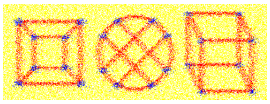

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)
DOI:

Embedding, Clustering and Coloring for Dynamic Maps

Yifan Hu[1] *Stephen G. Kobourov*[2] *Sankar Veeramoni*[2]

[1]AT&T Labs – Research, Florham Park, NJ
[2]Dept. of Computer Science, University of Arizona, Tucson, AZ

Abstract

We describe a practical approach for visualizing multiple relationships defined on the same dataset using a geographic map metaphor, where clusters of nodes form countries and neighboring countries correspond to nearby clusters. Our aim is to provide a visualization that allows us to compare two or more such maps (showing an evolving dynamic process, or obtained using different relationships). In the case where we are considering multiple relationships, e.g., different similarity metrics, we also provide an interactive tool to visually explore the effect of combining two or more such relationships. Our method ensures good readability and mental map preservation, based on dynamic node placement with node stability, dynamic clustering with cluster stability, and dynamic coloring with color stability.

Submitted:	Reviewed:	Revised:	Reviewed:	Revised:
April 25, 2013	August 1, 2013	September 6, 2013		
	Accepted:	Final:	Published:	
	Article type: Regular paper	Communicated by: A. Editor		

Research funded in part by NSF grants CCF-0545743, CCF-1115971.

A preliminary version of this paper appeared in the 5th IEEE PacificVis Symposium (2012). The current expanded version applies the proposed technique to new application domains (e.g., TV landscape that evolves over time), including a purely geographic one (clustering of regions in the US based on social interaction metrics).

E-mail addresses: yifanhu@research.att.com (Yifan Hu) kobourov@cs.arizona.edu (Stephen G. Kobourov) sankar@cs.arizona.edu (Sankar Veeramoni)

1 Introduction

Maps offer a familiar way to present geographic data (continents, countries, states). Contact graphs (e.g., [15]), where regions represent nodes and edges are represented by the corresponding regions sharing borders, are an example of map representations. By definition, contact graphs are limited to planar graphs, but the notion of a map representation can be generalized to non-planar graphs as follows: clusters of well-connected nodes form countries, and countries share borders when neighboring clusters are interconnected. In the context of graph drawing, such maps (e.g., [27]) allow us to show not only connectivity information in the underlying data (via nodes and edges between them), but also clustering (via countries). Specifically, by grouping nodes into different colored regions, we can easily see individual clusters and relations between the clusters. Such explicit grouping makes it easy to identify central and peripheral clusters, as well as central and peripheral nodes within clusters. Finally, cut-nodes and edges, often on the border between two clusters, make it clear which nodes and edges connect disparate parts of the data.

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 (e.g., [4]). 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[32]. 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 visualization for dynamic data and for multiple relationships on the same dataset.

1.1 Motivation

Our motivation comes from population genetics, where one of the major problems is better understanding of the genomic and evolutionary factors shaping human variation and testing models of human origins. Relatively inexpensive human genome sequencing technology allows for the study of patterns of variation across the entire genome in multiple human populations. When genetic information passes down from parents to children some special parts of the DNA remain virtually unaltered as they pass from parent to child. One such piece carrying paternal information is encoded in the non-recombining portion of the Y chromosome (NRY). Another piece carrying maternal information is encoded in the mitochondrial DNA (mtDNA). When comparing DNA information from a group of people, the NRY DNA similarity score, the mtDNA similarity score, and others can be computed from different parts of the DNA. Such scores can give an indication about the strength of the genetic similarity between the individuals. Interactively combining these multiple scores in an intuitive and visually appealing fashion facilitates knowledge discovery and hypothesis testing.

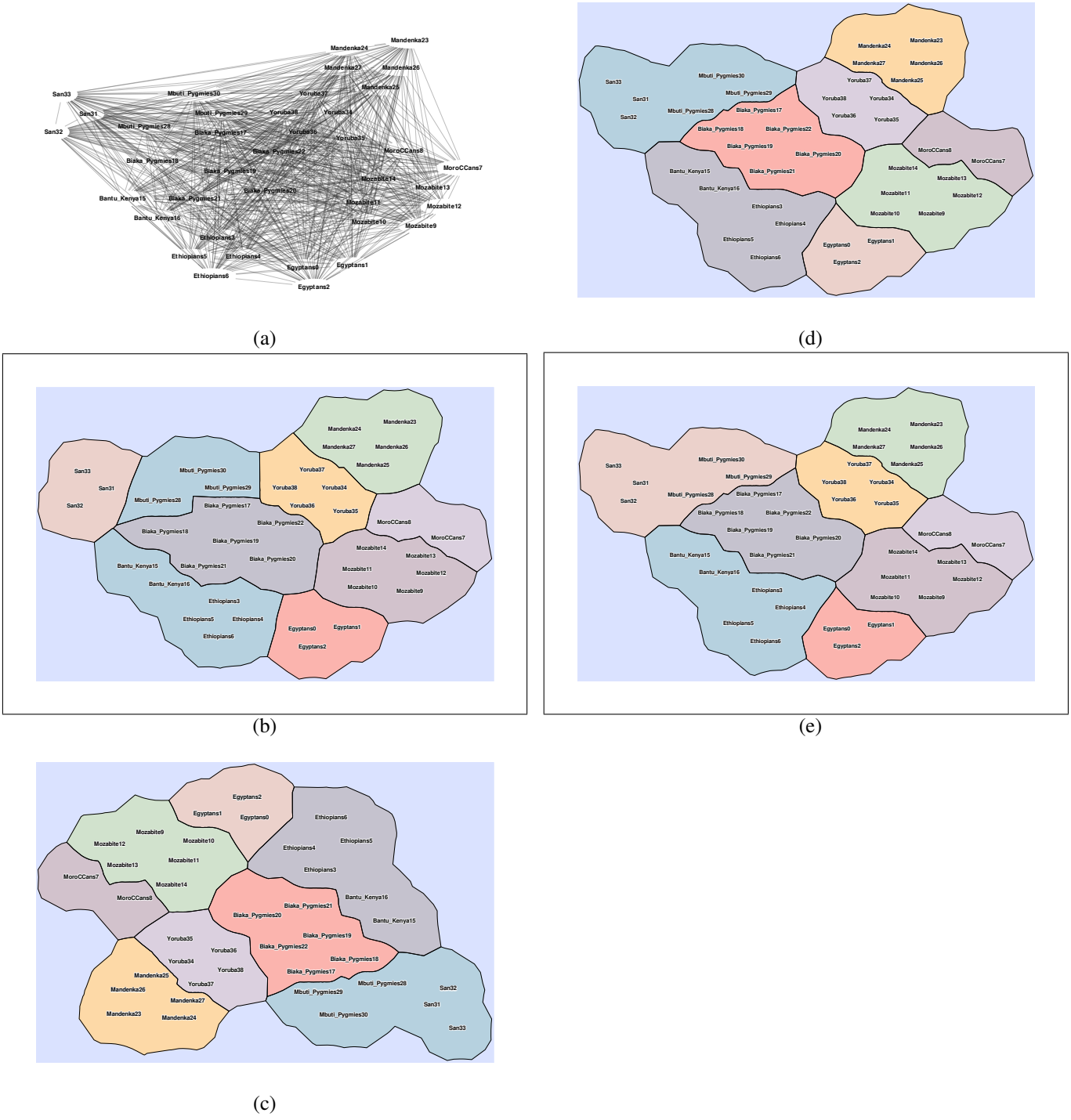


Figure 1: Visualizing multiple maps requires both layout stability and color stability. (a): mtDNA similarity visualized as a graph. (b): mtDNA similarity visualized as a map. (c): NRY DNA similarity map which is difficult to compare with (b) as the node layout is computed independently. (d): NRY DNA similarity map, computed to optimize node layout stability with regard to (b), which makes it possible to compare nodes, while clusters are still hard to compare. (e): NRY DNA similarity map with optimal node layout and color assignments, which makes it easy to compare with (b); e.g., it is clear that two clusters in the top left of (b) are now merged.

