Visualizing Graphs and Clusters as Maps

Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov ■ *AT&T Labs Research*

Information visualization is essential in understanding large datasets. Traditional dimensionality reduction techniques often don't capture the underlying structural information, clustering, and neighborhoods well. GMap, a practical algorithmic framework for visualizing relational data with geographic-like maps, is effective in various domains.

Researchers have considered many useful techniques for visualizing abstract datasets. Often, they use graphs made of nodes (or vertices) and links (or edges) to capture the relationships between objects, and graph drawing lets us visualize such relationships. Typically, points in 2D or 3D space represent nodes; lines between the corresponding vertices represent edges. For an example of a relational dataset, consider the Amazon.com graph, in which nodes represent books, and an edge between two books denotes that people who purchased one of the books also bought the other.

High-dimensional datasets, on the other hand, are often visualized as point clouds in 2D or 3D space. For example, in the listening pattern of users of Last.fm (an Internet radio and music community website), a long vector represents each user. The vector's dimension is the same as the website's number of available musicians, and each element's value is proportional to the number of times a user listens to the corresponding musician. In a point-cloud visualization, each point is a user, and two points are close to each other if the two users have similar musical tastes.

Unfortunately, comprehending these standard approaches of node-link diagrams and point cloud representations often requires considerable effort. If we want to make the data accessible to those outside computer science and statistics, we must provide more compelling drawings. This is certainly true for the ordinary user, who might be curious about Amazon's recommendations. However, it's also true for, say, the biologist who wishes to understand a statistician-created visualization of a biological experiment.

When large datasets are drawn as node-link graphs or point clouds, we can sometimes observe a visual similarity to geographic maps. For example, several small connected components next to a much larger component suggest several small islands next to a big continent. We took the next step in visualizing data directly as a geographic map. A map representation is familiar and intuitive; most people are familiar with maps, and well-drawn maps can provide hours of enjoyable exploration. Many people exploit this familiarity with maps and manually create maplike representations that portray relations among abstract concepts (for more information, see the "Related Work in Information Visualization" sidebar). We wanted to automate the process that begins with the data and ends with a drawing of a map. We had two main competing goals:

- be faithful to the data by maintaining the inherent structure and relationships, without adding or implying nonexistent structure, and
- obtain a final result that looks like a map, using standard cartographic conventions.

Our attempt to address these two goals led to GMap, a framework for map-based data representation.

GMap

GMap lets us generate maplike representations from an abstract dataset. Specifically, given a high-dimensional dataset or a graph with edge weights (for example, the Amazon.com graph), it produces a drawing with a maplike look. This drawing has countries that enclose similar objects, outer bound-

 Published by the IEEE Computer Society

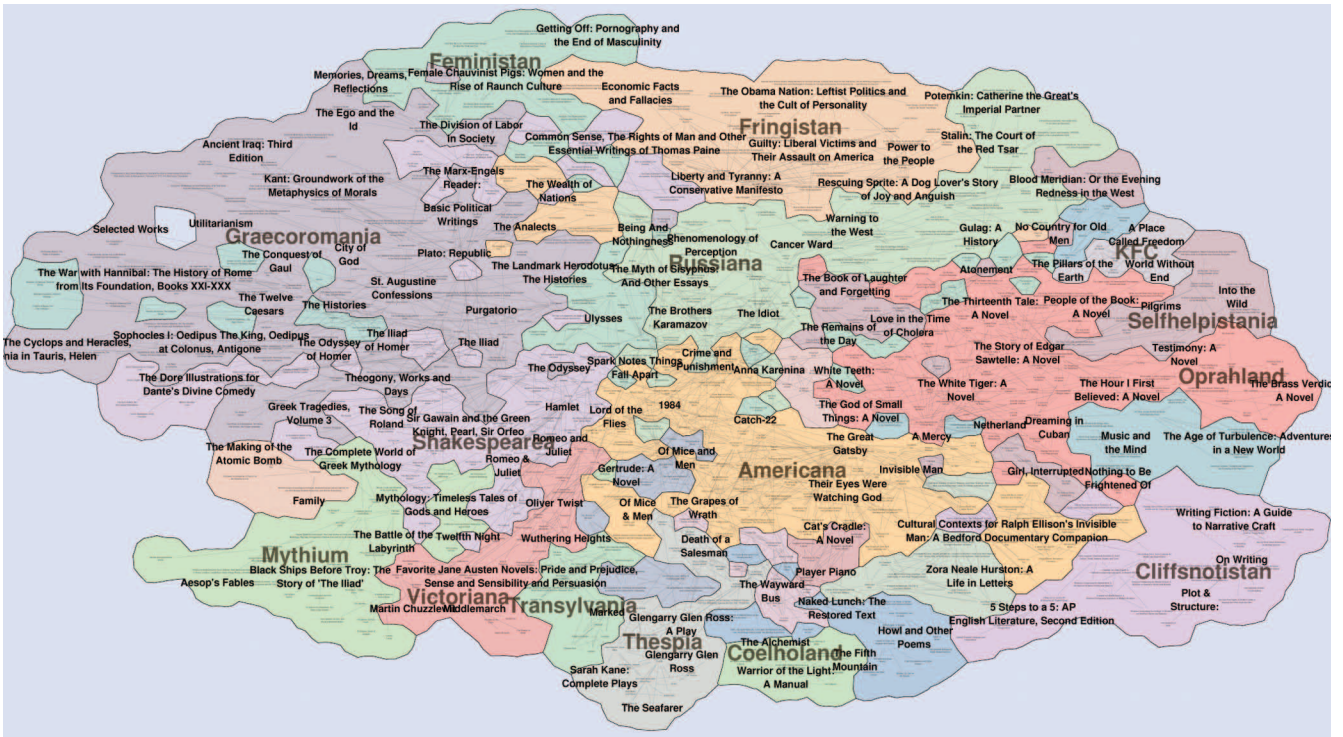

Figure 1. A map of books related to George Orwell's *1984* from Amazon.com. Two books are placed close to each other if people who purchase one also tend to purchase the other. Each country represents a group of highly related books based on a cluster analysis.

aries that follow the vertex set's outline, and inner boundaries that have the twists and turns found in real maps. A typical example in Figure 1 shows just under 1,000 books. Our maps also can have lakes, islands, and peninsulas, like in real maps.

GMap is a framework rather than a specific algorithm. It contains four main steps: the first two are achievable with existing algorithms, and we propose new algorithms for the last two.

First, we take a graph or high-dimensional dataset as input and embed it in a plane. The statistics and scientific-modeling communities have extensively explored this problem and provide many ways of doing this. Possible embedding algorithms include principal component analysis, multidimensional scaling (MDS), force-directed algorithms, and nonlinear dimensionality reductions such as locally linear embedding and Isomap.

Second, we take this collection of points in the plane and aggregate them into clusters. Matching the clustering algorithm to the embedding algorithm is important. For example, a geometric clustering algorithm such as *k*-means could be suitable for an embedding derived from MDS, because the latter tends to place similar points in the same geometric region with good separation between clusters. On the other hand, with an embedding derived from a force-directed layout, a modularity-based clustering[1] could be a better fit. The two algorithms are strongly related, so we can expect vertices in the same cluster to be physically near each other in the embedding.

