Viewing Abstract Data as Maps

Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov

Abstract From telecommunications and abstractions of the Internet to interconnections of medical papers to on-line social networks, technology has spawned an explosion of data in the form of large attributed graphs and networks. Visualization often serves as an essential first step in understanding such data, when little is known. Unfortunately, visualizing large graphs presents its own set of problems, both technically in terms of clutter and cognitively in terms of unfamiliarity with the graph idiom. In this chapter, we consider viewing such data in the form of geographic maps. This provides a view of the data that naturally allows for reduction of clutter and for presentation in a familiar idiom. We describe some techniques for creating such maps, and consider some of the related technical problems. We also present and discuss various applications of this method to real data.

1 Introduction

In an increasingly technological world, we find ourselves dealing with large, multivariate data sets in order to make informed decisions. For many, tables of numbers are a cue for the eyes to glaze over. Even experts can have difficulties determining patterns in "raw" data. For this reason, the statistics community has, over several centuries, developed a variety of visualizations for statistical data [5] which can expose correlations and structures that otherwise might be missed. Recently, a complementary effort in the information visualization community has

E.R. Gansner (✉) • Y. Hu
AT&T Labs – Research, Florham Park, NJ, USA
e-mail: erg@research.att.com; yifanhu@research.att.com

S.G. Kobourov
University of Arizona, Tucson, AZ, USA
e-mail: kobourov@cs.arizona.edu

W. Huang (ed.), *Handbook of Human Centric Visualization*,
DOI 10.1007/978-1-4614-7485-2_3,

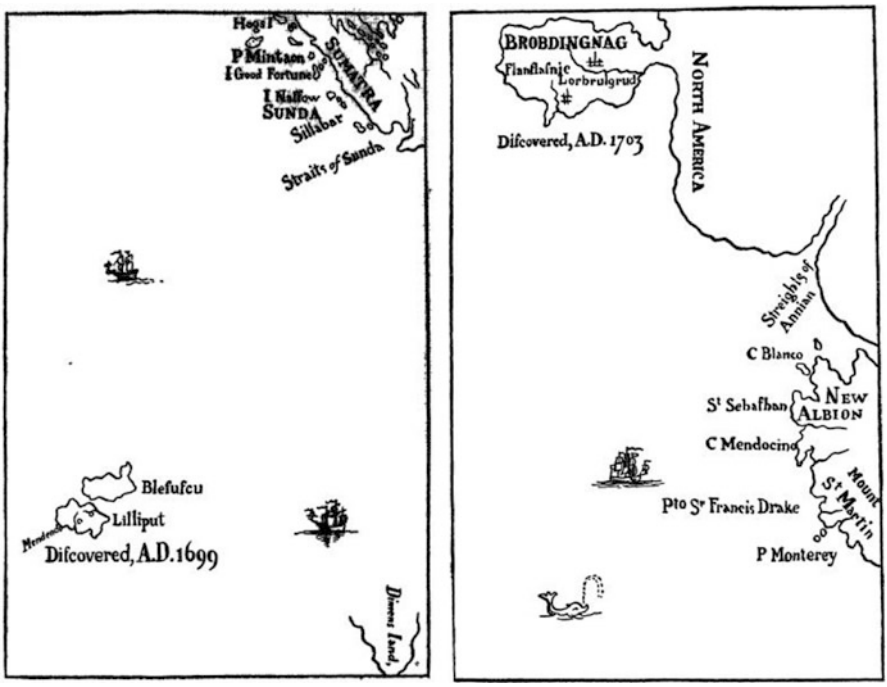

Fig. 1 Lilliput and Brobdingnag (Thanks to Project Gutenberg)

expanded the number of visual metaphors [30, 43]. The key to all of this work is to take the many dimensions inherent in the data and reduce it down to the two or three that are accessible to the human eye, and doing it in a fashion that reveals or at least maintains the data's most salient features.

One approach to making this information more accessible to the human is to rely on more familiar, less technical visual metaphors, in essence, to map abstract data into a more concrete or physical space, ones tangible to and experienced by the viewer. Not surprisingly, various researchers in the geographical information science community (GIS) were among the pioneers in promoting this style of visualization [27, 40, 41], which they termed *spatialization*.

Constraining the metaphor even more, one can consider how to present abstract data in the context of a geographic map. At the simplest level, this might involve merging geographic and abstract data [12]. Cartograms [24, 25] provide another level of abstraction, in which quantitative information about a geographic region is represented by area or distance, usually requiring a distortion of the geography. These appear particularly attractive to the popular press.

More interesting is to derive a pseudo-geographic map from the abstract data. People seem generally intrigued by maps, and seeing fictional or abstract information portrayed on a map makes it more real. Authors, from Swift (Fig. 1) to Hardy to Tolkien, have long provided maps to guide the reader through their world.

Fig. 2 Aspects of eighteenth century financial bubbles as a map (http://bigthink.com/strange-maps/554-the-fools-head-map-a-fossil-of-the-financial-bubbles-of-1720)

More relevantly, there is also a very long tradition of displaying abstract data in the form of a map. One early example is the Fool's Head Map (Fig. 2) diagramming the financial bubbles of the early 1700s. This blends real geography (the Thames and Seine Rivers) with allegorical aspects such as "Crazy Town" and islands of "Poverty" and "Despair." More recent examples include Cardelli's map of programming language concepts used as the cover for Ullman's ML book [44] and Randall Munroe's take on the evolution of online communities (Fig. 3). Clearly, creating these maps relied on the talent, knowledge and wit of someone with artistic talents.

For use in information visualization, it is necessary that we be able to automate the production of such maps, with an eye for aesthetics but also with the acceptance that some genius of the hand-made map will be missing. Various techniques have been devised for producing such maps. For example, from the GIS community, there is the notable work of Skupin, Fabrikant and others [10, 35, 37–39]. Adding to intuition, there is some evidence that map-based displays of abstract data provide an aid to comprehension [9].

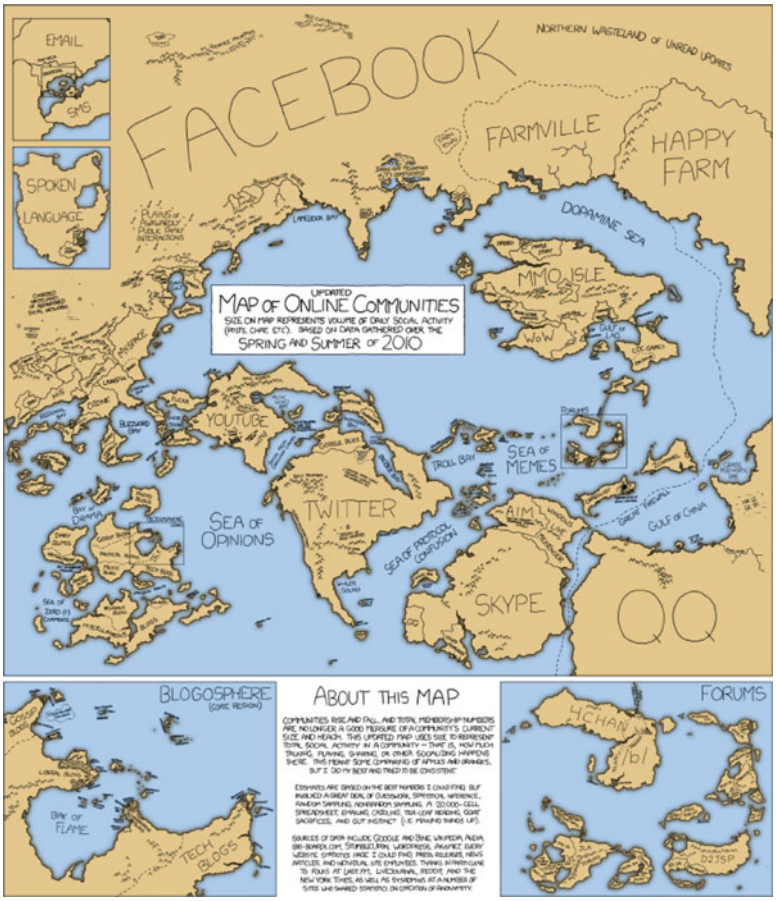

Fig. 3 Munroe's map of online communities (http://xkcd.com/802/)