1.2 Our Contributions

With this in mind, our first goal is to provide a way to compare two or more maps of the same objects, obtained using different similarity metrics. Our second goal is to provide an interactive tool to visually explore the effect of combining such similarity metrics. The first problem is related to simultaneous graph drawing, and the second problem is related to dynamic graph drawing. However, in our setting we have to deal not only with the traditional problems of finding suitable placement of nodes and edges, but also with the new problems of dynamic clustering and dynamic color assignment; see Fig. 1. Thus the main contributions of the paper are the following: first, we describe a heuristic to promote dynamic *cluster stability* (nodes from one cluster in one map should be kept in the same cluster in the other map, provided that doing so does not give suboptimal clustering); second, we propose an optimal color assignment algorithm to maximize *color stability* between maps (using the same color for countries that share most of their nodes); third, we analyze the use of Procrustes transformation to improve *layout stability* (node placement should be determined by the underlying similarity metric, while at the same time not differ unnecessarily from one map to the other). We describe an interactive tool for DNA visualization that utilizes these results. Finally, we apply the new methodology to the visualization of several additional data sets. In two data sets (evolution of the TV landscape and evolution of a collaboration network) we apply the methodology to the more traditional dynamic network visualization problem. In the other two the data sets (international trade map of the world and interaction maps of the US) we use part of our methodology (e.g., color stability) to help see patterns in regular geographic maps.

The paper is organized as follows. Related work is discussed in Section 1.3. Section 2 describes the dynamic map visualization problems that we address. Section 3 discusses clustering algorithms, in particular a heuristic for promoting clustering stability. Section 4 proposes to solve the color stability problem via maximal bipartite matching algorithms. Sections 5 analyzes the use of Procrustes and affine transformation for layout stability. Section 6 includes implementation of an interactive tool for exploration of DNA data, as well as examples of our methodology applied to mobile, TV and research collaboration data sets. We conclude the paper in Section 7 with discussion and future work.

1.3 Related Work

Our work builds on the GMap algorithm [27], which visualizes relational data and clustering information using a map metaphor. The map metaphor has also been used for information visualization in the cartographic community [19]. Cortese *et al.* [12] use a topographical map metaphor to visualize prefixes propagation in the Internet, where contour lines describing the propagation are calculated using a force directed algorithm. Also related is work on visualizing subsets of a set of items using geometric regions to indicate the grouping. Byelas and Telea [8] use deformed convex hulls to highlight areas of interest in UML diagrams. Collins *et al.* [11] use "bubblesets," based on isocontours, to depict multiple relations among a set of objects. Simonetto *et al.* [37] automatically generate Euler diagrams which provide one of the standard

ways, along with Venn diagrams, for visualizing subset relationships. The emphasis of such prior work is on highlighting sets of objects using the map metaphor, rather than on visualizing dynamic maps.

Initial work on dynamic maps was presented in the context of visualizing trends on Internet radio station last.fm [31]. In this offline setting all data are available in advance, the similarity between the objects is fixed, and the mental map is preserved by laying out and clustering all objects at once, but only showing relevant subsets that appears in a particular time frame. In contrast, in this paper all data are not assumed to be known in advance, the similarity among objects changes overtime, and we solve the problem of maintaining stability in node layout, clustering and map coloring.

A context-preserving dynamic text data visualization by Cui *et al.* [14], keeps the layout stable using a spring embedding algorithm, aided by a mesh constructed over existing nodes. Neither of these two approaches takes cluster and color stability under consideration, either because it is not relevant (in text data), or because clusters are fixed (in last.fm visualization). Holten and van Wijk [26] study the visual comparison of hierarchically organized data, and propose a method that highlights the differences. Similarly, the TimeRadarTrees of Burch and Diehl [7] are proposed to visualize the evolution of compound digraphs. Although these two approaches deal only with directed graphs and cannot be applied directly to our problem, they are related to the problem of highlighting similarities and differences in dynamic data, which is the aim of this paper.

Also related to our work is a large body of work on dynamic graph layout, where the goal is to maintain a nice layout of a graph that is modified via operations such as inserting/deleting edges and inserting/deleting nodes. Brandes and Wagner adapt the force-directed model to dynamic graphs using a Bayesian framework [5], while Herman *et al.* [25] rely on good static layouts. Diehl and Görg [16] consider graphs in a sequence to create smoother transitions. Brandes and Corman [2] present a system for visualizing network evolution in which each modification is shown in a separate layer of a 3D representation, with nodes common to two layers represented as columns connecting the layers. Thus, mental map preservation is achieved by pre-computing good locations for the nodes and fixing the position throughout the layers. While the issue of mental map preservation adopted in such approaches remains an important one for us, we note that in our population genetics application we want to avoid a distortion of the layout for the purpose of mental stability. This is because such distortions introduce biases that were not in the data itself, which is not important when visualizing the evolution of citation graphs or Internet graphs, but are undesirable when analyzing DNA similarity. This is also why we do not use techniques such as anchoring, or artificial edges linking the same node in different time frame [18], in the population genetics application. However we do use such or similar techniques in other applications in Section 6.

In the area of visualization of DNA data Yang *et al.* [33] study ancestry and ethnic affiliation using DNA genetic markers and make accurate predictions with the help of a Bayesian clustering algorithm. Novembre *et al.* [29] used principal component analysis to embed high dimensional genetic data for 3000 European individuals into 2-dimensional space, and observed a very high correspondence between the embedding distance and the geographic distance between pairs of individuals. The authors colored

the 2D data points (representing individuals) based on their country of origin, and observed a similarity between the colored point cloud and a map of Europe. The main difference to our paper is that they only have one metric of similarity, while we combine multiple metrics and use an interactive visualization to help understand the effect of different metrics. As a result we have to solve the problem of providing a stable mental map to facilitate comparison and contrast of between many (rather than just one) maps.

Clustering dynamic data to maintain the dual objectives of making the clustering faithful to the current data as much as possible, while avoiding dramatic changes from one time step to the next, is an active area of research (e.g., [24]). While there are attempts to minimize a combined cost function (e.g., [10]), we are not aware of specific work that maximizes modularity while maintain clustering stability, as our simple heuristic in Section 3 does.

2 Dynamic Map Visualization

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. However, this 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.

The maps in Fig. 1 show the nature of the problem. In Fig. 1 (a), 39 subjects are embedded in 2D space based on mtDNA similarity, using multi-dimensional scaling. Fig. 1 (b) shows a map obtained from the layout in (a), with subjects clustered using modularity clustering, and the clusters highlighted using the GMap algorithm. Here and later on, we used color palettes from ColorBrewer [6]. In Fig. 1 (c), we re-embedded the subjects using a different similarity metric (NRY DNA similarity), independent of Fig. 1 (b). Compared with Fig. 1 (b), 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. 1 (b) and Fig. 1 (c). In Fig. 1 (d), the embedding of NRY DNA similarity is done to minimize the difference to that of the embedding based on mtDNA similarity measure, making it possible to see that node positions are largely unchanged. However due to the color

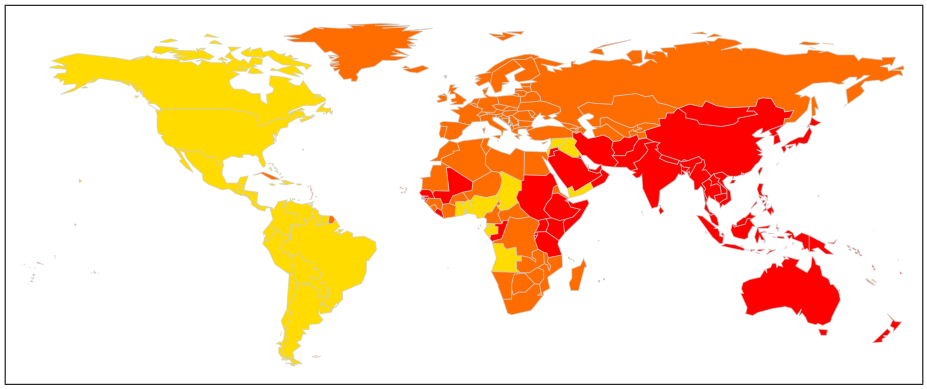
assignment, it is still difficult to compare with the other map in Fig. 1 (b). Finally, in Fig. 1 (e), colors are properly matched such that clusters with mostly the same nodes are colored using the same colors. This makes it easy to compare Figures 1 (b) and (e). For example, we can clearly see that two countries in the top left are now merged in a single country.

We seek to create an interactive tool that can smoothly animate from one similarity map to another, by dynamically changing the weight given to each of the two metrics using a slider that implements a linear combination. This approach can be generalized to three metrics with the help of an equilateral triangle in which each interior point corresponds to a convex combination of the three metrics.