Third, we use the 2D embedding together with the clustering to create the actual map by delineating country boundaries, carving continental outlines, and separating islands from continents. You can do this with the help of plane-partitioning techniques such as Voronoi diagrams, augmented with new algorithmic techniques to ensure realistic outer and inner boundaries.

Finally, we add additional graphical attributes to the drawing to enhance its clarity, serve as keys to the abstract data, or simply increase the aesthetic appeal. This involves assigning an appropriate set of colors to the various regions. We propose a spectral algorithm that maximizes color difference between neighboring countries. In addition, we could add mountains to the map or overlay it with a heat map to indicate scalar values associated with geographic positions.

Although hundreds of researchers have tackled the first two steps, the last two steps are new. Here, we explain informally how we combine the embedding and clustering information to make the map appealing. For technical details and more references, see our previous paper.[2]

Making the Map

Given the placement and clustering of the points from the first two steps, we want to create a map,

Related Work in Using Maps for Information Visualization

Much geography research addresses accurately and appealingly representing a given geographic region, or redrawing an existing map subject to additional constraints. We find examples of the first kind of problem in traditional cartography, such as the 1569 Mercator projection of the sphere onto 2D Euclidean space. An example of the second kind of problem is cartograms, which redraw a map so that the geographical areas are proportional to some metric. This idea dates back to 1934 but is still popular (for example, the *New York Times*' red-blue maps of the US, showing the presidential election results in 2000 and 2004 with states drawn proportionally to population).

Information visualization research has produced many ways to represent data, some even adopting the name "map." Most of these have little visual connection with geographic maps. The map of science uses vertex coloring in a graph drawing to provide an overview of the scientific landscape on the basis of journal article citations.[1] Treemaps,[2] squarified treemaps,[3] and news maps represent hierarchical information by means of space-filling tilings, allocating area in proportion to some metric.

Concept maps are diagrams showing relationships among concepts.[4] Somewhat similar are cognitive maps and mind-maps that represent words or ideas linked to and arranged around a central keyword.

In self-organizing maps, an unsupervised learning algorithm places objects on a 2D grid such that similar objects are close.[5] Unlike GMap (see the main article), a self-organizing map creates a "map" by coloring the grid's cells according to a feature value. So, it operates on a discrete grid space without a clear inner boundary between "countries." Furthermore, the grid tends to fit a rectangular box, so that the point set's overall outline often follows that shape.

Also related is research on visualizing subsets of a set of items using geometric regions to indicate the grouping. Heorhiy Byelas and Alexandru Telea use deformed convex hulls to highlight areas of interest in Unified Modeling Language diagrams.[6] Christopher Collins and his colleagues use "bubble sets," based on isocontours, to depict multiple relations among a set of objects.[7] Paolo Simonetto and his colleagues automatically generate Euler diagrams—a standard way, along with Venn diagrams, to visualize subset relationships.[8] Apart from differences in the algorithms used to generate regions, these methods differ from ours in that they create overlapping regions, with the goal of faithfully representing set relationships. In contrast, we take the map metaphor seriously, assuming that regions don't overlap and aiming for cartographic verisimilitude. However, our approach can also help visualize sets (see Figure 5 in the main article).

Representing imagined places on a map as if they were real countries has a long history (for example, J.R.R. Tolkien's map of Middle Earth and Alphons Woelfle's Bücherlandes map date to the 1930s). More recent drawings include maps of programming language concepts and online communities. Although most such maps are generated ad hoc by hand and aren't based strictly on underlying data, they're often visually appealing.

Generating synthetic geography has a large literature, connected to its use in computer games and movies. Most research relies on variations of a fractal model. These techniques could provide additional photorealism, and we might use them in future research.

References

1. K. Boyack, R. Klavans, and K. Borner, "Mapping the Backbone of Science," *Scientometrics*, vol. 64, no. 3, 2005, pp. 351–374.
2. B. Shneiderman, "Tree Visualization with Tree-Maps: A 2-D Space-Filling Approach," *ACM Trans. Graphics*, vol. 11, no. 1, 1992, pp. 92–99.
3. M. Bruls, K. Huizing, and J. van Wijk, "Squarified Treemaps," *Data Visualization 2000: Proc. Joint Eurographics and IEEE TCVG Symp. Visualization 2000*, Springer, 1999, pp. 33–42.
4. J.D. Novak and A.J. Cañas, *The Theory Underlying Concept Maps and How to Construct Them*, tech. report, Florida Inst. for Human and Machine Cognition, 2006.
5. T. Kohonen, *Self-Organizing Maps*, Springer, 2000.
6. H. Byelas and A. Telea, "Visualization of Areas of Interest in Software Architecture Diagrams," *Proc. 2006 ACM Symp. Software Visualization* (SoftVis 06), ACM Press, 2006, pp. 105–114.
7. C. Collins, G. Penn, and S. Carpendale, "Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 6, 2009, pp. 1009–1016.
8. P. Simonetto, D. Auber, and D. Archambault, "Fully Automatic Visualisation of Overlapping Sets," *Computer Graphics Forum*, vol. 28, no. 3, 2009, pp. 967–974.

with inner boundaries separating points not in the same cluster and outer boundaries preferably following the point set's general outline. A naive approach for creating the map is to form the Voronoi diagram of the vertices on the basis of the embedding information, together with four points on the corners of the bounding box (see Figure 2a). 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. Our method accepts a random point only if its distance from any real point is more than some preset threshold. We can efficiently implement

this step using a suitable space-decomposing data structure, such as a quadtree. This leads to boundaries that follow the point set's shape. In addition, the randomness of the points on the outskirts gives rise to some randomness of the outer boundaries, making them more maplike (see Figure 2b). Furthermore, depending on the threshold's value, this step can also create lakes and fjords where vertices are far from each other. Nevertheless, some inner boundaries remain artificially straight.

At this point, the "countries" all have roughly the same area (see Figure 2b), whereas we might prefer some areas to be larger than others (for example, to signify the importance of the entities they represent). To illustrate this, in Figure 2, we assume that node 1 is the most important, and we use a larger label for it. (We could use a weighted Voronoi diagram to make each Voronoi cell's area proportional to its weight. We don't use this approach, however, because we want the Voronoi cell to also contain a specific shape.)

To make areas follow the label's shape, we first generate artificial points along the label's bounding boxes (see Figure 2c). To make the inner boundaries less uniform and more maplike, we perturb these points randomly instead of running strictly along the boxes. Here, a vertex's Voronoi cells are the same color, and cells corresponding to the random points on the outskirt aren't shown. We then merge same-color cells to give the final map (see Figure 2d). Instead of a label's bounding boxes, we could use any 2D shapes (such as real countries' outlines) to obtain a desired look and proportion of area, as long as these shapes don't overlap.

Not all real maps have complicated boundaries. For example, boundaries of western US states often have long, straight sections. We believe that irregular boundaries are more typical of historical geographic boundaries and lead to more maplike results. However, this is just our personal taste; our technique can generate maps of both styles.

When mapping vertices containing cluster information, besides merging cells belonging to the same vertex, we merge cells belonging to the same cluster. This forms 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, where significant space exists between vertices in neighboring clusters, we can add lakes, rivers, or mountain ranges to the map to indicate the distance.