Until the recent past, much data visualization dealt with simple attributed data. That is, the data was viewed as a collection of records, each record being one of a small number of types, which determined the fields it had. A further complication has been added that now many data sets have a graph or explicit relational structure as well. (Theoretically, a graph can be represented using a simple attributed data model, and in practice, the data often contained a graph implicitly. Making the graph explicit also implies the desire to visualize the graph explicitly.) Canonical examples of graph-based data are the various graphs induced by the Internet, the friendship graph induced from Facebook, or the "following" relation from Twitter. Driven by the increasing presence and importance of graphs, in software engineering, biology, telecommunications, social networks, etc., there has been a great deal of work in the theory and practice of drawing graphs [1], including graphs with thousands of vertices.

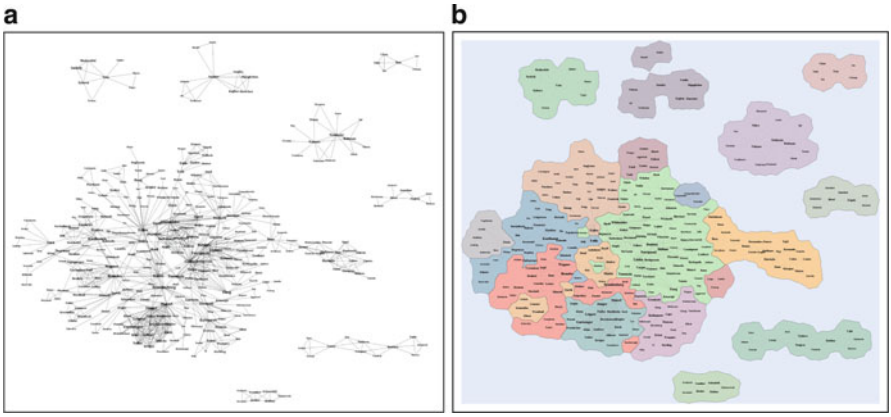

Fig. 4 A node-link drawing compared to a map representation of the same graph

As with other forms of data, representing graphs as maps can make the data more accessible to the reader, replacing a typical node-link diagram or point-cloud visualization with a more compelling drawing. A further impetus for considering graphs as maps arises naturally from standard graph drawing. Figure 4a displays a typical graph layout using a node-link diagram. The graph represents the author collaborations between 1994 and 2004 at the Symposium on Graph Drawing. The drawing exhibits the connected components, and closely related nodes are indicated by proximity, but cluster structure is only hinted at. It is not difficult to already see a map there, so why not go the next few steps and arrive at the rendering in Fig. 4b? In this version, the cluster structure is obvious. Coloring the nodes in the node-link drawing would still only imply the clusters. The map representation makes the clusters explicit as well as indicating strong cluster relations where two clusters share a border.

In the remainder of this chapter, we will explore a technique for displaying graphs as geometric maps. The creation of the basic geometry is described in the next section. Section 3 addresses some of the auxiliary problems that arise in making a good map, such as how to best color the regions or provide additional features to help the reader. Unlike in the real world, the geography of much abstract data is in constant motion, changing with each packet or phone call. We look at the issue of dynamic maps in Sect. 4. Section 5 applies the methods described to real-world data and shows how such maps can provide insights into the data. We conclude with a few thoughts in Sect. 6.

2 Making the Map

The technique we describe here, which we refer to as GMap, allows us to generate map-like representations from an abstract graph. In particular, given a graph with weighted edges, such as how similar or dissimilar two books are based on customer

Fig. 5 A map of books related to "1984" from Amazon.com

purchases, we produce a drawing with a map-like look, with countries that enclose similar objects, outer boundaries that follow the outline of the vertex set, and inner boundaries that have the twists and turns found in real maps. A typical example is given in Fig. 5, showing just under 1,000 books, with edges determined by Amazon.com's record of related purchases. Our maps also can have lakes, islands, and peninsulas, similar to those found in real geographic maps. The technique is applicable outside of the domain of graphs; it can typically be used on most high-dimensional data sets.

This technique is a framework in the true sense of the word, rather than a specific algorithm. It consists of three main steps. The first two steps are fairly generic and can be achieved by a variety of existing algorithms. The last step is tuned to creating a map and involves a special-purpose algorithm. When finished, we have a drawing that has the basic appearance of a geographic map. With the addition of colors for the countries, perhaps coastal shading, mountains or some other effects, we have an acceptable imitation of a map. As noted above, these final features will be covered in Sect. 3.

Much of our presentation is narrative and informal. We refer the interested reader to the articles [14, 15, 19] for technical details and more references and examples. A prototype implementation is available as part of the open source Graphviz software package [16, 18].

2.1 Laying Out the Boundaries

For the first step, we take as input a graph or high-dimensional data set, and embed it into the 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, or non-linear dimensionality reductions such as Locally Linear Embedding and Isomap.

The second step takes this collection of points in the plane and aggregates them into clusters. Here, it is important to match the clustering algorithm to the embedding algorithm. For example, a geometric clustering algorithm such as k-means [31] may be suitable for an embedding derived from MDS, as 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 [32] could be a better fit. The two algorithms are strongly related, 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, we use the two-dimensional embedding together with the clustering to create the actual map by delineating country boundaries, carving continental outlines, and separating islands from continents. This can be accomplished with the help of plane partitioning techniques such as Voronoi diagrams, along with the addition of new algorithmic techniques to ensure realistic looking outer and inner boundaries. We want to create a map, with inner boundaries separating points not in the same cluster and outer boundaries preferably 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. This is illustrated in Fig. 6a. 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 some preset threshold. Note that this step can be implemented efficiently using a suitable space decomposing data structure, such as a quadtree. This leads to boundaries that follow the shape of the point set. In addition, the randomness of the points on the outskirts gives rise to some randomness of the outer boundaries, thus making them more map-like, as seen in Fig. 6b. Furthermore, depending on the value of the threshold, this step can also result in the creation of lakes and fjords in areas where vertices are far apart from each other. Nevertheless, some inner boundaries remain artificially straight.

At this point, we still note the undesirable feature that the "countries" all have roughly the same area (Fig. 6b), whereas we might prefer some areas to be larger than others (e.g., due to the importance of the entities they represent). As an illustration, in Fig. 6, we assume that "node 1" is more important than the other two nodes, and use a larger label for that area.[1] To make areas follow the shape of the labels, we first generate artificial points along the bounding boxes of the labels as shown in Fig. 6c. To make the inner boundaries less uniform and more map-like,

[1] A weighted Voronoi diagrams can be used to make the area of each Voronoi cell proportional to its weight. We do not use this approach, however, because we want the Voronoi cell to also contain a specific shape, e.g., the bounding box of a label.