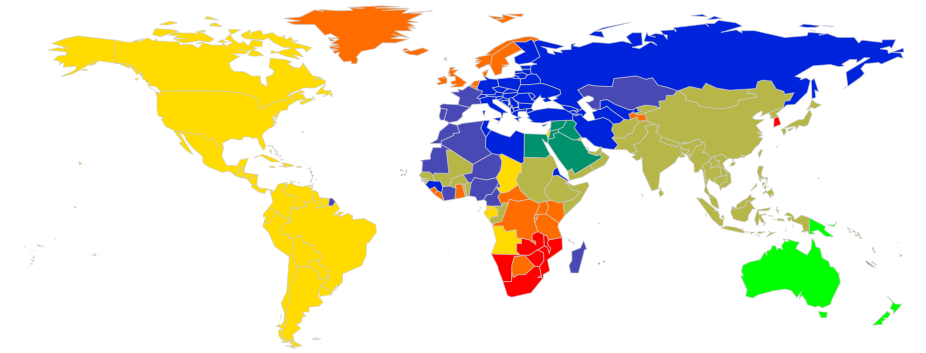
To construct such a tool requires us to overcome a number of challenges outlined earlier and illustrated in Fig. 1, namely: *layout stability* (node placement should be determined by the underlying similarity metric, while at the same time not differ unnecessarily from one map to the other), *cluster stability* (nodes from one cluster in one map should be kept in the same cluster in the other map, provided that doing so does not give suboptimal clustering), and *color stability* (using the same color for countries that share most of their nodes). All the above serves to facilitate the quick discovery of differences and similarities between two maps, while not being distracted by meaningless differences in layout, or by meaningless changes in groupings of nodes, or by meaningless changes in colors.

Note that the problem of color stability applies equally well to real geographic maps that have been clustered based on two different metrics. For example, Fig. 2 shows clusters of countries in Europe, Asia, Africa and Oceania based on two types of international trade metrics. We generate a country graph where each country is a node, and an edge exists between two countries if they trade with each other. Fig. 2 (a) is clustered using the total import/export values among countries as edge weights. Fig. 2 (b) is clustered using the total import/export fractions among countries as edge weights. (The import/export fraction is a number between 0 to 1, representing the percentage of import/export from one country to another, as a fraction of its total import/export value.) It is difficult to compare the maps in (a-b), because the colors for the clusters are chosen at random. On the other hand, when we apply color-stabilization we obtain the pair of maps (a-c), where clusters in the two maps that share many countries are colored using the same color. Here it is easy to see that the new clusters in Europe match closely with their former colonies in Africa: UK and former colonies in British East Africa such as Kenya, Uganda, Tanzania, as well as France and former colonies Algeria, Morocco, Mauritania. The large red cluster in (a) with China, India, Japan, Korea has the same color in map (c). Consider also the "not in the Eurozone" light green cluster in (c) with the UK, Greenland, Norway, and Sweden that has split away from the large European orange cluster in (a). The latter cluster inherits the orange color because it contains more countries, and all these countries were in the same orange cluster in (a). The application of our color-stabilization algorithm makes it easier to see such similarities and changes.

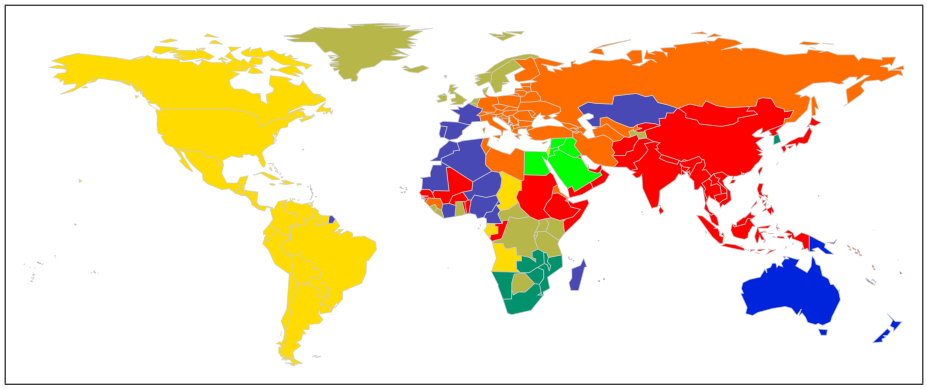
In the following section we discuss some of the background algorithms we used as a basic building block for our interactive tools.



(a)



(b)



(c)

Figure 2: World trade maps: (a) clustering of countries based on total import/export values between countries; (b) clustering of countries based on total import/export fractions; (c) same clustering as (b), but with color stability w.r.t. (a).

3 Dynamic Map Clustering

Our dynamic map clustering is based on static graph clustering. With this in mind we briefly review modularity graph clustering for static graphs and then consider the dynamic problem in the context of maps.

3.1 Modularity clustering

In graph clustering we look for a partition of the nodes into disjoint subsets so that most of the edges go between nodes within a cluster and there are not too many edges between nodes in different clusters. Formally, it is a mapping from the node set to a set of integers $C : V \to \{1,2,\ldots,k\}$, where $k = |C|$ is the number of clusters. Roughly speaking, clustering algorithms are either divisive or agglomerative. Agglomerative algorithms merge similar nodes or communities recursively, while divisive algorithms detect the inter-cluster edges and remove them from the network. In both cases, the goal is to optimize some cluster quality measure. One such measure is modularity, defined as a value between -1 and 1 that measures the density of links inside communities compared to links between communities [34, 35]. For a weighted undirected graph $G = (V,E)$ modularity is defined as follows:

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

Here $A_{i,j}$ is the weight of the edge between nodes i and j and it represents the similarity between these two nodes (e.g., the number of mutual friends in a social network). 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 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.

Finding a clustering which will maximize modularity is known to be a computationally hard problem, as shown by Brandes *et al.* [3]. We implemented a heuristic algorithm proposed by Blondel *et al.* [1] (available as the `Graphviz` [17] function `cluster`).

3.2 Dynamic 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.

We propose 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.* [1]. Heuristics are a reasonable approach, as the dynamic modularity clustering problem is also NP-hard [23].

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.* [1]. We give the algorithm in a pseudo code below, where $C_{\text{old}}(i)$ denotes the clustering of node i in the previous time frame.

```
Assign each node to a singleton cluster.
Set iteration = 1 and improved = true
while improved do
  improved = false
  for each node i do
    for each neighbor node j of i do
      if iteration == 1 && C_old(i) ≠ C_old(j) then
        continue
      else if merging i and j improves modularity then
        improved = true
        merge nodes i and j
        update the graph
      end if
    end for
  end for
  iteration = iteration + 1
end while
```

Algorithm 1: Dynamic Clustering

We evaluate the effectiveness of our heuristic with a measure of cluster similarity given by Rand [36]. This measure is based on counting cluster membership of each of the $n(n-1)/2$ pairs of nodes ($n = |V|$) 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

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

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. 1, we evaluated the quality between each pair of successive iterations (out of 100 iterations) and averaged these values over all successive pairs. Without our heuristic, the average Rand distance was 0.0631, and with the heuristic, it was 0.0252. This shows an improvement of a little more than 60% with the heuristic. Note that this does not negatively impact the modularity value of the clustering. To verify this, we computed the modularity of the clustering of the maps obtained at each iteration. In 64 of the 100 iterations there are no changes to the Rand distance and

hence no changes in modularity. In the 36 instances where the Rand distance with the heuristic is different from the Rand distance without the heuristic, there was little impact on modularity. In 16 iterations the modularity with our heuristic was slightly better than the one without the heuristic and in 20 iterations the modularity was slightly worse. Specifically, the average modularity value with the heuristic was 0.274 and the average modularity without the heuristic was 0.275.

We collected similar data for another dataset (evolving TV landscape in Section 6.3). For this dataset we evaluated the quality between each pair of 38 successive maps. Without our heuristic, the average Rand distance was 0.1832, and with the heuristic, it was 0.1358. This shows an improvement of a little more than 26% with the heuristic. We also computed the modularity of the clustering of the maps obtained at each iteration. We consider the 33 iterations where the Rand distance with the heuristic is different from the Rand distance without the heuristic. In 28 of these iterations, the modularity with our heuristic was slightly better than the one without the heuristic and in 5 iterations the modularity with our heuristic was slightly worse than the one without the heuristic. Specifically, the average modularity value with the heuristic was 0.3648 and the average modularity without the heuristic was 0.3315.