With the regions determined, we have a representation of the data in which closely related objects, as determined by the graph topology and

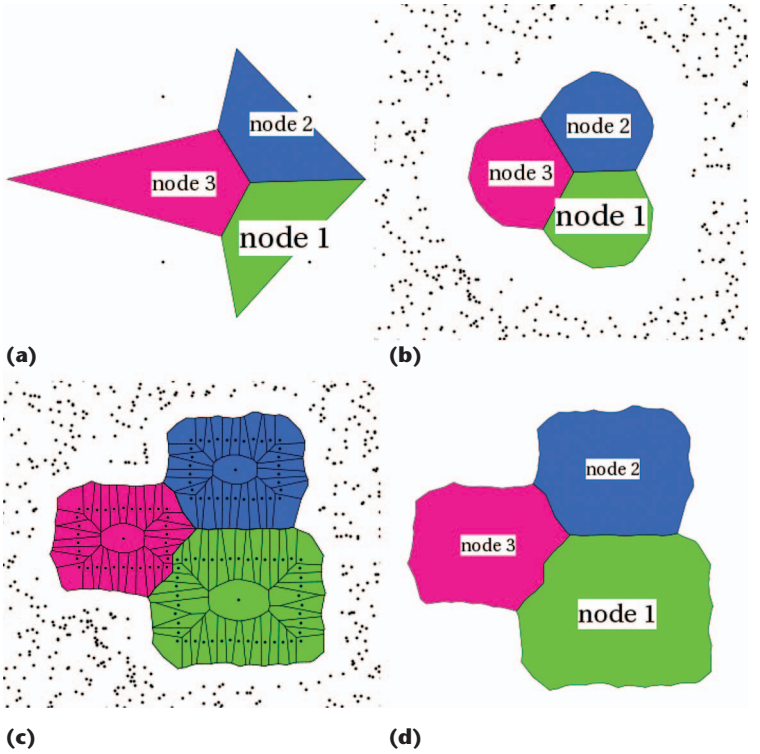

Figure 2. Creating a map: (a) a Voronoi diagram of vertices and corners of a bounding box, (b) better construction of outer boundaries through placement of random points, (c) a Voronoi diagram of vertices and points inserted around the labels' bounding boxes, and (d) the final map.

possibly edge weights, are drawn closely together. We then use this geometric information to discover clusters among the objects. To emphasize the clusters, we represent each as a collection of geometric regions.

Figure 3 compares a typical node-link graph layout and a GMap map of the same data. (Indeed, the graph layout is the first step of the GMap layout.) The data represents the graph of author collaborations between 1994 and 2004 at the Symposium on Graph Drawing. The node-link graph layout (see Figure 3a) exhibits the connected components and indicates closely related nodes by proximity, but only hints at cluster structure. In the GMap version (see Figure 3b), the cluster structure is obvious. Coloring the nodes in the node-link drawing would still only imply the clusters. The GMap figure makes the clusters explicit and indicates strong cluster relations where two clusters share a border.

When projecting high-dimensional data into low-dimensional space, distance distortion is inevitable, and the resulting figure will often have anomalies and distortions. So, some strongly related objects might be separated by seemingly unrelated objects. For example, in Figure 3b, we see a cluster of North American authors split into

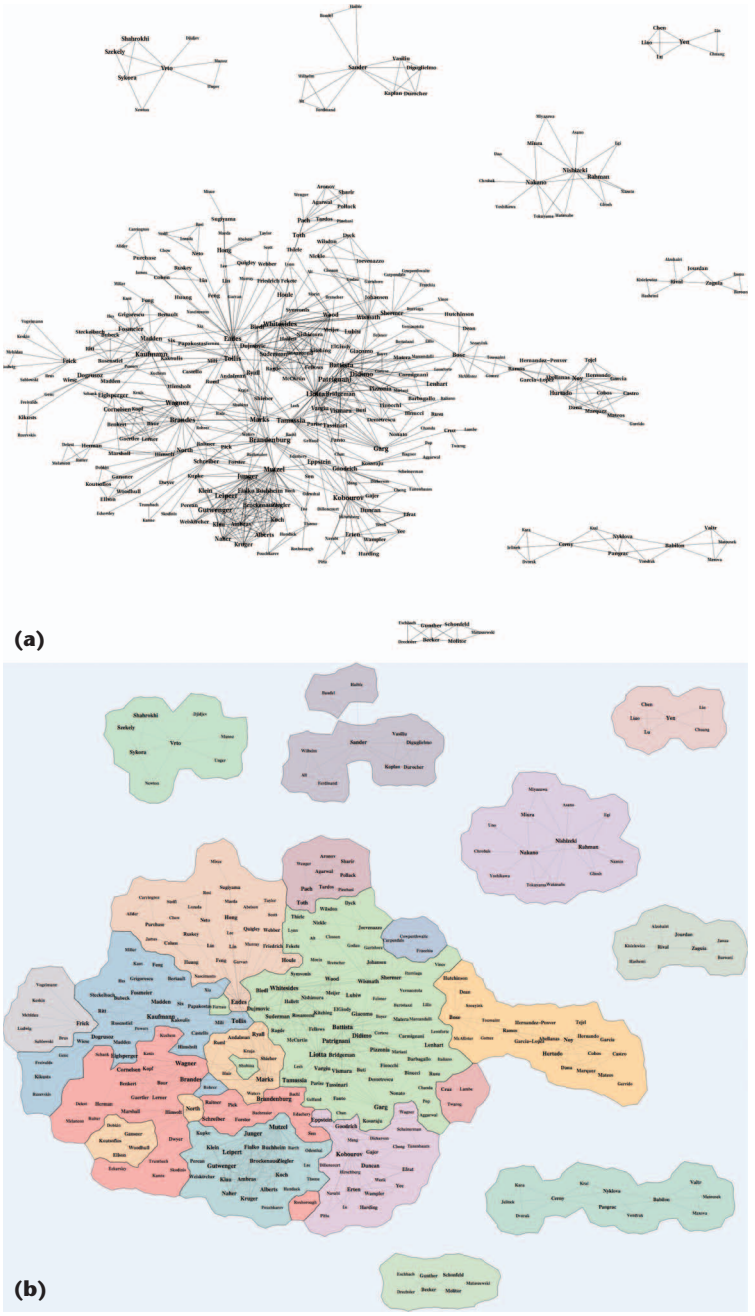

Figure 3. Comparing (a) a node-link drawing to (b) the GMap representation. Compared with the node-link drawing, the cluster structure is much more evident in the GMap drawing.

the three beige components in the map's southwest part. The authors in the upper-right component have collaborated much more with European authors; those in the bottom-left component, much less. The singleton component in the middle reflects that the author, North, collaborated both with the European authors and the authors in the bottom-left component. Such fragmentations are inherent in the embedding and clustering algorithms used in the first two steps. We've proposed ways to use the clustering information to adjust the layout, so that the regions of countries are more contiguous.[2] However, this loses some relational information captured in the original embedding.

We believe that the maplike visualization better captures some of the cluster structure while providing an illustration that's more attractive to the typical user than the traditional scatterplots and node-link diagrams. Although we haven't yet run a formal user study, anecdotal evidence from our customers and colleagues corroborates this claim.