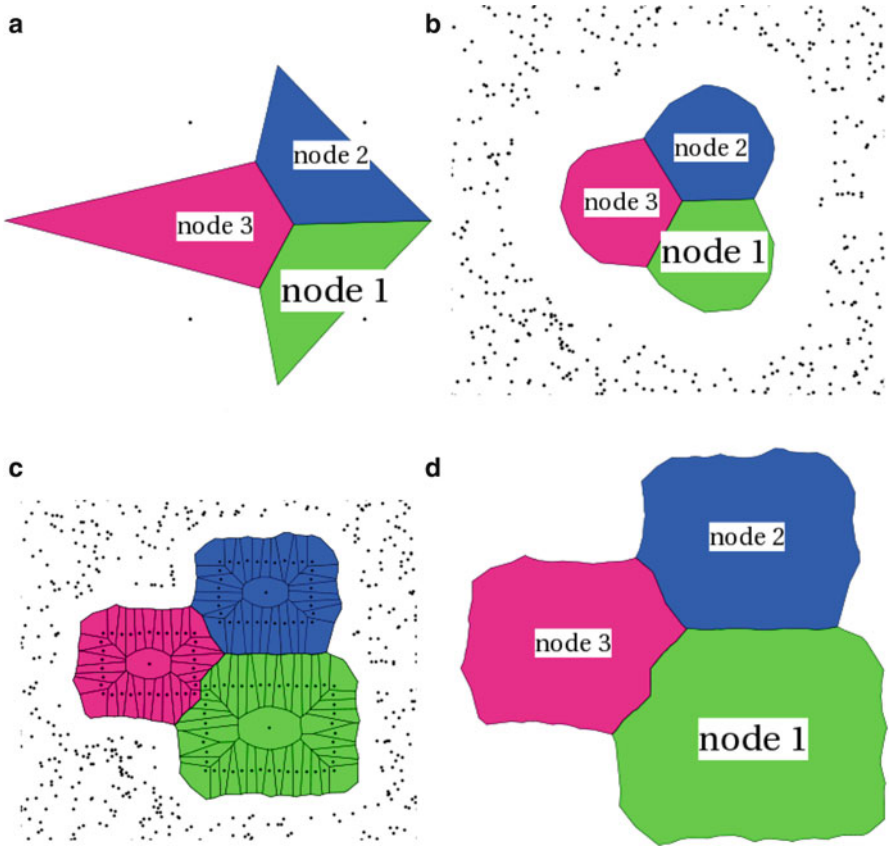

Fig. 6 (**a**) Voronoi diagram of vertices and corners of bounding box; (**b**) better construction of outer boundaries through placement of random points; (**c**) Voronoi diagram of vertices and points inserted around the bounding boxes of the labels; (**d**) the final map

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. 6d. 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, boundaries of the western states in the United States often have long straight sections. We believe that irregular boundaries are more typical of historical and geographic boundaries, and lead to more map-like results. But this is a matter of personal taste and our technique 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.

With the regions determined, we have a representation of the data in which closely related objects, as determined by the graph topology and possibly edge weights, are drawn closely together. This geometric information is then used to discover clusters among the objects. To emphasize the clusters, each is represented as a collection of geometric regions.

When projecting high dimensional data into low dimensional space, distance distortion is inevitable, and the resulting figure will often have some anomalies and distortions. Thus, some strongly related objects may be separated by seemingly unrelated objects. For example, in Fig. 13, we see several Richard Pryor shows connected to the purple country by color but lying outside of the country's main region. These shows are also closely connected to shows in other countries further down the map and are therefore pulled away. Such fragmentation is inherent in the embedding and clustering algorithms used in the first two steps. However we have proposed ways [14] to use the clustering information to adjust the layout, so that the regions of countries are more contiguous, at the expense of some loss of relational information captured in the original embedding.

3 Map Features

Once the map geometry is in place, we can add additional graphical attributes to the drawing in order to enhance its clarity, to serve as keys to the abstract data, or to simply make it more aesthetically appealing. To this end, one natural approach is to employ additional cartographic or topographic conventions, such as overlaying mountains, rivers or roads, applying coastal shading, or generating a relief map such as the one shown in Fig. 15.

One feature common to almost all maps is a coloring of the regions to emphasize commonality or separation. Thus, in past centuries, one could rely on all the states of the British Empire being colored pink. Achieving a good coloring for our artificial maps brings its own set of problems, which we now address.

3.1 Map Coloring

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. 13 as we previously noted, we have several Richard Pryor shows that belong to the *Saturday Night Live* cluster, but are separated due to his connections with films in other regions. 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. Instead, we will have to use one unique color for each cluster to avoid ambiguity.

Estimation of the number of colors an "average human" can discriminate, when color pairs are presented side by side, ranges from tens of thousands to a million. However, the number of colors a person can differentiate, when similar colors are not immediately next to each other, is far smaller. A further limiting factor is that 5 % of males are color blind, which rules out certain coloring schemes. Finally, some coloring schemes are used more often than others in maps, reducing the number of colors even more.

In coloring our maps, we start with a coloring scheme from ColorBrewer [3], and generate as many colors as the number of countries by blending the base colors. As a result our color space is piecewise linear and discrete. It remain to be decided which color should be assigned to which country. Because the number of countries can be as many as 30 in many examples, and because we blend a few distinctive colors to form a discrete 1D array of colors, 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 adjacent pairs of colors. Although two non-neighboring countries with similar colors can lead the viewer to believe that they are disjoint regions of the same country, this problem diminishes when the two countries are sufficiently far apart, as it is unlikely that distant regions that are far away belong to the same cluster. 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 $\{i, j\} \in E_c$, indicating how important it is to color node i and j with distinctive colors. 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:

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

and

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

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. [6] 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. In our map coloring problem, however, we are limited to "map-like" colors for aesthetic reasons, and our color space is discrete. Therefore, for simplicity, 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

$$\max_{\pi} \sum_{\{i,j\} \in E_c} w_{i,j} (\pi_i - \pi_j)^2, \quad \text{(2-norm)} \tag{1}$$

and

$$\max_{\pi} \min_{\{i,j\} \in E_c} w_{i,j} |\pi_i - \pi_j|, \quad \text{(MaxMin)} \tag{2}$$

where π_i is the i-th element of the permutation π of $\{1, 2, \ldots, |V_c|\}$.

It turns out that the MaxMin problem (2) is known as the antibandwidth problem, and arises in a number of practical applications. For example, it belongs to the family of obnoxious facility location problems. Here the "enemy" graph is one for which nodes are people and there is an edge between two people if and only if they are enemies. The problem is to build each person a house along a road so that the minimal distance between enemies is maximized [4]. Another example is the radio frequency assignment problem in which the nodes correspond to transmitters and the edges are between interfering transmitters; the objective is to assign the frequencies so that those for the interfering transmitters are as different as possible.

This antibandwidth maximization problem is NP-Complete [29]. In the literature, theoretical results have been presented for some special graphs, including paths, cycles, rectangular grids, special trees and complete bipartite graphs (see, for example, [34] and the references therein).

For more general graphs, heuristics algorithms are being developed. Hu et al. [20] have developed an algorithm `GSpectral` (Greedy Spectral) that is based on computing the eigenvector corresponding to the largest eigenvalue of the Laplacian associated with the graph and then using a greedy refinement algorithm. Duarte, Martí, Resende and Silva [7] have proposed a linear integer programming formulation and several heuristics based on `GRASP` (Greedy Randomized Adaptive Search

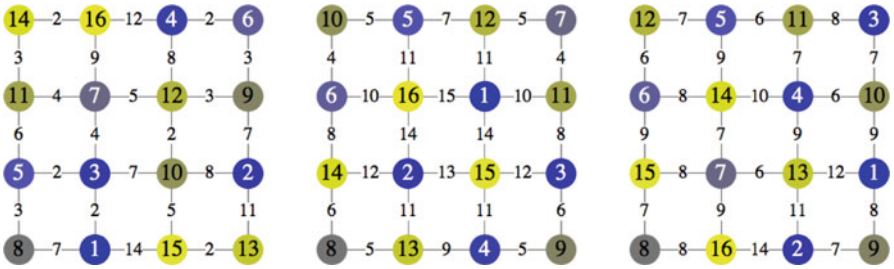

Fig. 7 Coloring schemes RANDOM, SPECTRAL, and SPECTRAL+GREEDY. Each node is colored by the color index shown as the node label. Edge labels are the absolute difference of the endpoint labels

Procedure) with path relinking. They present some high-quality computational results for general graphs, although the run-times for their relatively modest-sized test problems (graphs with fewer than 9,000 nodes) are quite high (typically several minutes for their fastest approach applied to their largest problems). Scott and Hu [36] presented a faster heuristic with the rough idea of finding a pseudo diameter of the graph first, then ordering the corresponding level sets in an alternating fashion, followed by a greedy refinement. The algorithm was found to give comparable ordering to GRASP, but works for much larger graphs.