4 Dynamic 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 of the same underlying graph $G = (V,E)$, let C_{old} and C_{new} be vectors representing clustering information of these two maps. Let $u_1, u_2 \cdots u_{|V|}$ be some ordering of the vertices V in G. We have to minimize the number of nodes whose cluster is different in C_{old} and C_{new}. Let $s(C_{\text{old}}, C_{\text{new}})$ be the number of nodes that do not undergo clustering change:

$$s(C_{\text{old}}, C_{\text{new}}) = \sum_{u_i \in V} \delta(C_{\text{old}}[i], C_{\text{new}}[i]), \tag{3}$$

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

The cluster matching problem is to find a permutation Π of the clustering C_{new}, such that $\Pi(C_{\text{new}})$ maximizes $s(C_{\text{old}}, \Pi(C_{\text{new}}))$. For example let $C_{\text{old}} = \{1,1,2,2,3\}$ be the clusters assigned to the 5 nodes v_1, v_2, v_3, v_4, v_5; let $C_{\text{new}} = \{2,2,1,3,4\}$ be the new clustering in which v_3 and v_4 split into two clusters. Clearly $s(C_{\text{old}}, C_{\text{new}}) = 0$. The optimum matching is the permutation $\Pi : \{1,2,3,4\} \rightarrow \{2,1,4,3\}$. The resulting clustering, $\Pi(C_{\text{new}}) = \{1,1,2,4,3\}$ gives $s(C_{\text{old}}, \Pi(C_{\text{new}}))$ = 4.

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

$$w(i,j) = \sum_{u_k \in V} \phi(i,j,u_k), \tag{4}$$

where $\phi(i,j,u_k) = 1$ if $C_{\text{old}}[k] = i$ and $C_{\text{new}}[k] = j$.

The maximum weighted bipartite matching of G_C gives a matching Π between the clusters C_{old} and C_{new} that will maximize $s(C_{\text{old}}, \Pi(C_{\text{new}}))$. The MWM for bipartite graphs can be computed with the Hungarian algorithm [30]. For bipartite graphs, an efficient implementation of the Hungarian algorithm using Fibonacci heaps [20] 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 Gabov and Tarjan algorithm for MWM [21] can be implemented in $O(n^{\frac{3}{2}}\log(n|V|))$. In practice, the number of clusters is typically small (less than 30) and our implementation of the Hungarian algorithms runs in less than a millisecond.

To evaluate our algorithm, we measure, via equation (3), how many nodes keep the same color in two successive maps. With the data from Fig. 1, we computed this quantity for 100 pairs of maps and took the average. With a random color assignment, on average 16.4 nodes remain the same color and 22.6 nodes are assigned different colors. With our color matching algorithm on average 34.27 nodes remain the same color and 4.73 nodes are assigned different colors, showing a non-trivial improvement. We also evaluated the effect of the clustering algorithm on the coloring algorithm. Without the clustering heuristic on average 31.8 nodes remain the same color and 7.2 nodes are assigned different colors, which corresponds to a 9% improvement.

We collected similar data for another dataset (evolving TV landscape in Section 6.3). For this dataset we computed the number of nodes whose color remains unchanged in 38 pairs of maps with 100 nodes each. With a random color assignment, on average 4.55 remain the same color and 95.45 are assigned different colors. With our color matching algorithm on average 78.27 TV nodes remain the same color and 21.73 TV shows are assigned different colors, again showing a substantial improvement. We also evaluated the effect of the clustering algorithm on the coloring algorithm. Without the clustering heuristic on average 72.33 nodes remain the same color and 27.66 nodes are assigned different colors, which corresponds to a 8% improvement.

5 Dynamic Map Layout

Abstractly, the problem of dynamic maps 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, if 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 approaches is that even with the node-position initialization two consecutive graphs in the sequence with very similar topology can have very different drawings, causing node-jumping between frames, and failing to preserve the mental map.

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 [18]. However, such approaches can introduce biases that were not in the data

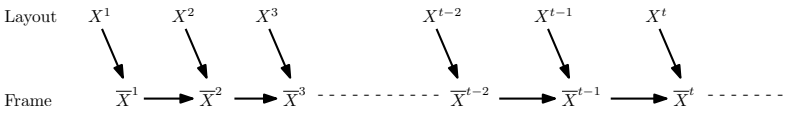


Figure 3: The affine transformation pipeline. X^t denotes the layout at time t. This layout is transformed to fit the previous frame $\bar{X}^{t-1}$ as close as possible yielding the current frame $\bar{X}^t$, which is what the user sees.

itself, which is undesirable when analyzing highly sensitive real-world data, such as DNA similarity.

5.1 Procrustes Transformation

We preserve the mental map via a Procrustes transformation of the coordinates of the nodes, following a multidimensional scaling (MDS) layout, initialized with the node positions from the previous iteration. The Procrustes transformation is a well-known technique [13], which attempts to find the best alignment of two inputs via rotation, translation, and scaling. Let the layout of the previous frame be $y_i,\ i = 1,2,\ldots,|V|$, and the current layout $x_i,\ i = 1,2,\ldots,|V|$ (if the node sets in the two layout are different, we take the coordinates for common nodes to derive the transformation). We would like to find a translation vector b, scaling value ρ and rotation matrix T such that:

$$\text{minimize} \sum_{i=1}^{|V|} \|y_i - (\rho T x_i + b)\|^2. \tag{5}$$

The solution to this problem is

$$T = (XY^TYX^T)^{1/2}(YX^T)^{-1},\ \rho = \frac{tr((XY^TYX^T)^{1/2})}{tr(XX^T)},$$

where X is the $2 \times |V|$ matrix of x_i's. The translation vector is $b = \frac{1}{|V|}(\sum_{i=1}^{|V|} y_i - \rho T(\sum_{i=1}^{|V|} x_i))$. The minimal value of (5),

$$1 - \frac{tr((X^TYY^TX)^{1/2})}{tr(X^TX)tr(Y^TY)}$$

is known as the Procrustes Statistic.

Note that if we do not restrict T to be an orthogonal rotation matrix, then ρ is subsumed by T and we have a general affine transformation, with $T = YX^T(XX^T)^{-1}$.

The difference between the Procrustes transformation, and a general affine transformation, is that the Procrustes transformation preserves the relative distance between nodes. Fig. 3 shows the pipeline of how the transformation is applied to a sequence of graphs.

We experimented with both the Procrustes and affine transformations, and found that although affine transformation are better at minimizing the differences between

two frames, the Procrustes transformation is good enough, but with the added benefit of preserving relative distances. However, when using either in dynamic map visualization involving many frames, we found an unexpected problem: On inputs where the first frame is tall and thin, and the second frame is a square with the same height as the first, the image size after transformation shrinks as we progress through the time steps, even beyond the second frame. Further analysis confirms that although each transformation is strictly derived from the current layout, and the previous frame, the Procrustes transformations have a memory of layouts that goes all the way to the first layout. This can be proven as the following theorem, which relates the layout at time t, X^t, and the frame at time t, $\bar{X}^t$, obtained by the attempt to "fit" X^t with the previous frame $\bar{X}^{t-1}$:

Theorem 1 *Let $\rho(Y,X)$ be the scaling factor for the Procrustes transformation of points $\{x_1,\ldots,x_{|V|}\}$ to fit points $\{y_1,\ldots,y_{|V|}\}$. Then*

$$\rho(\bar{X}^{t-1},X^t) = \rho(X^{t-1},X^t)\rho(\bar{X}^{t-2},X^{t-1}).$$

Proof: Denote the translation vector and transformation matrix for the Procrustes transformation of points $\{x_1,\ldots,x_{|V|}\}$ to fit points $\{y_1,\ldots,y_{|V|}\}$ as $b(Y,X)$ and $T(Y,X)$. Then by definition, $\rho(\bar{X}^{t-1},X^t)$ is the solution of

$$\text{minimize}_{\rho,\,T,\,b} \sum_{i=1}^{|V|} \|\bar{x}_i^{t-1} - (\rho T x_i^t + b)\|^2.$$