Coloring the Map

Annotating the map with graphical attributes involves assigning good colors to the countries. The four color theorem states that you need only four colors to color any map, so that no neighboring countries share the same color. However, because this theorem implicitly assumes that each country forms a connected region, it's less useful for us, because countries in our maps aren't always connected. So, we must use one color for each cluster to avoid ambiguity. The estimated number of colors an average human can discriminate, when color pairs are presented side by side, ranges from tens of thousands to one million. On the other hand, comparing similar colors takes much more effort. A further limiting factor is that 5 percent of males are color blind, which rules out certain coloring schemes. Finally, the limited palette of maplike coloring schemes reduces our choice of colors even more.

In GMap, we start with a coloring scheme from ColorBrewer (http://colorbrewer2.org) and generate a color for each country by blending the base colors. So, our color space is linear and discrete. Because of the blending, any two consecutive colors in the linear array of colors are similar.

When applying these colors to the map, we want to avoid coloring neighboring countries with such pairs of colors. (Although two nonneighboring countries with similar colors can lead the viewer to believe they're disjoint regions of the same country, this problem diminishes when the two countries are sufficiently far apart, because it's unlikely that distant regions belong to the same cluster.) With this in mind, we must solve a discrete-optimization problem in which we find the best color permutation such that we maximize the color difference between neighbors.

To do so, we use two procedures that operate on the *country graph*, which is created from the original graph after it has been embedded and clustered. Specifically, the country graph contains a node for each country, with each edge representing two countries sharing a border. The first procedure turns the map-coloring problem into

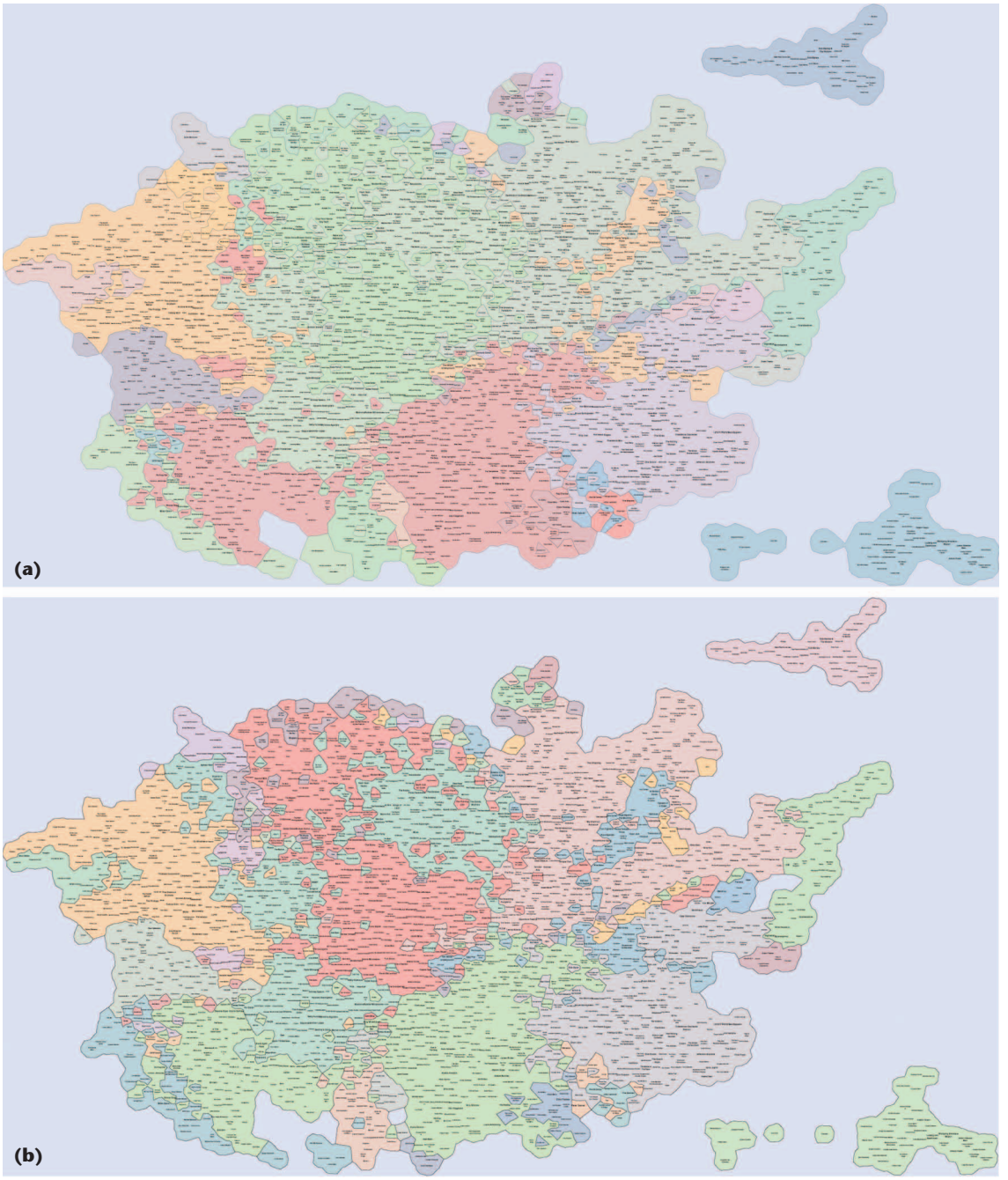

Figure 4. Coloring maps: (a) a random color assignment and (b) an optimized color assignment that maximizes color differences between neighbors. In the optimized color assignment, countries are much easier to differentiate.

that of a continuous-optimization problem on the country graph,

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

where E_c is the set of edges in the country graph, scalar value c_i is the color (index) assigned to country i, and weighting factor $w_{i,j}$ measures the importance of promoting color difference between country i and country j. The solution is the largest eigenvector of the weighted Laplacian of the country graph. We then use the ordering induced by the eigenvector's values as the color permutation.

We follow this with a greedy color-swapping procedure to see whether we can further improve the color difference. This heuristic algorithm significantly improves the color difference between neighbors, compared to a random color assignment.

Figure 4 illustrates the difference between a simple, random coloring and one obtained using our approach. The random coloring assigns a green-gray

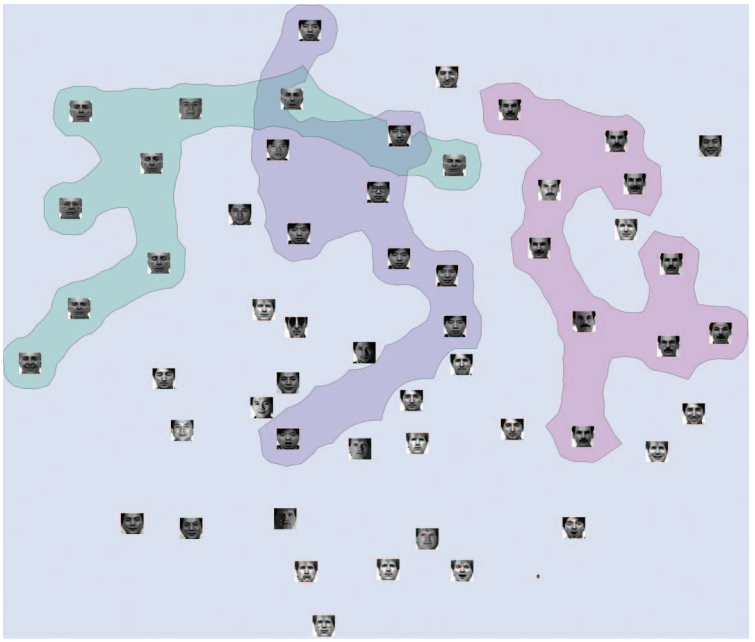

Figure 5. Using GMap to visualize multiple sets. Photos in the same set belong to the same person. The photos are from the Yale Face Database.

palette to the countries in the middle, making it difficult to distinguish them. In our optimized coloring, the separate regions in the central part are much more evident.