Here we describe the `GSpectral` algorithm. The algorithm is motivated by the fact that 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 [23, 28, 42] were proposed. One such heuristic is to order vertices using the Fiedler vector. This is found to be very effective when combined with a refinement strategy. Motivated by this approach, we approximate (1) by

$$\max_{c} \sum_{\{i,j\}\in E_c} w_{i,j}(c_i - c_j)^2, \text{ subject to } \sum_{k\in V_c} c_k^2 = 1 \tag{3}$$

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 (3) is solved, we use the ordering of the eigenvector as an approximate solution for (1). We call this algorithm SPECTRAL.

Figure 7 illustrates three coloring schemes on a 4×4 unweighted grid graph given 16 colors in some discrete 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

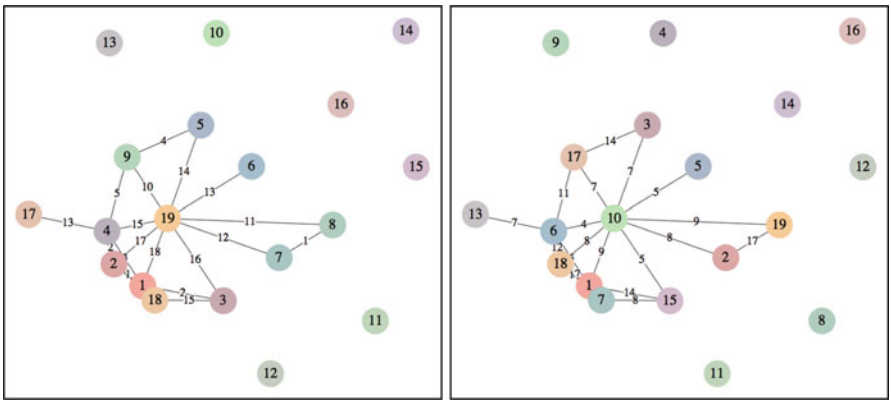

Fig. 8 Applying coloring schemes for the country graph corresponding to the map in Fig. 4b. *Left*: SPECTRAL. There are two edges of color difference 1. *Right*: SPECTRAL+GREEDY, the smallest color difference along any edges is now 4. Node labels are the color index given to the node, and edge label are the absolute difference of the node color index. Nodes are positioned at the center of the polygons in Fig. 4b

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. Figure 8 (left) gives the country graph corresponding to the map in Fig. 4b, with color assignment given by SPECTRAL. There are two edges of color difference 1. Applying the GREEDY algorithm guided by the MaxMin cost function to the result of SPECTRAL gives Fig. 8 (right). Now the minimum color difference along any edge is 4, a large improvement. This is indeed the coloring scheme used to create Fig. 4b.

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 chapter were colored using SPECTRAL+GREEDY, the `GSpectral` algorithm.

We note in passing that GREEDY is flexible enough to be used with any other cost functions. 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 Dynamic Maps

Unlike maps of the real world, where changes happen on a historical, if not geological, time scale, the data we consider here is frequently changing daily, hourly or even every second. To understand the evolution of this streaming data, it is important that stability can be provided by visual cues. Dynamic map visualization deals with the problem of effectively presenting relationships as they change over time. Traditionally, dynamic relational data is visualized by animations of node-and-link graphs, in which nodes and edges fade in and out as needed. One of the main problems in dynamic visualization is that of obtaining individually readable layouts for each moment in time, while at the same time preserving the viewer's mental map. A related problem is that of visualizing multiple relationships on the same dataset. Just as with dynamic data, the main problem is guaranteeing readability while preserving the viewer's mental map. Representations based on the geographic map metaphor could provide intuitive and appealing visualizations for dynamic data and for multiple relationships on the same dataset.

We give some motivation for dynamic map layout first, then describe a heuristic to promote dynamic *cluster stability*, an optimal color assignment algorithm to maximize *color stability* between maps, and heuristics to improve *layout stability*. Additional details can be found in [21].

4.1 Dynamic Maps: A Motivation

Consider the problem of computing a "good" distance measure between a set of known DNA samples that is based on multiple similarity measures (e.g., NRY and mtDNA), with the goal of creating a "canonical map" of the DNA space spanned by these samples. In this map, DNA samples are nodes, two nodes are close to each other if they have a high similarity, and groups of similar nodes are clustered into "countries."

Next an unknown DNA sample can be compared to the known ones and then placed on the map, in a way that minimizes its distance to the most similar known samples. In order to do this, we must compute such a "good" distance measure from multiple similarity metrics, for example, by assigning weights to each metric and taking a weighted sum, or some non-linear combination thereof. Once appropriate weights have been assigned we can create the canonical map where we will place unknown DNA samples.

Thus the main problem here is figuring out how to appropriately combine a set of different similarity metrics. Given two different similarity metrics on the same set of DNA samples, a simple way to visualize them is to create two static maps. This, however, is not very helpful to the scientists who would like to understand the correspondences and differences between these two metrics, as node positions on the static maps are likely to be unrelated. In addition, color assignment for the countries are random, making it even harder to understand the relationships.

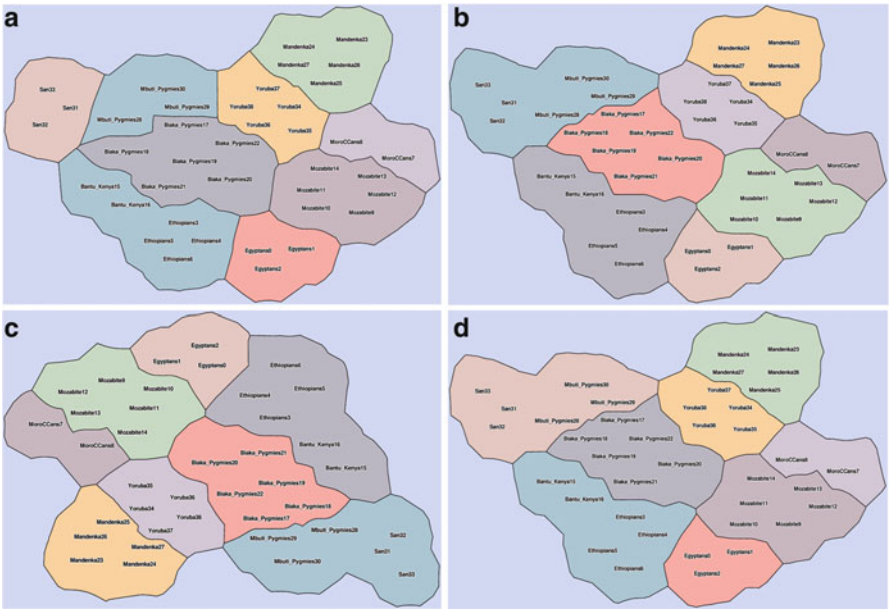

Fig. 9 Visualizing multiple maps requires both layout stability and color stability. (**a**): original map. (**b**): a new map based on different similarity data, and with node layout computed independently. It is difficult to see the corresponding nodes in the two maps. (**c**): the new map computed to optimize node layout stability with regard to (**a**), which makes it possible to compare nodes, while clusters are still hard to compare. (**d**): the new map with optimal node layout and color assignments, which makes it easy to compare with (**a**); e.g., it is clear that two clusters in the top left of (**a**) are now merged