As $\bar{X}^{t-1}$ is itself a Procrustes transformation of X^{t-1}, it must be of the form

$$\bar{x}_i^{t-1} = \rho(\bar{X}^{t-2},X^{t-1})T_2 x_i^{t-1} + b_2$$

with T_2 an 2×2 orthogonal matrix and $b_2 \in R^2$. Substituting in the previous equation shows that $\rho(\bar{X}^{t-1},X^t)$ is the solution of

$$\text{minimize}_{\rho,\,T,\,b} \sum_{i=1}^{|V|} \|\rho(\bar{X}^{t-2},X^{t-1})T_2 x_i^{t-1} + b_2 - (\rho T x_i^t + b)\|^2.$$

Since the 2-norm is invariant under orthogonal transformation, the above is the same as

$$\text{minimize}_{\rho,\,T,\,b} \sum_{i=1}^{|V|} \|x_i^{t-1} - \Big(\frac{\rho}{\rho(\bar{X}^{t-2},X^{t-1})} T_2^T T x_i^t + T_2^T \frac{b-b_2}{\rho(\bar{X}^{t-2},X^{t-1})}\Big)\|^2.$$

Because here $T_2^T T$ represents any 2×2 orthogonal matrix for a suitably chosen T, this means that $\frac{\rho(\bar{X}^{t-1},X^t)}{\rho(\bar{X}^{t-2},X^{t-1})}$ must be a solution of

$$\text{minimize}_{\rho,\,T,\,b} \sum_{i=1}^{|V|} \|x_i^{t-1} - (\rho T x_i^t + b)\|^2,$$

in other words,

$$\frac{\rho(\bar{X}^{t-1}, X^t)}{\rho(\bar{X}^{t-2}, X^{t-1})} = \rho(X^{t-1}, X^t)$$

or

$$\rho(\bar{X}^{t-1}, X^t) = \rho(X^{t-1}, X^t)\rho(\bar{X}^{t-2}, X^{t-1}).$$

□

Applying this theorem recursively gives

$$\rho(\bar{X}^{t-1}, X^t) = \Pi_{i=2}^{t}\rho(X^{i-1}, X^i).$$

Hence any scaling difference between two previous layouts will have a multiplicative effect on the current layout. For example, if the first layout X^1 is 20% smaller than X^2, then a Procrustes transformation to any subsequent layout will cause a downscaling by a multiple of 0.8. Over time, for some cases the scaling can drift far from 1, causing problems such as severe label overlap. Since for a smooth visualization, it is necessary to preserve the mental map only among a few nearby frames (rather than over the entire sequence of frames), we took the approach of combining the transformed coordinates and the original coordinates with a parameter $0 < s < 1$, where $s = 1$ yields the transformed coordinates and $s = 0$ the original coordinates. In our implementation we use $s = 0.5$.

5.2 Evaluating layout stability

To evaluate the different layout stability approaches, we compare the trajectories of a set of randomly selected nodes from the dataset in Fig. 1. Fig. 4 (top) 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. Fig. 4 (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. 4 (bottom) 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.

Fig. 5 compares two sets of maps: one drawn without affine transformation (left) and one drawn with affine transformation. The maps were computed by interpolating between the mtDNA and NRY DNA maps from Fig. 1 with weights 0.3, 0.5, 0.7 and 0.9 for w as explained in Section 6.1. Note that the differences between the left and right maps in Fig. 5 grows with each iteration. That is, each successive pair of frames shows a greater difference than its predecessor. This is because the changes due to the affine transformation are cumulative since it is applied at every stage. Consequently, the final results in the sequences are not affine transformations of each other.

In all cases we experimented with, the approach where we use the Procrustes transformation results in the least "jittery" trajectories. We compared the performance of our algorithm using the average distance traveled by a node. We computed the position of a node in two maps and the Euclidean distance between the two points. We then average this over all nodes in the two maps. We repeated the experiment on 100 successive

pairs of maps obtained from the different weighted sum of the similarity matrices and then averaged these values over all successive pairs. We use this node-travel distance per map pair to quantify the different approaches. In the example where the similarity matrices are obtained from mitochondrial DNA and from Y chromosome, the distances traveled using independent layout, initialized positions and Procrustes transformation 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. Fig. 4 shows the trajectories of 9 randomly chosen nodes. We can see from the bottom figure that initializing with positions from the previous frame and using the Procrustes projection yields trajectories that are the shortest and the least jittery. We note that node-travel distance per map pair is computed over all nodes in the maps, not just the random sample shown in the figure.

6 Applications

In this section, we apply our dynamic map visualization approach to the problem that first motivated this research – understanding the effect to the clustering of a group of people when combining and mixing two different metrics based on NRY DNA or mtDNA similarity scores. Furthermore, we apply our methodology to a diverse group of datasets, including mobile phone call graphs, TV show similarities, as well as author collaboration graph data from the Symposium on Graph Drawing. These applications illustrate how the color stability algorithm can be applied to geographic data, as well as to relational data where in addition, our algorithms for layout stability and cluster stability are employed.

6.1 Population Genetics

One of the major problems in evolutionary biology is to better understand the genomic and evolutionary factors shaping patterns of human variation and to test models of human origins. Genetics is a powerful tool for uncovering the evolutionary history of our species. DNA is comprised of four building blocks: guanine (G), adenine (A), thymine (T), and cytosine (C). The sequence of the vast majority of these 3 billion building blocks (base pairs or nucleotides) is identical among humans. However, sprinkled throughout the genome are small differences in the DNA sequence known as polymorphisms. Most polymorphisms are simple substitutions of one base for another, called single nucleotide polymorphisms or SNPs. These kinds of polymorphisms are found at only about 1 out of 1000 nucleotide sites among humans. When many of these polymorphisms are compared among individuals from different human populations, they are informative for inferring the timing and order of branching events among ancestral populations, as well as the history of changes in population sizes. The signature of these historical processes, as well as natural selection, can be read from patterns of polymorphism in our genome.

DNA is found in nearly every cell of our body. Most cells have two compartments that contain DNA. Nuclear DNA is packaged in chromosomes found in the nucleus of the cell. Mitochondrial DNA (mtDNA) is a circular molecule found in the mitochon-

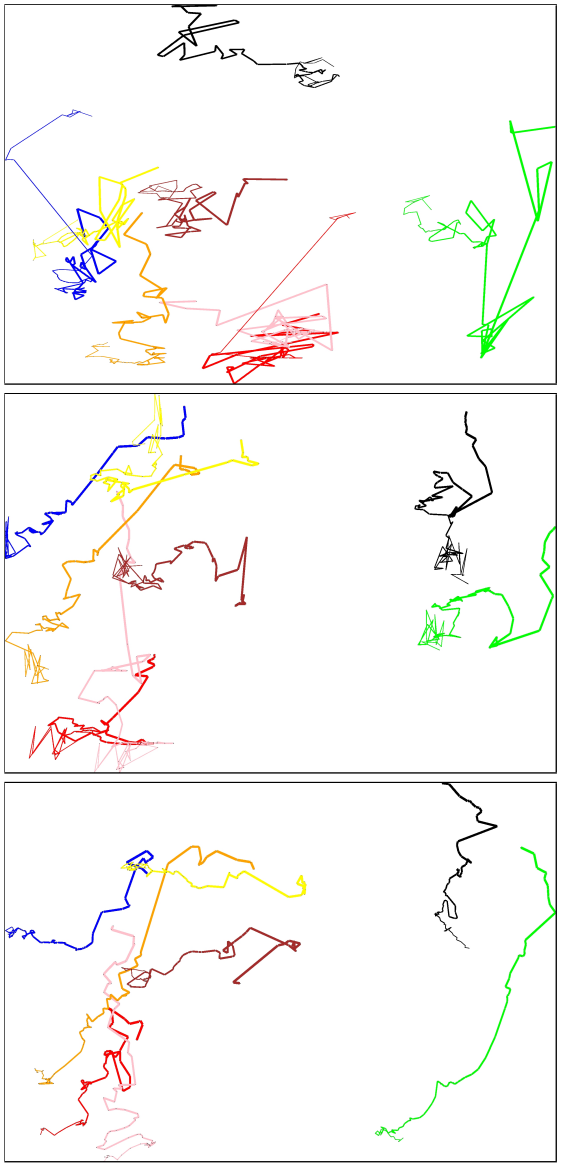


Figure 4: Trajectories of nine randomly selected nodes with three different layout stability methods. (Top) independent layout. (Middle) layout initialized with positions from the previous frame. (Bottom) initialized positions and Procrustes transformation. We see that overall, initializing with positions from the previous frame and using Procrustes transformation (bottom) give the shortest and least jittery trajectories.