Using GMap to Visualize Set Relations

We designed GMap to visualize cluster relations as maps, in which each item belongs to one cluster and, hence, one country. However, we can easily adapt the same algorithm to visualize multiple relationships among a set of objects. Heorhiy Byelas and Alexandru Telea proposed visualizing set relations using deformed convex hulls,[3] whereas Christopher Collins and his coauthors created bubble sets based on isocontours.[4] GMap provides a different approach.

We work on one set at a time. We consider items not in the active set to be obstacles, and we treat them the same as random points inserted in the GMap algorithm. Applying GMap then gives us a map of one country. By repeating this process for multiple sets and using transparency to let us see the regions' overlapping parts, we achieve a visualization of multiple sets.

To avoid disconnected countries, we add edges to link items in the same set. The edge addition process is similar to that used by Collins and his coauthors. When possible, we route edges as splines to avoid hitting items not in the active set. We insert artificial points along the spline edges. When we apply GMap, it creates "bridges" along the edges to connect distant items.

Figure 5 shows 55 photos from the Yale Face Database,[5] representing portraits using different expressions and lightings. The photos are embedded in 2D using MDS, with the distance between two photos calculated using principal component analysis of a 55-row matrix, with each row a vector of a photo's pixel values. Even with careful preprocessing, the embedding doesn't always put the same persons in the same neighborhood. Without set visualization, identifying all photos of the same person is difficult. Figure 5 shows three sets of photos, each set belonging to a single person. As you can see, each country is connected, and each avoids photos not in the set. This example shows that GMap provides a good alternative for set visualization.

GMap Case Studies

The GMap layout accentuates clusters in fairly large graphs. Gleaning information from the maps typically involves an interactive, multiscale process, similar to that used for exploring geographic maps. Users can view the map at a small scale to sense the overall layout, the major regions, and their relationship to each other. Users can then zoom in to see local detail and to traverse the map along small features. At some point, users can zoom out again to put the local details into a global context.

On the basis of this usage style, GMap figures are most effective when displayed as a large image, often a meter or more in width, or via an interactive viewer. In the former case, the user can physically move to change the scale. In the latter case, the viewer provides the scale change and, at the same time, can provide semantic zoom, so that the more the user zooms in, the more detail appears. In addition, an interactive viewer can provide additional features, such as textual search or links connecting a feature on the map to external information. For example, clicking on a book in the BookLand layout (which we describe later in this section) might take the user to its Amazon entry.

Given page size limitations, we've scaled the figures we include here either to illustrate GMap layout or to give a high-level view of such maps.

In all cases, the countries and their geography in the resulting maps aren't part of the input data but emerge from the graph layout and clustering algorithms. This gives users a potential tool to discover structure based solely on local data.

MusicLand

To create a land of music, we collected data from a web crawl of Last.fm. This website has more than 30 million users and recommends music on the basis of user profiles. Over several years, the recommender system has collected information

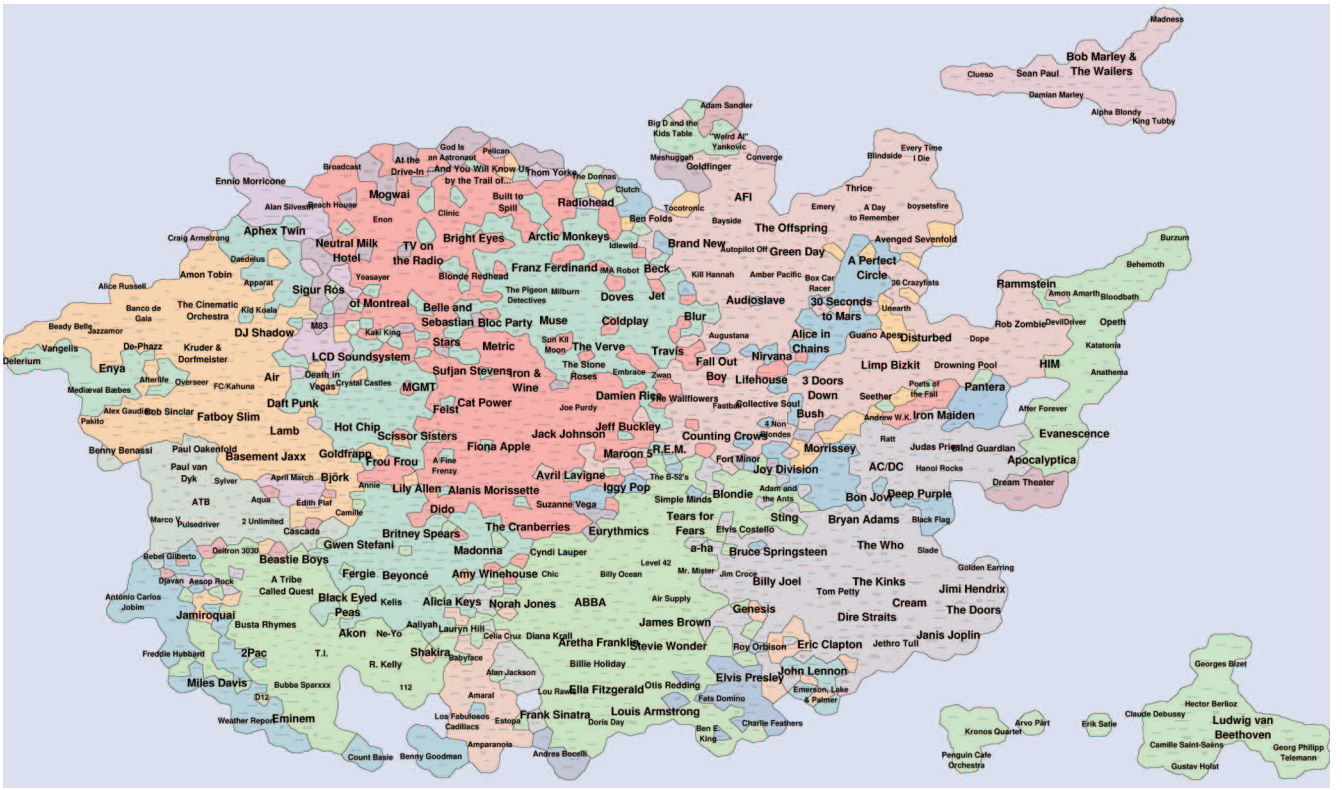

Figure 6. The landscape of music. Two musicians are placed close to each other if people who listen to one also tend to listen to the other. Each country represents a group of highly related musicians based on a cluster analysis. Notice the archipelago of classical composers in the southeast, and the island of reggae musicians in the northeast.

about how one band, musician, composer, or artist relates to another in terms of how many listeners of one also enjoy the other. For each composer, Last.fm lists the top 250 related composers. For example, Beethoven is considered to have "super similarity" to Mozart, Bach, and Brahms and "very high similarity" to Mendelssohn, Schumann, and Vivaldi. The website also provides the number of each musician's listeners.