The maps in Fig. 9 show the nature of the problem. In Fig. 9a, 39 subjects are embedded in 2D space based on mtDNA similarity, using multi-dimensional scaling, and clustered and mapped. In Fig. 9b we have re-embedded the subjects using a different similarity metric (NRY DNA similarity), independent of Fig. 9a. Compared with Fig. 9a, the layout changed significantly. Furthermore, although the same color palette is used, colors are assigned independently, making it even harder to figure out the relationship between Fig. 9b,a. In Fig. 9c, the embedding of NRY DNA similarity is done to minimize the difference to that of the embedding based on the mtDNA similarity measure, making it possible to see that node positions are largely unchanged. However due to the color assignment, it is still difficult to compare it with the map in Fig. 9a. Finally, in Fig. 9d, colors are properly matched such that clusters with mostly the same nodes are colored using the same colors. This makes it easy to compare Figs. 9a,d. For example, we can clearly see that two countries in the top left are now merged in a single country.

From this example, we can see that to give the viewer a stable mental map when viewing dynamic maps, we first have to ensure that the layout is done such that the same node should appear at the same or a nearby position if possible (layout

Fig. 10 Trajectories of randomly selected nodes with three different layout stability methods. (*Left*) independent layout with average distance traveled 21.41; (*Middle*) layout initialized with positions from the previous frame with average distance traveled 13.19; (*Right*) initialized positions and Procrustes transformation, with average distance traveled 8.43

stability); secondly, the clustering of the data should be stable, without losing the quality of the clustering (clustering stability). Finally, coloring of the maps should be done so that clusters with more or less the same nodes should be associated with the same color if possible (color stability).

4.2 Stable Dynamic Graph Layout

Abstractly, the problem of dynamic map layout is that of computing node positions, which is related to the well-known readability versus mental map preservation problem for dynamic graph drawing. Traditionally, given a sequence of graphs, one can compute node positions for the current graph in the sequence by starting with the node positions from the previous graph in the sequence and followed by local node position refinement. One shortcoming of such an approach is that even with the node-position initialization, two consecutive graphs in the sequence with very similar topologies can have very different drawings, causing node-jumping between frames, and failing to preserve the mental map. One technique that can help to moderate the change in position is to apply a Procrustes transformation of the coordinates of the nodes, so that the new layout matches the old layout as much as possible by using scaling, translation and rotation.

Another dynamic layout approach is to "anchor" some, or all, of the nodes, or to limit node movement by adding artificial edges linking graphs in different time frames [8]. However, such approaches can introduce biases that were not in the data itself, which is undesirable when analyzing highly sensitive real-world data, such as DNA similarity.

To evaluate the different layout stability approaches, we compare the trajectories of a set of randomly selected nodes from the dataset in Fig. 9. Figure 10 (left) shows such node trajectories, where the position of a node in the new graph is obtained by an independent MDS computation of the two layouts. Figure 10 (middle) shows the node trajectories, when using an MDS layout of the current frame, where the position of each node is initialized with the position obtained

from the previous frame. Finally, Fig. 10 (right) shows node trajectories, where the position of each node is initialized with the position obtained from the previous iteration and combined with a Procrustes transformation to fit the previous frame.

In all cases we experimented with, the last strategy was the best one, its trajectories the least jittery. We quantify these strategies by computing the average node-travel distance per frame (over all nodes in the graph, not just the random sample shown in the figure). In this example the distances traveled are 21.41, 13.19 and 8.43 pixels, respectively. This confirms that there are non-trivial improvements when we use the initial nodes position together with a Procrustes transformation.

4.3 Stable Clustering

We now consider the clustering problem on dynamic graphs, where the changes are adding/removing nodes, adding/removing edges, and modifications in node weights and edge weights. For the purpose of mental map preservation, we seek to preserve the clustering structure between the iterations as much as possible, provided that doing so does not result in suboptimal clustering.

One commonly used clustering for graph data is based on minimizing the modularity of a partition of the nodes. Here the modularity of a partition is defined as

$$Q = \frac{1}{2m} \sum_{i,j \in V} [w_{i,j} - \frac{k_i k_j}{2m}] \delta(C(i), C(j)) \tag{4}$$

where $w_{i,j}$ is the weight of the edge between i and j. The scalar $k_i = \sum_j A_{i,j}$ is the sum of the weights of the edges attached to node i, and $C(i)$ is the cluster node i is assigned to. The $\delta-$ function $\delta(C(i), C(j))$ is 1 if $C(i) = C(j)$ and 0 otherwise, and $m = \frac{1}{2} \sum_{i,j} A_{i,j}$ is the sum of all edge weights.

We describe a simple heuristic to combine the two objectives of modularity and cluster stability for dynamic clustering. This heuristic is a dynamic variation of the agglomerative clustering algorithm of Blondel et al. [2]. Heuristics are a reasonable approach, as the dynamic modularity clustering problem is also NP-Hard [17].

We begin with each node as a singleton. During the first level of clustering, we consider merging only node pairs which belong to the same cluster in the clustering of the previous iteration. When no more node pairs are left for merging, the current clustering is used to construct a "contracted graph" with each cluster as a super node and appropriately adjusted adjacencies and edge weights. We proceed iteratively with the contracted graph as input. The clustering of the previous iteration is explicitly used in the first level and afterwards we apply the algorithm of Blondel et al. [2].

We evaluate the effectiveness of our heuristic with a measure of cluster similarity given by Rand [33]. This measure is based on node-pair clustering as follows. Let C and C' denote two clusterings of a graph G, and let S_{11} denote the set of pairs that

are clustered together in both clusterings, and S_{00} denote the set of pairs that are in different clusters. Then the Rand distance between the two clusterings is given by

$$rand(C, C') = 1 - \frac{2(|S_{11}| + |S_{00}|)}{n(n-1)} \tag{5}$$

The value will be 0 if the two clusterings are identical, and 1 if one clustering is a singleton clustering and the other one is that of all nodes in the same cluster.

With the data from Fig. 9, we evaluated the quality between each pair of successive iterations and averaged these values over all successive pairs. Without our heuristic, the average Rand measure was 0.0631, and with the heuristic, it was 0.0252. This shows an improvement of a little more than 60 % with the heuristic.

4.4 Stable Map Coloring

Color stability, that is, using the same color for countries on the two maps that share most of their nodes, is an essential ingredient in visualizing dynamic maps. In order to maintain color stability, we need to match the best pairs of clusters in different maps.

Given two maps, let C_{old} and C_{new} be vectors representing clustering information of these two maps. We have to minimize the number of nodes whose cluster is different in C_{old} and C_{new}. Let $s(C_{old}, C_{new})$ be the number of nodes that do not undergo clustering change.

$$s(C_{old}, C_{new}) = \sum_{u \in V} \delta(C_{old}(u), C_{new}(u)); \tag{6}$$

$\delta(u, v) = 1$ if $u = v$, and 0 otherwise.