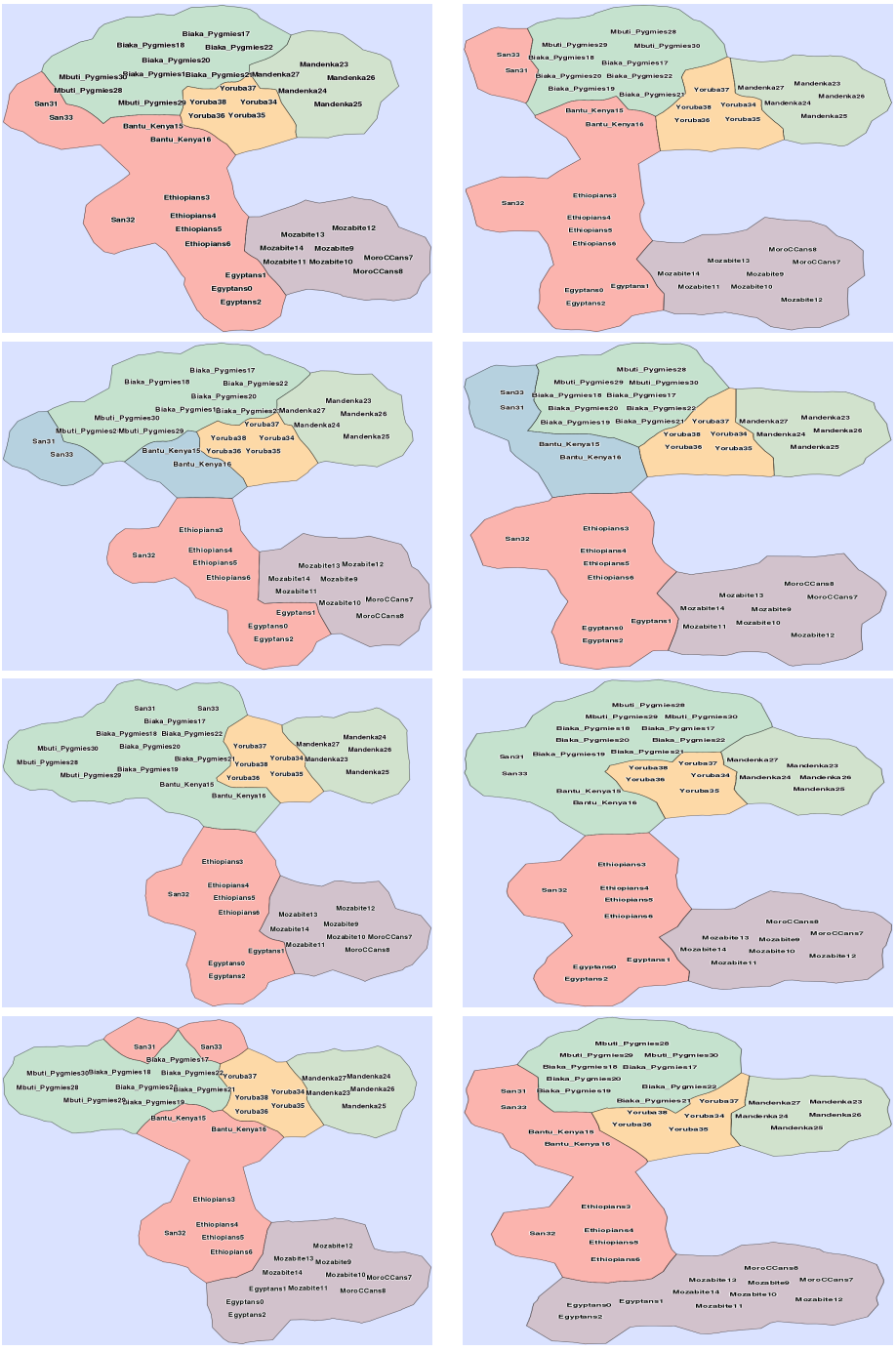


Figure 5: A sequence of maps obtained by combining mtDNA and NRY DNA similarity metrics. Note that in both sequences the clusters match well and the colors of the clusters also match well. The sequence in the left and right employ different node stabilization: (Left) sequence obtained without affine transformation; (Right) a sequence obtained with affine transformation. Note that although both this Figure and Fig. 1 are part of the same animation, this Figure is the result of quite different weight values from that of Fig. 1, thus the figures do not look similar.

dria in the cytoplasm of our cells. Humans have 23 pairs of chromosomes, with one chromosome of each pair inherited from our mother and the other from our father. The first 22 pairs of chromosomes are called the autosomes, while the 23rd pair is called the sex chromosomes, namely the X and Y chromosomes. Females are born with two X chromosomes while males are born with an X and a Y chromosome. The Y chromosome is inherited only from the fathers line, and because there is no shuffling of genetic material (known as recombination) between the Y and X chromosomes, most of the Y chromosome (the non-recombining portion of the Y chromosome or NRY) traces to only a single male in each previous generation. Similarly, mitochondrial DNA (mtDNA) is inherited only through females and does not undergo recombination, so our mtDNA traces back to a single mother in each previous generation.

When making inferences about human history from DNA sequence data, it is important to take account of processes that differentially affect the parts of the genome (NRY and mtDNA), the X chromosome, and autosomes. In addition to the different inheritance patterns among the four parts of the genome, there are differences in copy number, recombination rate, and mutation rate. While genes carried on the mtDNA or NRY are present in only a single copy in an individual, X-linked genes are present in two copies in females and only one copy in males (but can be transmitted by either sex). Autosomal genes are maintained on two chromosomes in both sexes. As a result, when equal numbers of males and females are breeding, the relative effective population size of the autosomes, X chromosome, NRY, and mtDNA is 4:3:1:1, respectively.

Until recently, questions about human prehistory have primarily been addressed with polymorphism data from mtDNA and the NRY. However, these parts of the genome make up only a small fraction of our total DNA. While the mtDNA and NRY each trace back to only a single ancestor each generation, the hundreds of thousands of independent segments comprising the remainder of our genome trace to many different ancestors in the past. While the patterns of genetic inheritance dictate that there is a single ancestor (and a single phylogenetic tree) for all non-recombining portions of our genome, each portion of our genome will not necessarily have the same history. In other words, a gene tree can tell us about the history of a single locus, but not about the structure of the whole population or about the forces that determine the spread of alleles at that locus. Due to chance effects and selection, a single gene tree may not accurately reflect the phylogenetic relationships of the populations studied. Since neither selection nor chance effects is expected to influence all loci equally, gene trees from independently segregating loci must be compared to test hypotheses about population history. It is only through an examination of many independent regions that we are in a stronger position to reconstruct the evolutionary history of our species. However, it is difficult to make inferences from patterns of variation in different parts of the genome. Currently, there is not a single methodology that allows inference across these systems. Using the dynamic map representation would allow us to put together these different relationships defined on the same set of objects, while allowing the interactive exploration of the different ways to combine them.

When comparing DNA information from a group of people, the NRY DNA similarity score and the mtDNA similarity score are computed from different parts of the DNA (paternal information and maternal information). Such scores can give an indication about the strength of the genetic similarity between the individuals. Combining

these scores using a linear interpolation can give a better indication of the genetic similarity. With this in mind we have implemented an interactive system which allows for the exploration and visualization of the effect of linear interpolation between two similarity metrics (represented by a pair of adjacency matrices). We obtain smooth transitions from one matrix to another, and which the viewer can manipulate or just watch. Let w be a weight in the range from 0 to 1 and let M_1 and M_2 be the two input matrices. In the example in Figure 5 the matrix M_1 corresponds to the NRY DNA similarity score and M_2 corresponds to the mtDNA similarity score. We compute the weighted matrix $M_w = w \times M_1 + (1-w) \times M_2$. The weighted sum M_w carries a w fraction of the paternal information and a $(1-w)$ fraction of the maternal information. The weighted matrix is then passed on to the module that computes the dynamic map clustering, then to the module that computes the dynamic map coloring, and finally to the module that computes the dynamic map layout.