In April 2009, we crawled the website by starting with Beethoven and the top 20 musicians most similar to him, provided that each had at least 100,000 listeners. (We added the cutoff of 100,000 because the number of listeners seems to follow a power law distribution. Without the cutoff, our crawl didn't finish after more than a week of crawling, during which time we observed well over 500,000 musicians, many with only a few hundred listeners.) We then found the 20 musicians most similar to each of those with at least 100,000 listeners and proceeded recursively. Our crawl yielded a graph with 2,782 musicians, with edge weights corresponding to the similarity between musicians. We pruned this graph by taking only edges that had "super similarity." Ignoring singletons, we ended up with 2,588 vertices. We then laid out the graph, clustered the vertices, and generated the MusicLand map (see Figure 6).

The mainland. The vast majority of musicians and bands are in a single continent. It's not as easy to spot major trends along the main axes, but many clusters are well defined, and neighboring clusters make sense musically. A cluster of classic rock on the east shore begins with Eric Clapton and Janis Joplin in the south, goes through The Who in the middle before reaching the heavy rock cluster of AC/DC and Iron Maiden in the north. Farther to the north is the grunge cluster, anchored by Nirvana and Alice in Chains.

On the west coast is a cluster of electronic music, featuring Fatboy Slim, Daft Punk, and DJ Shadow. To the north are avant-garde electronic bands such as Aphex Twin, whereas the extreme west houses electronic classics such as Vangelis. To the south is a compact, well-defined cluster of dance music represented by Paul Oakenfold and ATB.

In the map's center is a well-defined concentration of female singer-songwriters such as Alanis Morissette, Norah Jones, and Amy Winehouse. Pop music is southeast, with ABBA and the Eurythmics, whereas hip-hop and rap are southwest with Beyonce and Alicia Keys.

The islands. MusicLand has two notable island regions: Reggae Island in the northeast and the Classical Archipelago, the island chain off Rocky

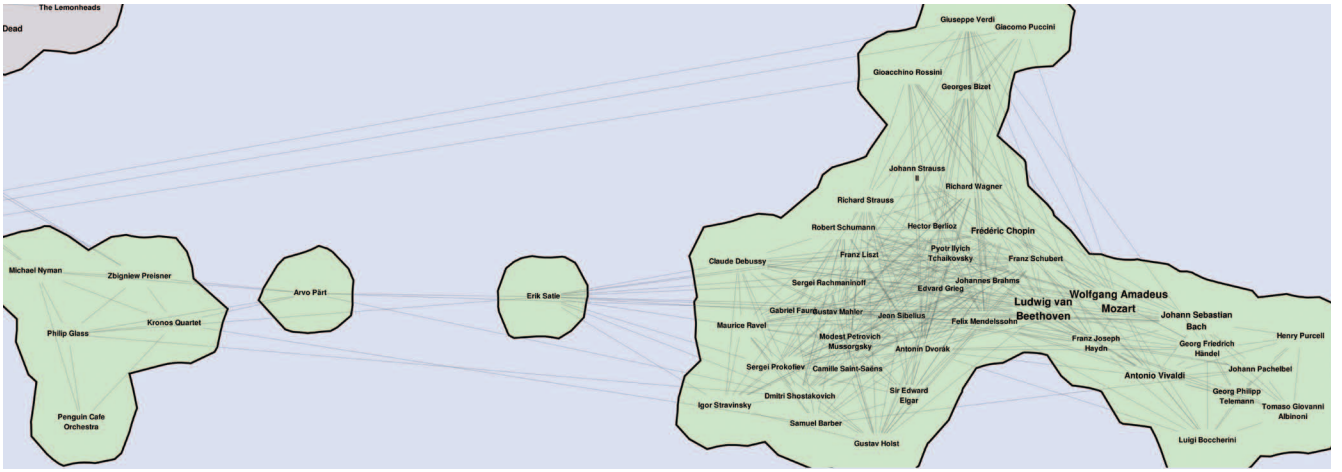

Figure 7. Part of the classical archipelago. The map shows a clear chronology from east to west, from the Renaissance to modern times.

Coast in the southeast. By zooming in (see Figure 7), we can easily spot general patterns in the archipelago's layout along the east-west and north-south axes. Along the first axis, the west contains modern composers such as Ravel, Satie, and Pärt, whereas the east contains 17th-century composers such as Bach, Handel, and Albinoni. Along the second axis, the north is highly populated by opera composers such as Verdi, Rossini, and Puccini, whereas the south has more orchestral and instrumental composers such as Holst, Elgar, and Stravinsky. Not surprisingly, Mozart and Beethoven are the most popular composers in the classical music cluster. The islands of Erik Satie and Arvo Pärt connect the big island in the east with the westernmost island of contemporary classical music, represented by minimalists Philip Glass and Michael Nyman.

BookLand

Many e-commerce websites provide recommendations to allow exploration of related items. Traditionally, this takes the form of a flat list. For example, Amazon typically lists five to six books under a "customers who bought this item also bought" heading, with a clickable arrow to show customers further related items.

Instead of a flat list, which provides a limited view of the neighborhood, researchers have attempted to convey the products' underlining connectivity through graph visualization. However, no approach displays a comprehensive view of the relationship and the clustering structures.

Using GMap, we produced the map in Figure 1. We obtained the underlying data with a breadth-first traversal of Amazon's "customers who bought this item also bought" links, starting from the root node, George Orwell's *1984*. We followed links up to a distance of 12 from the root node. We then trimmed the graph by keeping only vertices of distance nine or less from the root vertex. We further merged nodes that represented the same book, but with different publishers or different bindings, by matching books with the same title. This map's underlying graph contains 913 vertices and 3,410 edges. With an average degree of nearly eight, this map's peripheral vertices have only a handful of edges, whereas central vertices have more than 20 immediate neighbors. We now examine two BookLand countries in more detail.

Americana. Figure 8a shows this country. Somewhat surprisingly, 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 *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 J.D. Salinger's *Catcher in the Rye* transition to various well-known novels: John Steinbeck's *Grapes of Wrath* and *Of Mice and Men*, Ernest Hemingway's *For Whom the Bell Tolls*, F. Scott Fitzgerald's *Great Gatsby*, Joseph Heller's *Catch-22*, and Ken Kesey's *One Flew over a Cuckoo's Nest*.

Russiana. To the north of Americana lies one of BookLand's largest countries, dominated by Russian literature and history (see Figure 8b). The core contains classic novels by Fyodor Dostoyevsky (*The Brothers Karamazov*), Leo Tolstoy (*War and Peace*), and Aleksandr Solzhenitsyn (*The Gulag Archipelago* and *Cancer Ward*). In the west, unexpectedly, we find a cluster of Albert Camus works (*The Stranger*, *The Plague*, and *The Fall*), all well connected with the Russian classics.