The cluster matching problem is to find a permutation Π of the clustering C_{new}, such that $\Pi(C_{new})$ maximizes $s(C_{old}, \Pi(C_{new}))$. For example, let $C_{old} = \{1, 1, 2, 2, 3\}$ be the clusters assigned to the five nodes v_1, v_2, v_3, v_4, v_5; let $C_{new} = \{2, 2, 1, 3, 4\}$ be the new clustering in which v_3 and v_4 split into two clusters. Clearly $s(C_{old}, C_{new}) = 0$. The optimum matching is the permutation $\Pi : \{1, 2, 3, 4\} \rightarrow \{2, 1, 4, 3\}$. The resulting clustering, $\Pi(C_{new}) = \{1, 1, 2, 4, 3\}$ gives $s(C_{old}, \Pi(C_{new})) = 4$.

The problem can be modeled with a maximum weighted matching (MWM) of a bipartite graph. The corresponding bipartite graph G_C has node set $\{1, 2, \ldots, |C_{old}|\} \times \{1, 2, \ldots, |C_{new}|\}$. The edge weight, $w(i, j)$, corresponds to the number of nodes that are common between cluster i of C_{old} and cluster j of C_{new}.

$$w(i, j) = \sum_{u \in V} \phi(i, j, u) \tag{7}$$

$\phi(i, j, u) = 1$ if $C_{old}(u) = i$ and $C_{new}(u) = j$.

The maximum weighted bipartite matching of G_C gives a matching Π between the clusters C_{old} and C_{new} that will maximize $s(C_{old}, \Pi(C_{new}))$. The MWM for bipartite graphs can be found using the Hungarian algorithm [26]. For bipartite graphs, an efficient implementation of the Hungarian algorithm using Fibonacci heaps [11] runs in $O(mn + n^2 \log n)$, where m and n are the number of edges and nodes in G_C, respectively. If we assume that a cluster in the old clustering does not split into more than a constant number of clusters in the new clustering, then $m = O(n)$. This yields a $O(n^2 \log n)$ algorithm for MWM. Since $w(i, j)$ are all integers in the range 0 to $|V|$, the algorithm by Gabov and Tarjan algorithm [13] for MWM can be implemented with $O(n^{\frac{3}{2}} \log(n|V|))$ complexity. In practice, the number of clusters is typically small and the Hungarian algorithm is fast enough.

5 Case Studies

In this section, we walk through some sample views of graphs as maps derived from real-world data and note aspects found in the drawings that might be perceived by a typical user. As noted above, the technique is intended for fairly large graphs.

Gleaning information from the maps, as with any large data set, typically involves an interactive, multi-scale process, similar to that used for exploring geographic maps. One can view the map at small scale to sense the overall layout, the major regions, and how they relate to each other. One then zooms in to see local detail, and to traverse the map along small features. At some point, one may zoom out again to put the local details into a global context.

Based on this style of use, our figures are most effectively 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 some version of semantic zoom, so that more detail is added the more the user zooms in. In addition, an interactive viewer can provide such additional features as textual search or links connecting a feature on the map to some external information. For example, clicking one of the books shown in Fig. 5 might take the user to the books entry at Amazon.com or to a Wikipedia article on the author.

It is important to note that in all cases the countries and their geography in the resulting maps are not part of the input data, but emerge from the graph layout and clustering algorithms. This gives the user a potential tool to discover structure based solely on local data. On the other hand, if a desired clustering is known, this can be used as the basis for “country” construction.

5.1 Maps and Recommendations

Recommender systems provide a motivating reason for displaying graphs as maps. Many content providers, both to assist their customers in making choices and to motivate them to make more selections, have systems to suggest additional picks based on various individual and group statistics, processed and refined with various algorithms. Typically, the user is provided with a small list or table of options, perhaps with some associated numbers giving some clue as to how the selections were made.

We feel that the map metaphor can give the user the relevant information in a more familiar way, with country placement and the underlying edges suggesting the connections. In addition, with the appropriate GUI, the user has access to a large volume of data, rather than a pruned list. With the full map, the user is not limited to a small region but can explore the map, following connections far from initial centers of interest. Guided to the movie *The Exorcist*, the user may wander down to *Babylon 5*.

The examples in the next two sections can be viewed as visual bases for recommender systems for movies and television shows, respectively. Indeed, the provenance of the movie data is directly tied to recommender systems.

5.2 A Map of Movies

Figure 11 shows a map derived from the data used for the Netflix competition to invent a better recommender algorithm. The underlying graph uses movies (and television shows) as nodes. Closely related shows are connected with an edge. In addition, the edges are weighted based on how strong the connection is. The base graph contains 11,283 nodes and 71,449 edges. Using a minimum edge weight as a threshold, we obtain a graph with 11,831 edges. Most of this graph resides in a single connected component with 3,407 nodes and 11,116 edges. It is this final graph that is used in the figure.

Zooming in, one can readily identify various countries. In the north, one finds a country of teen/adult animation (Fig. 12) containing the likes of *The Simpsons*, *Futurama* and *South Park*, with some Dave Chappelle shows pulled into the mix. These last form a segue to the purple country to the right (Fig. 13) containing shows with a certain style of adult humor, such as *Saturday Night Live*, *George Carlin* and *Chris Rock*.

At the bottom of the map, we find two adjacent countries (Fig. 14). The more southerly consists of classic, space-based science fiction shows such as *Star Wars*, *Star Trek*, *Stargate* and *The X-Files*. To the north we discover the eerier science fiction of *The Twilight Zone* and *The Outer Limits*. Moving clockwise around the periphery of the map, we encounter clearly defined regions of Japanese films; *Mystery Science Theater 3000* shows; Michael Moore documentaries; cerebral

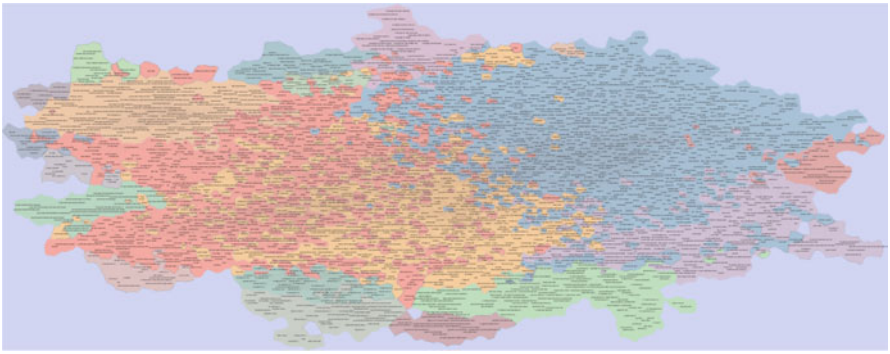

Fig. 11 A map of movies and TV shows (3,407 nodes, 11,116 edges)

Fig. 12 Mostly teen/adult animation

British detectives such as Campion and Lord Wimsey abutting a whole separate country of British mysteries; Ken Burns documentaries; horror films such as *Nightmare on Elm Street*; and several contiguous countries of juvenile fare.