Figure 5 shows a sequence of frames of the system when the weight w is changed from 0 to 1. The sequence in the left and the right employ different node stabilization. The top pair of maps are obtained by assigning a weight $w = 0$, i.e., they show only the maternal information. The bottom pair of maps are obtained by assigning a weight $w = 1$, i.e., they show only the paternal information. The intermediate maps are obtained by changing the value of w from 0 to 1 which shows the effect of linear interpolation between the two similarity measures. For the sequence of maps on the right, the two nodes *San*31 and *San*33 on the north west corner of the top map remain in the northwest in all the maps below, which shows the effect affine transformation in layout stability. Even though the difference between the layout on the left and the layout on the right is only the affine transformation which involves translation, rotation and scaling, the two layouts look quite different. This is because we initialize the position of each node based on the previous layout.

The embedding of the graph was done using `Graphviz` [17]. In particular, we used the `neato` layout engine of `Graphviz`. In addition to `neato`, the `gvpr` scripting language is used for manipulating graphs. Finally, GMap [27], available as part of `Graphviz` provides most of the geographic map functionality. Movies of the system in action and a collection of images can be found at `http://cs.arizona.edu/people/sankar/dynamap`.

6.2 Maps of the Connected States of America

Calabrese *et al.* [9] studied the problem of partitioning the land area of the United States of America based on how people interact, rather than based on states and counties. The resulting maps, which they call "maps of the Connected States of America", illustrate the emerging communities based on the social interactions manifested in anonymized mobile phone data, as well as in census data. The idea is to cluster regions based on various types of interaction (e.g., phone calls). By visually examining the results one can see whether the communities defined purely by social interactions coincide with the administrative boundaries (e.g., state boundaries). Studying the communities emerging from such social and physical interaction allows us to see mobility of the population.

Different interaction data sets result in different ways to cluster the counties in the US, and hence in different maps. In the paper by Calabrese *et al.*, the colors of

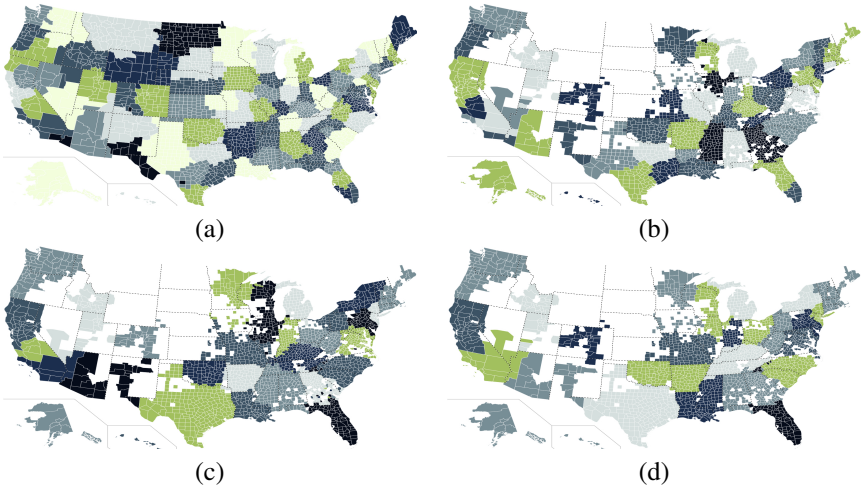


Figure 6: Original maps from Calabrese *et al.* [9]. Map of the Connected States of America based on (a): commuting, (b): migration, (c): voice calls and (d): SMS messages. The colors in each of these maps are assigned independently and thus there is no pairwise correspondence.

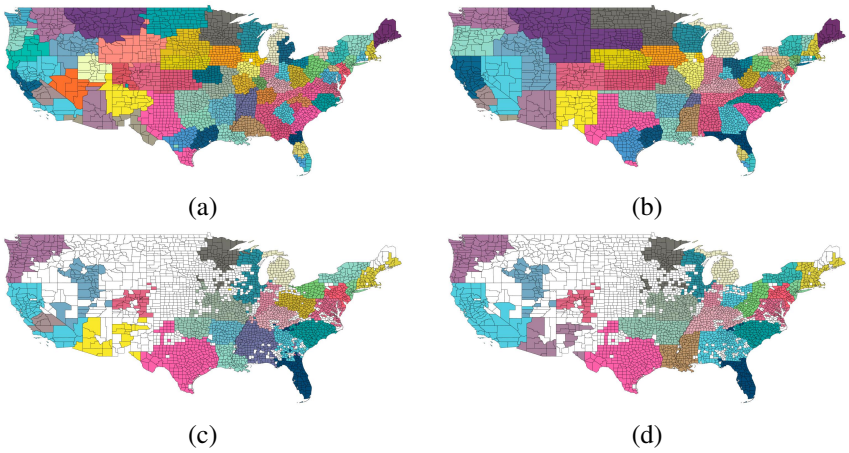


Figure 7: From top to bottom: Map of the Connected States of America based on (a): commuting, (b): migration, (c): voice calls and (d): SMS messages. The coloring of the bottom three maps are based on matching with that of the commuting map (a), making it easier to see similarities and differences.

the regions in each map is done independently; see Fig. 6. This makes it difficult to visually compare the resulting maps. We have redrawn four of these maps and applied the color stabilization algorithm, making it easier to see similarities and differences in the maps; see Fig. 7.

Specifically, we use four data sets from Calabrese *et al.* [9]. (Two of these sets, the voice and SMS clustering, are available at `http://senseable.mit.edu/csa/downloads.html`) In each data set, counties in the USA are assigned to clusters based on aggregated interactions among them. In particular, a graph of counties of the US was first created. An edge exists between two counties if there were interactions between them, with the edge weight proportional to the amount of interaction. A modularity clustering algorithm was then applied to group connected counties together. Four different graphs were derived based on different types of interactions.

The first one is a commuting graph created from the 2000 US Census, based on county-to-county worker flow. Here nodes are counties and an edge exists between two counties i and j if there are workers living in county i and working in county j. The edge weight equals the number of such workers.

The second one is a migration graph created from 1995-2000 US Census, based on migration flow. Here the edge weight between county i and j is proportional to the number of people who migrated from i to j.

The third one is a voice call graph created from phone records, with edge weights determined by the call volume between corresponding pairs of counties.

The fourth one is an SMS graph created from SMS records, with edge weights determined by the number of SMS messages. Note that the voice and SMS graphs contain incomplete data, reflecting the service regions of the cell phone operator.

To create the map colors we begin from the commuting map (the base map) where we color counties in the same cluster with the same color. We color the other three maps (migration, voice, and SMS) so as to optimize the color match with the base map. We use the maximum bipartite matching coloring algorithm described in Section 4. This has the desired effect as clusters in the new maps are assigned colors so that the correspondence between clusters in map pairs is easy to see. Here we maximize the total number of matches counties. An alternative metric (that also makes sense in this geographic setting) would be to maximize the total matched areas.

The results in Fig. 7 show that our algorithm is successful in providing a consistent and stable coloring. While it is difficult to compare among the maps in Fig. 6, the color matching in our maps improves readability. For example, the good color stability makes it easy to see that the voice and SMS maps (bottom two) are very similar. Small differences stand out better and we can spot that Alabama and Mississippi are grouped in the voice map while Louisiana and Mississippi are grouped in the SMS map. Similarly, when comparing the commuting and migration maps (top two figures), it is easy to see that the commuting clustering is more localized, which suggests that people tend to live within a "commuting distance" to work, but are often willing to relocate further away. In the commuting map, California is divided into five regions, while in the migration map, the state is largely made of three parts, North, South, and Southeast.

We note that Fig. 6 (b) is much sparser than Fig. 7 (b). The maps in Fig. 6 come from the the original paper by Calabrese *et al.*. The data we used for Fig. 7 is from the same authors, but is likely not identical to the data they used for Fig. 6.

6.3 The Evolving TV Landscape

Many recommender systems rely on knowledge about how items are related to each other through a similarity measure. Consider TV recommendations as an example: two TV shows can be considered similar if people who watch one also tend to watch the other. A visualization of this similarity relation offers an intuitive ways for the user to understand the landscape of the collection of items (TV shows in this case). In such a visualization, shows that are highly similar are placed close to each other. Discovery can then be made by inspecting this landscape and finding shows that are closest to those that the user has already watched. Over time, however, new shows will come up and old shows go off season. In this section we apply our algorithm to the problem of visualizing the evolving TV landscape over time.