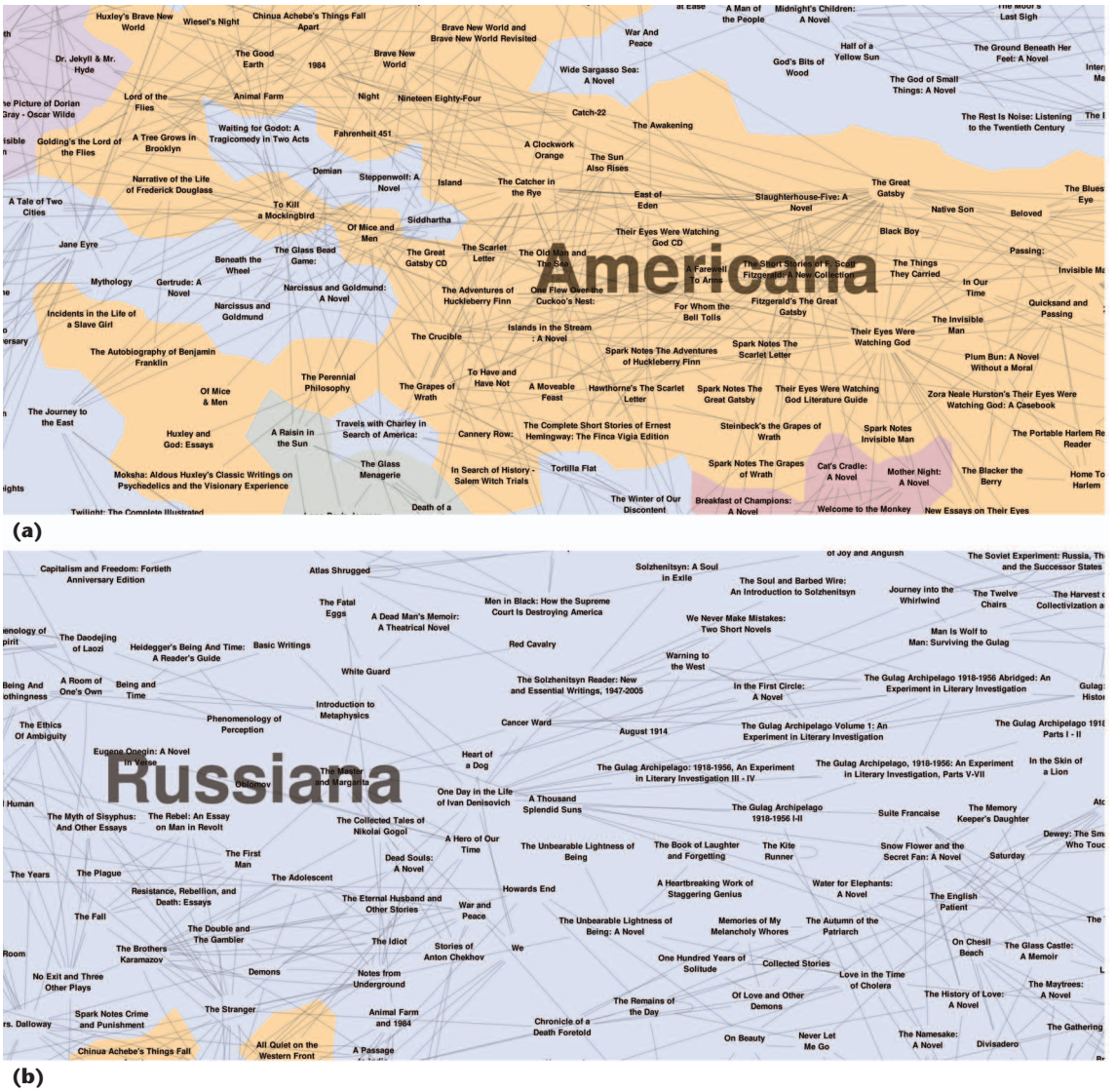

Figure 8. Two central clusters in BookLand: (a) Americana and (b) Russiana. These two portions of the map show the main clusters of American and Russian literature.

Other connections. The map also reveals the unsurprising proximity of self-help books with books recommended by Oprah Winfrey. In addition, recent vampire novels are adjacent to the cluster of Victorian novels. A closer investigation shows that vampire novels are connected to the rest of the graph only through Jane Austen.

TradeLand

Figure 9 is a map visualizing the trade relations between all countries. We acquired bilateral trade data between each of the 209 countries and their top trading partners from Mathematica's CountryData package. A label's font size is proportional to the logarithm of the country's total trade volume; the label's color reflects whether a country has a trade surplus (black) or deficit (red).

The label color makes it easy to spot oil-rich areas with large surpluses, which are distributed all over the world, as well as in our map: the Middle East (Saudi Arabia and Kuwait), Europe (Russia), South America (Venezuela), and Africa (Nigeria and Equatorial Guinea). On the other hand, the countries with huge deficits are mostly in Africa (Sierra Leone, Senegal, and Ethiopia), with the US as the clear outlier.

Many countries in close geographic proximity end up close in our map. For example, Central American countries such as Honduras, El Salvador, Nicaragua, Guatemala, and Costa Rica are close to each other in the northeast. Similarly, the three Baltic republics (Latvia, Lithuania, and Estonia) are near each other in the northwest. We can easily explain this by noting that geographically close countries tend to trade with each other. Two exceptions are easy to spot: on our map, North Korea isn't near South Korea, and Israel isn't particularly close to Jordan or Syria.

Figure 9. A map of trade relations between countries. Two nations are placed close to each other if there is a strong trade relation. Label size is proportional to the nation's trade volume. Black labels are used for nations with trade surplus, and red for deficits.

The G8 countries (Canada, France, Germany, Italy, Japan, Russia, the UK, and the US) are all close to each other in the map's center. Two of the largest, closest countries in our map are China and the US. Clearly, the proximity is due to the large trade volume rather than geographic closeness. All these countries are in the largest cluster, which is dominated by European countries in the west, Asian countries in the east, and Middle Eastern countries in the south.

Interestingly, African countries are distributed in several clusters close to China (a major trading partner with many African countries), the US (trading less with Africa these days), and former colonizers (for example, Togo, Cameroon, and Senegal are all close to France). On the other hand, Caribbean and South and Central American countries form several clusters in the map's north. In addition, these clusters are mostly contiguous, essentially forming a supercluster. The GMap figure clearly brings out the differentiation between Latin America and Africa.

Finally, the map's periphery contains small countries from around the world and countries with few trading partners.

TVLand

Finally, we consider a map derived from TV viewer data. The objects are TV shows, with an edge between two shows if many viewers watched both. The number of such viewers is used as the weight of an edge. We illustrate how GMap works in the context of recommendation systems with the aid of heat map overlays, tailored to individual viewers.