5.3 Personalized Recommendations

Most of the maps we have seen so far have been constructed independent of any particular person, with data based on the aggregate behavior of many people. This information can then be tailored to an individual. For example, starting from a map of related TV shows, we could generate a personalized heat map version where regions of low interest are colored with cool colors, and regions containing highly

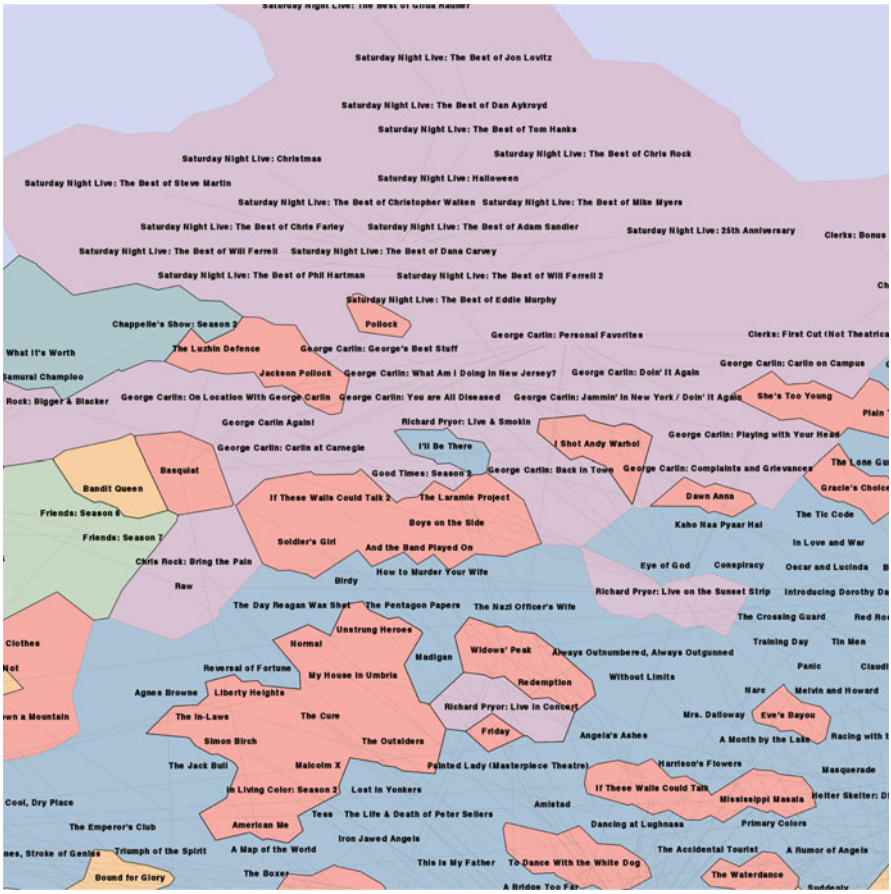

Fig. 13 *Saturday Night Live* and friends

recommended shows are colored with a hot color. Figure 15 shows such a heat map, where shows are scored using a factorization based recommender [22], with dark colors for shows that score low, and light for shows that score high. Such maps would be generated dynamically based on the viewing preference of an individual, and based on what TV shows are available at this moment in time, much like a personalized weather forecast, but for TV shows. These maps uniquely capture the viewing preferences of the user or household, and evolve as the availability of TV shows, and the user's taste, change with time. We can also generate a heat map profile, determined by how often the user watches certain shows over a fixed time period, say, a week or a month. Handling such fluid scenarios well requires the dynamic techniques discussed in Sect. 4 to be well honed.

Note that this approach could be extended to social networks: What recommendations could be made that might appeal to my friends on Facebook as a group?

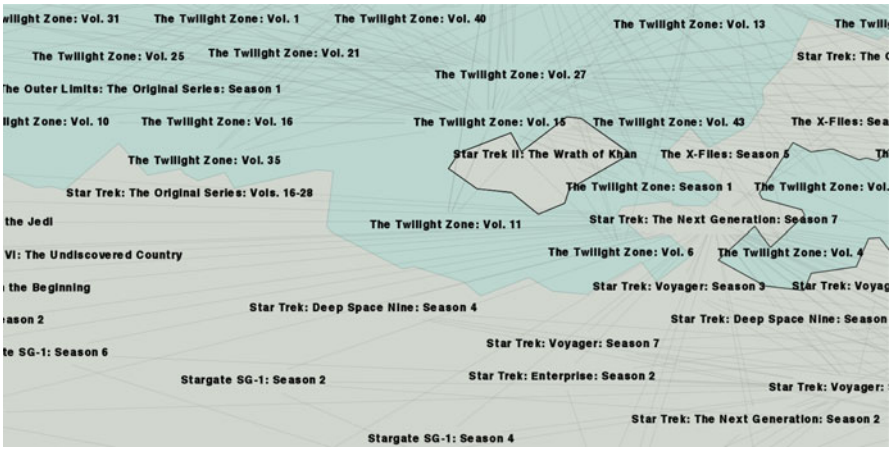

Fig. 14 Parts of two science fiction countries

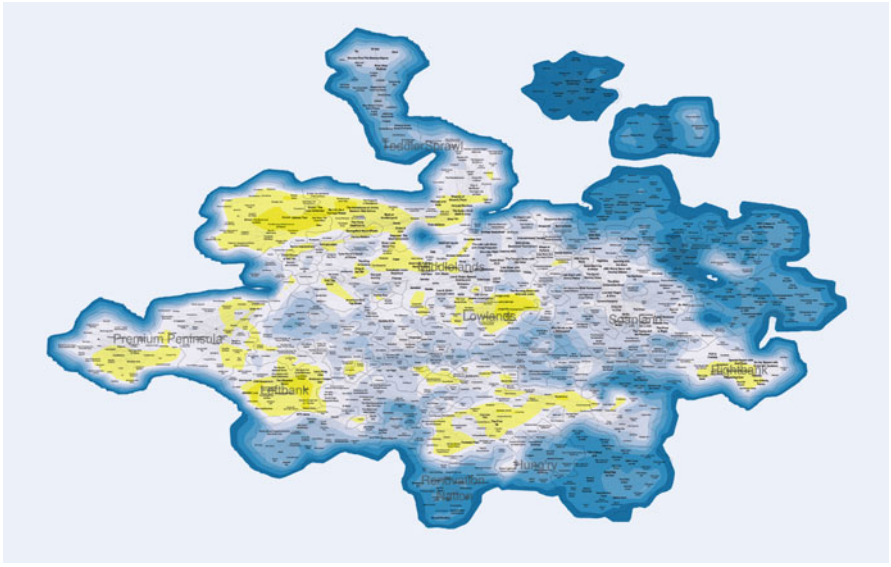

Fig. 15 Sample heat map giving personalized recommendations

5.4 Trade

Figure 16 is a map visualizing the trade relations between all countries. Bilateral trade data between each of the 209 countries and its top trading partners were acquired from Mathematica's `CountryData` package. The font size of a label is

Fig. 16 A map of trade relations between countries

proportional to the logarithm of the total trade volume of the country, and the color of a label reflects whether a country has a trade surplus (black) or deficit (red).

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

Many countries in close geographic proximity end up close in our map, e.g., Central American countries like 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 close to each other in the northwest. This is easily explained by noting that geographically close countries tend to trade with each other. There are easy-to-spot exceptions: North Korea is not near South Korea, Israel is not particularly close to Jordan or Syria.

The G8 countries (Canada, France, Germany, Italy, Japan, Russia, United Kingdom, and the United States) are all in close proximity to each other in the center of the map. Two of the largest and closest countries in our map are China and the United States. Clearly, the proximity is due to the very 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, we see from the map that African countries are distributed in several clusters in close proximity to China (a major trading partner to many African countries), the United States (trading less with Africa these days), and around former colonizers (e.g., Togo, Cameroon and Senegal, which are all close to France). On the other hand, Caribbean and South and Central American countries form several clusters in the north of the map. In addition, these clusters are mostly contiguous, essentially forming a supercluster. This differentiation between Latin America and Africa is clearly brought out by the GMap figure.

Finally, we note that the periphery of the map contains small countries from around the world, and countries with few trading partners.

6 Final Thoughts

One rule that most researchers in information visualization discover is that people tend to be most comfortable and adept when they see data the way they have always seen it. Ignoring this can lead to a "demonstrably" better but alien visualization being rejected by its intended users. For us, this provides a strong argument for the effectiveness of displaying graphs and clusters as geographic maps to promote a more human centric visualization. At present, we have mostly anecdotal and non-scientific evidence that this is a preferable way for viewing relational data. But we have seen people spend long periods of time poring over these maps. It would be desirable to perform more extensive user studies to explore how well maps compare to other metaphors, and to explore more ways to use maps. At the same time, there is still much work to be done enhancing and tuning the underlying algorithms, especially in the context of dynamic graphs.

