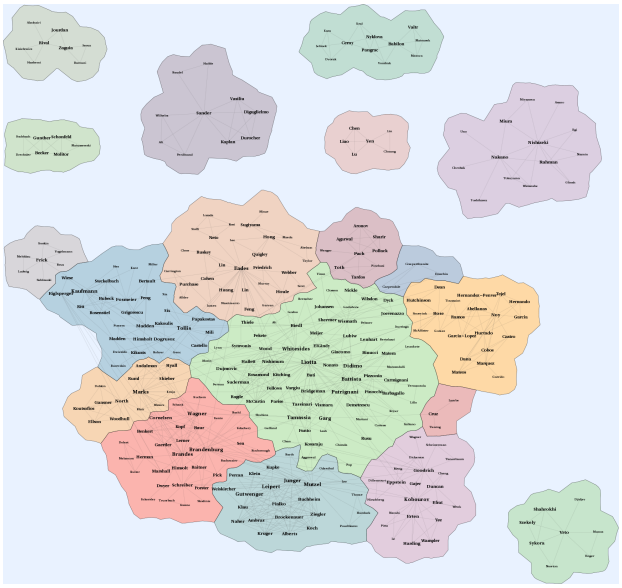

Figure 10: Author collaboration graph for the GD conference, 1994-2004, using the defragmentation algorithm.

Moreover, users tend to spend significantly longer time studying the maps, and they find non-trivial structural information without any prompting. For example, several viewers of BookLand observed that the "gateway" to Fringistan is Ayn Rand's *Atlas Shrugged*. We plan to perform formal user studies of the interaction with graphs and maps.

There are practical and theoretical obstacles to obtaining "perfect" maps, that is, maps that do not omit or distort the underlying information. However, a similar drawback plagues any 2-dimensional representation of data that is not 2-dimensional, including the standard geographical maps of Earth. Clearly, in dense graphs it is impossible to realize all graph adjacencies as neighboring countries. For example, with 8 countries we can have at most 18 pairwise neighbors (from Euler's formula for planar graphs), possibly forming some unavoidable "false negative associations". It is easier to deal with "false positive associations". Such an association between two countries can be formed if they are physically adjacent in the map but there is no strong relationship between the objects in the two countries. One way to alleviate such a problem is to add "rivers" or "fords" along such borders near the coasts and "mountain ranges" inland, to convey that the two sides are close but not strongly connected.

Finally, while our algorithm is efficient and can handle large graphs, the resulting maps look best on large wall-sized posters and display walls. To make such maps more useful for exploration of large data sets on commonly available media, we plan to develop an interactive interface that can zoom and pan easily on the maps. We are exploring the combination of GMap and a map service, such as OpenLayer (`http://openlayers.org`), as an effective tool for displaying relations among a large collection of objects, such as database search results. A prototype is available at `http://www.research.att.com/~yifanhu/MAPS/imap.html`.

Acknowledgments

We would like to thank Stephen North for helpful discussions.

References

[1] Karte des Bücherlandes. `strangemaps.wordpress.com/2009/04/08`.

[2] `www.research.att.com/~yifanhu/MAPS/`.

[3] `www.touchgraph.com/TGAmazonBrowser.html`.

[4] Map of online communities. `xkcd.com/256`.

[5] K. Boyack, R. Klavans, and K. Borner. Mapping the backbone of science. *Scientometrics*, 64(3):351–374, 2005.

[6] C. Brewer. Colorbrewer - selecting good color schemes for maps. `www.colorbrewer.org`.

[7] M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Press, 1999.

[8] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994.

[9] M. B. Dillencourt, D. Eppstein, and M. T. Goodrich. Choosing colors for geometric graphs via color space embeddings. In *14th Symposium on Graph Drawing (GD)*, pages 294–305, 2006.

[10] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. GraphAEL: Graph animations with evolving layouts. In *11th Symposium on Graph Drawing*, pages 98–110, 2003.

[11] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[12] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force directed placement. *Software - Practice and Experience*, 21:1129–1164, 1991.

[13] E. R. Gansner, Y. Koren, and S. C. North. Graph drawing by stress majorization. In *Proc. 12th Intl. Symp. Graph Drawing (GD '04)*, volume 3383 of *LNCS*, pages 239–250. Springer, 2004.

[14] X. He. On floor-plan of plane graphs. *SIAM Journal of Computing*, 28(6):2150–2167, 1999.

[15] Y. F. Hu. Efficient and high quality force-directed graph drawing. *Mathematica Journal*, 10:37–71, 2005.

[16] Y. F. Hu and J. A. Scott. A multilevel algorithm for wavefront reduction. *SIAM Journal on Scientific Computing*, 23:1352–1375, 2001.

[17] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.

[18] P. Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig. Math.-Phys. Klasse*, 88:141–164, 1936.

[19] T. Kohonen. *Self-Organizing Maps*. Springer, 2000.

[20] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Press, 1978.

[21] G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, 35:1–32, 1997.

[22] C.-C. Liao, H.-I. Lu, and H.-C. Yen. Compact floor-planning via orderly spanning trees. *Journal of Algorithms*, 48:441–451, 2003.

[23] I. Y. Liao, M. Petrou, and R. Zhao. A fractal-based relaxation algorithm for shape from terrain image. *Computer Vision and Image Understanding*, pages 227–243, 2008.

[24] S. Lloyd. Last square quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[25] F. K. Musgrave. *Methods for Realistic Landscape Imaging*. PhD thesis, Yale Univeristy, 1993.

[26] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577–8582, 2006.

[27] A. Noack. Modularity clustering is force-directed layout. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 79, 2009.

[28] E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24(2):292–296, 1934.

[29] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[30] B. Shneiderman. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[31] J. B. Tenenbaum, V. V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[32] J. R. R. Tolkien. *The Shaping of Middle-Earth*. Houghton Mifflin Harcourt, 1986.

[33] M. Weskamp. Newsmap. `marumushi.com/apps/newsmap`, 2004.