Figure 10 shows such a map for a typical but fictitious viewer. On the basis of the viewing habits of a specific viewer and of other viewers, a recommendation system can suggest possible new shows. To make the labels readable in this magazine format, our map displays a small subset of the TV shows and consists of six countries. The countries capture several types of shows by genre. The southwest has two countries corresponding to children's TV shows, with the smaller targeting younger kids with *Dora the Explorer* and *Wow Wow Wubbzy*. In the northwest is a fashion and entertainment cluster, with shows such as *Say Yes to the Dress* and *E! News*. The large country in the north contains popular sitcoms such as *Seinfeld*, *Cheers*, and *Frasier*. In the northeast is a cluster of crime shows (*NCIS* and *Law & Order*). The cluster in the southeast isn't as thematically focused; it contains popular shows of several types: *The Oprah Winfrey Show*, *So You Think You Can Dance*, and *Dateline NBC*. In each country, the color's intensity indicates the recommendation's strength. That is, the

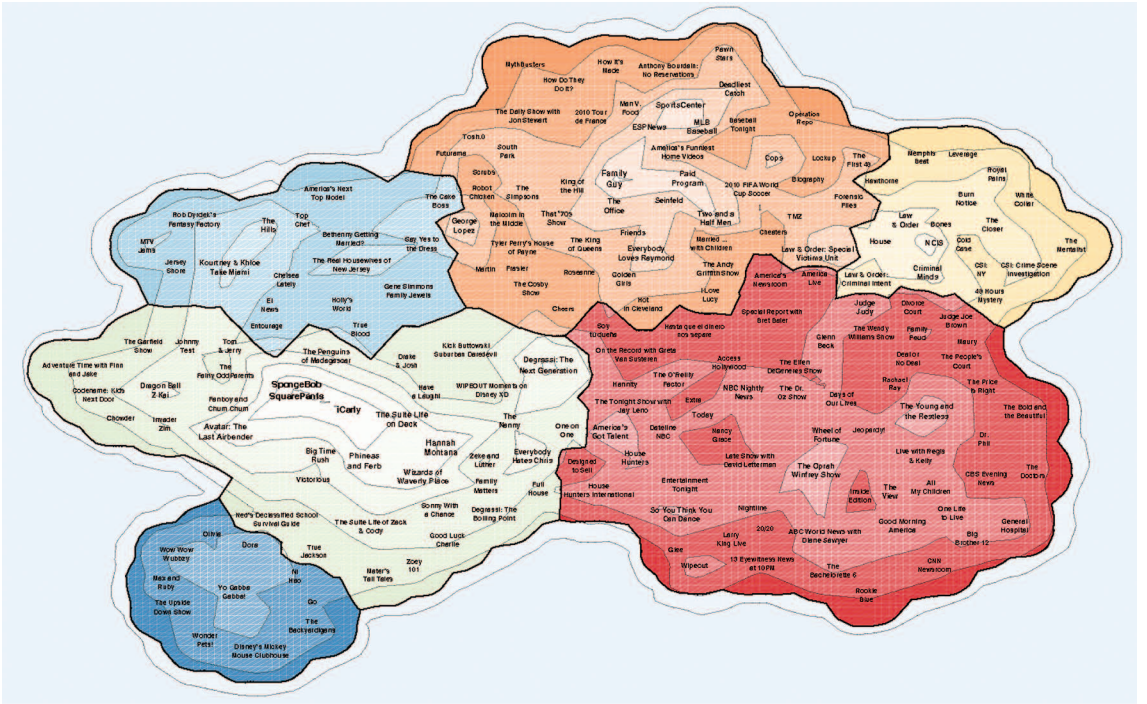

Figure 10. TVLand: a personalized recommendation heat map for a typical television viewer. Regions with the most highly recommended shows are lighter; regions of low scores are darker.

lighter the color is, the stronger the recommendation is, as in a typical geographic map that uses shading to represent hills and valleys.

Maps such as this provide a global view unavailable in the traditional list of recommendations and offer a more appealing presentation than a scatterplot or a node-link diagram. They provide a context for recommendations, letting users understand the reasoning behind the recommendation. In addition, when considering highly recommended shows, for example, users can check out related shows or explore a path of shows to a new area.

The GMap framework, by capitalizing on an ancient, familiar visual metaphor, introduces a significant new style of viewing abstract relational and cluster data that users will find aesthetically appealing and helpful for understanding. From informal observation, we've found that people, when faced with a traditional graph drawing and a map of the same data, spend significantly longer time studying the map, and they find nontrivial structural information without any prompting. For example, several BookLand viewers observed that the gateway to Fringistan (north in Figure 1) is Ayn Rand's Atlas Shrugged. We plan to perform formal user studies of the interaction with graphs and maps, in the context of visualizing recommendations.[6]

We also plan to address specific idiosyncrasies. For example, GMap can produce disconnected countries. As we mentioned earlier, we've proposed ways to use the cluster information to adjust the layout so that the regions of countries are more contiguous, at the expense of some loss of graph information.[2] However, we're planning further investigation into balancing country connectedness and preserving graph information.

Finally, our algorithm is efficient and can handle large graphs. As a reference point, we generated all maps in this article in a few seconds. Mapping a larger graph with 440,000 vertices took 4 minutes on a typical processor. The resulting maps look best on large, wall-sized posters and display walls. However, our interactive interface lets users easily search, zoom, and pan. To try it, visit http://www2.research.att.com/~yifanhu/MAPS/imap.html.

Acknowledgments

We thank Stephen North for helpful discussions and Carlos Scheidegger for help with the image process example.

References

1. M.E.J. Newman, "Modularity and Community Structure in Networks," *Proc. National Academy of Sciences*, vol. 103, no. 23, 2006, pp. 8577–8582.
2. E.R. Gansner, Y.F. Hu, and S.G. Kobourov, "GMap: Visualizing Graphs and Clusters as Maps," *Proc.*

IEEE Pacific Visualization Symp., IEEE CS Press, 2010, pp. 201–208.
3. H. Byelas and A. Telea, "Visualization of Areas of Interest in Software Architecture Diagrams," *Proc. 2006 ACM Symp. Software Visualization* (SoftVis 06), ACM Press, 2006, pp. 105–114.
4. C. Collins, G. Penn, and S. Carpendale, "Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 6, 2009, pp. 1009–1016.
5. A. Georghiades, P. Belhumeur, and D. Kriegman, "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, 2001, pp. 643–660.
6. E.R. Gansner et al., "Putting Recommendations on the Map: Visualizing Clusters and Relations," *Proc. 3rd ACM Conf. Recommender Systems*, ACM Press, 2009, pp. 345–348.

Emden R. Gansner *is a lead member of the technical staff in AT&T Labs Research's Information Visualization Research Department. His research interests include graphs (drawing, theory, and algorithms), information visualization, graphical user interfaces, programming languages, and combinatorics. Gansner has a PhD in mathematics from the Massachusetts Institute of Technology. Contact him at erg@research.att.com.*

Yifan Hu *is a principal member of the technical staff in AT&T Labs Research's Information Visualization Research Department. His research interests include numerical and combinatorical algorithms, information visualization, and data mining. Hu has a PhD in mathematics from Loughborough University. Contact him at yifanhu@research.att.com.*

Stephen G. Kobourov *is an associate professor in the University of Arizona's Computer Science Department. His research interests include graph drawing and geometric algorithms, emphasizing problems relating to graph visualization. Kobourov has a PhD in computer science from Johns Hopkins University. Contact him at kobourov@cs.arizona.edu.*