Acknowledgements We would like to thank Stephen North and Chris Volinsky for helpful discussions and encouragement.

References

1. Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: Algorithms for the visualization of Graphs. Prentice-Hall (1999)
2. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Stat. Mechanics: Theory and Experiment **2008**, P10,008 (2008)
3. Brewer, C.: ColorBrewer - selecting good color schemes for maps. www.colorbrewer.org
4. Cappanera, P.: A survey of obnoxious facility location problems. Technical Report TR-99-11, Dipartimento di Informatica, Universit a di Pisa (1999)
5. Cleveland, W.S.: Visualizing Data. Hobart Press, Summit, New Jersey, U.S.A. (1993)
6. Dillencourt, M.B., Eppstein, D., Goodrich, M.T.: Choosing colors for geometric graphs via color space embeddings. In: 14th Symposium on Graph Drawing (GD), pp. 294–305 (2006)
7. Duarte, A., Martí, R., Resende, M., Silva, R.: GRASP with path relinking heuristics for the antibandwidth problem. Networks (2011). Doi: 10.1002/net.20418

8. Erten, C., Harding, P.J., Kobourov, S.G., Wampler, K., Yee, G.V.: Graphael: Graph animations with evolving layouts. In: G. Liotta (ed.) Graph Drawing, *Lecture Notes in Computer Science*, vol. 2912, pp. 98–110. Springer (2003)
9. Fabrikant, S.I., Montello, D.R., Mark, D.M.: The distance-similarity metaphor in region-display spatializations. IEEE Computer Graphics & Application **26**, 34–44 (2006)
10. Fabrikant, S.I., Montello, D.R., Mark, D.M.: The natural landscape metaphor in information visualization: The role of commonsense geomorphology. JASIST **61**(2), 253–270 (2010)
11. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**, 596–615 (1987). DOI http://doi.acm.org/10.1145/28869.28874. URL http://doi.acm.org/10.1145/28869.28874
12. Fuchs, G., Schumann, H.: Visualizing abstract data on maps. In: Proceedings of the Information Visualisation, Eighth International Conference, IV '04, pp. 139–144. IEEE Computer Society, Washington, DC, USA (2004). DOI 10.1109/IV.2004.152. URL http://dx.doi.org/10.1109/IV.2004.152
13. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. **18**, 1013–1036 (1989). DOI 10.1137/0218069. URL http://portal.acm.org/citation.cfm?id=75795.75806
14. Gansner, E.R., Hu, Y.F., Kobourov, S.G.: Gmap: Drawing graphs as maps. http://arxiv1.library.cornell.edu/abs/0907.2585v1 (2009)
15. Gansner, E.R., Hu, Y.F., Kobourov, S.G., Volinsky, C.: Putting recommendations on the map - visualizing clusters and relations. In: Proceedings of the 3rd ACM Conference on Recommender Systems. ACM (2009)
16. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Softw., Pract. Exper. **30**(11), 1203–1233 (2000)
17. Görke, R., Maillard, P., Staudt, C., Wagner, D.: Modularity-driven clustering of dynamic graphs. In: 9th Symp. on Experimental Algorithms, pp. 436–448 (2010)
18. *Graphviz* graph visualization software. www.graphviz.org/
19. Hu, Y., Gansner, E.R., Kobourov, S.G.: Visualizing graphs and clusters as maps. IEEE Computer Graphics and Applications **30**(6), 54–66 (2010)
20. Hu, Y., Kobourov, S., Veeramoni, S.: On maximum differential graph coloring. In: Proceedings of the 18th international conference on graph drawing (GD'10), pp. 274–286. Springer-Verlag (2011)
21. Hu, Y., Kobourov, S., Veeramoni, S.: Embedding, clustering and coloring for dynamic maps. In: Proceedings of IEEE Pacific Visualization Symposium. IEEE Computer Society (2012)
22. Hu, Y.F., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 8th IEEE International Conference on Data Mining (ICDM), pp. 263–272 (2008)
23. Hu, Y.F., Scott, J.A.: A multilevel algorithm for wavefront reduction. SIAM Journal on Scientific Computing **23**, 1352–1375 (2001)
24. Keim, D.A., Panse, C., North, S.C.: Medial-axis-based cartograms. IEEE Computer Graphics and Applications **25**(3), 60–68 (2005)
25. van Kreveld, M.J., Speckmann, B.: On rectangular cartograms. Comput. Geom. **37**(3), 175–187 (2007)
26. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955). DOI 10.1002/nav.3800020109. URL http://dx.doi.org/10.1002/nav.3800020109
27. Kuhn, W., Blumenthal, B.: Spatialization: spatial metaphors for user interfaces. In: Conference companion on Human factors in computing systems: common ground, CHI '96, pp. 346–347. ACM, New York, NY, USA (1996). DOI 10.1145/257089.257361. URL http://doi.acm.org/10.1145/257089.257361
28. Kumfert, G., Pothen, A.: Two improved algorithms for envelope and wavefront reduction. BIT **35**, 1–32 (1997)
29. Leung, J.Y.T., Vornberger, O., Witthoff, J.: On some variants of the bandwidth minimization problem. SIAM J. Comput. **13**, 650–667 (1984)

30. Lima, M.: Visual Complexity: Mapping Patterns of Information. Princeton Architectural Press (2011)
31. Lloyd, S.: Last square quantization in pcm. IEEE Transactions on Information Theory **28**, 129–137 (1982)
32. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA **103**, 8577–8582 (2006)
33. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. of the American Statistical Association pp. 846–850 (1971)
34. Raspaud, A., Schröder, H., Sýkora, O., Török, L., Vrt'o, I.: Antibandwidth and cyclic antibandwidth of meshes and hypercubes. Discrete Mathematics **309**, 3541–2552 (2009)
35. Salvini, M.M., Gnos, A.U., Fabrikant, S.I.: Cognitively plausible spatialization of network data. In: Proceedings of the 20th International Cartographic Conference (2011)
36. Scott, J., Hu, Y.: Level-based heuristics and hill climbing for the antibandwidth maximization problem. Technical Report RAL-TR-2011-019, Ritherford Appleton Laboratory, UK (2011)
37. Skupin, A.: A cartographic approach to visualizing conference abstracts. IEEE Computer Graphics & Application **22**(1), 50–58 (2002)
38. Skupin, A.: The world of geography: Visualizing a knowledge domain with cartographic means. Proc. National Academy of Sciences **101**(Suppl. 1), 5274–5278 (2004)
39. Skupin, A.: Discrete and continuous conceptualizations of science: Implications for knowledge domain visualization. Journal of Informetrics **3**(3), 233–245 (2009)
40. Skupin, A., Buttenfield, B.P.: Spatial metaphors for visualizing information spaces. In: Proc. AUTO-CARTO 13, pp. 116–125 (1997)
41. Skupin, A., Fabrikant, S.I.: Spatialization. In: Handbook of Geographic Information Science, pp. 61–80. Blackwell Publishers (2008)
42. Sloan, S.W.: An algorithm for profile and wavefront reduction of sparse matrices. International Journal for Numerical Methods in Engineering **23**, 239–251 (1986)
43. Steele, J., Iliinsky, N.: Beautiful Visualization: Looking at Data through the Eyes of Experts, 1st edn. O'Reilly Media, Inc. (2010)
44. Ullman, J.D.: Elements of ML programming - ML 97 edition. Prentice Hall (1998)