The dataset we use comes from a digital TV service with millions of set-top boxes, from which we compute the show-show similarity matrix for the TV graph. All the data was collected in accordance with appropriate end user agreements and privacy policies. The analysis was done with data that was aggregated and fully anonymized. No personally identifiable information was collected in connection with this research. Here nodes of the graph are TV shows and the edges are computed based on viewership information using a factorization model [28]. The full similarity matrix is dense and very large (tens of thousands of rows and columns and millions of non-zero values).

For the maps in this paper we consider only the top 100 most popular shows. For each show, we take the top 10 most similar shows, which gives us a sparse graph, with the weight of each edge proportional to the similarity, and length inversely proportional to the edge weight. We build up a total of 42 such graphs for each of the 30 day time periods from February 1, 2012 to April 21: February 1 to March 2, February 2 to March 3, ..., March 22 to April 21; see Fig. 8. Note that since the graphs are often not fully connected, we display only the largest connected component for each graph.

We begin by computing a layout for the first graph, then the second, third, etc. To compute the layout for the $(t+1)$-th graph G^{t+1} in the series, we identify nodes that were also present in the previous graph G^t. For each such surviving node v, we add a duplicated dummy node v' to G^{t+1} that is pinned to the position of v in the layout of G^t, and add the edge (v,v') with ideal edge length 0. For each node u in G^{t+1} which is not in G^t, we identify a surviving node w which is closest to u in G^{t+1}, and initialize the position of u with the position of w. We then compute a layout for graph G^{t+1} which has been augmented with the dummy nodes and edges with the following stress model:

$$\min_x \sum_{(i,j)\in E} w_{ij}(\|x_i - x_j\| - d_{ij})^2 + \sum_{(i,i')\in E'} w_{ii'}\|x_i - x_{i'}\|^2 \tag{6}$$

where E is the set of original edges of G^{t+1}, and E' the set of dummy edges. The dummy nodes and edges have the effect of keeping existing nodes near their positions in the previous layout (of G^t), while at the same time allowing some flexibility in their movement, provided that the overall stress is minimized. A short animation of the 42 graphs in the TV Landscape sequence shows that this approach leads to a reasonably stable layout: `http://cs.arizona.edu/people/sankar/dynamap/TV.gif`.

Once the layout is calculated, we compute clusters in the graph using the GMap [27] algorithm to highlight groups of similar TV shows. The traditional GMap algorithm computes grouping via modularity-based clustering [34, 35]. For the evolving TV landscape we compute the clustering of G^{t+1} based on the clustering of G^t. Since we are computing the clustering of G^{t+1} based on the clustering of the G^t our algorithm expects each node to have an old cluster identity. So each node which is in G^{t+1} but not in G^t is initially assigned to a singleton cluster, then the clustering algorithm of Section 3 is applied. The colors are assigned using the coloring algorithm of Section 4.

If we compare the middle and bottom subfigures of Fig. 8, we can observe that the a new cluster was formed with shows "Real Housewives of Atlanta", "Real Housewives of Orange County" and "Khole and Lamar". This we think was because of the season premiere of "Khole and lamar" (on 02/19/2012 and 02/20/2012) and new season episodes in the "Real Housewives" franchise (on 02/19/2012 and on 02/21/2012) .

6.4 The Growth of a Research Area

We conclude this section with another example using a different type of data, to further demonstrate that our methodology is flexible and can be applied to domains other than the motivating DNA similarity analysis. We consider the growing co-authorship graph of researchers who published papers in the Annual Symposium on Graph Drawing. Here each node is an author and an edge exist between two authors if they co-authored a paper for the Symposium. The edge weight is the number of co-authored papers, and our data set covers years 1994 to 2004, in a cumulative fashion: we have one graph for each of the periods 1994–1995, 1994–1996, ..., 1994–2004. To achieve mental map stability, we compute the layout of the 1994-2004 map (the base map) first, then create dummy edges linking nodes in other maps to the corresponding dummy nodes in the base map. A layout of this graph gives us an initial position for all nodes. We then apply our layout stability, cluster stability, and color stability strategies described earlier, and obtain maps that allow us to study the evolution of collaborations and clusters over time.

A short animation showing the growth of the community of authors in the graph drawing area over the entire period can be found in our webpage. `http://cs.arizona.edu/people/sankar/dynamap/gd.gif`. Fig. 9 shows three snapshots of the largest connected component of the collaboration graphs for the period ending 2001, 2002, and 2003. One observation is that as time progresses, there are more and more authors, but fewer clusters. While the first map contains five clusters with German authors(most of the southwest and south of the map), the last map contains only three such clusters, but of larger size.

7 Conclusion and Future Work

In this paper we presented a practical approach for visualizing dynamic relational data and multiple relationships on the same data, using a geographic map metaphor. We proposed a heuristic to address the challenge of dynamic cluster stability, as well as a strategy to preserve color stability, and analyzed the use of affine transformations

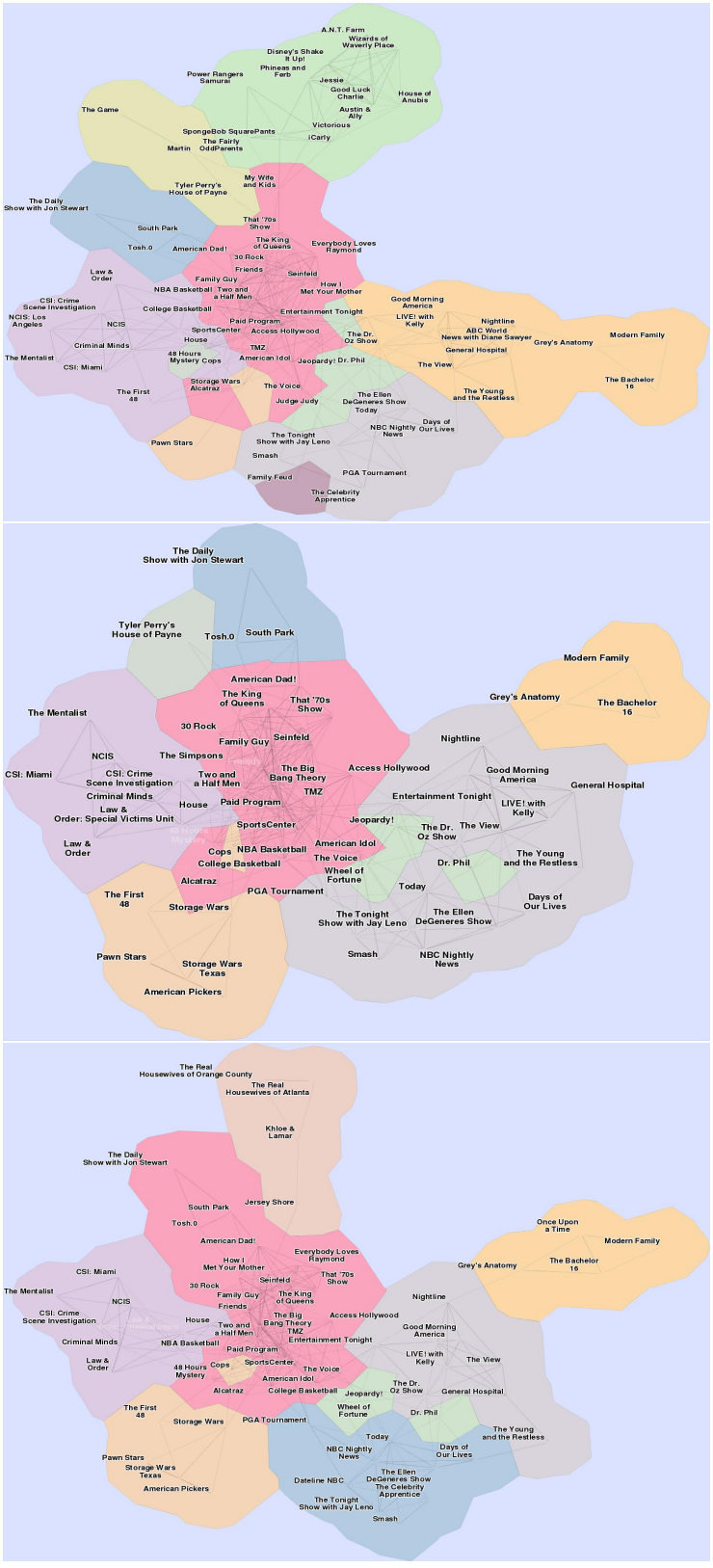


Figure 8: Evolving TV Landscape: three consecutive 30 day time periods beginning Feb 22, 23 and 24 of 2012.

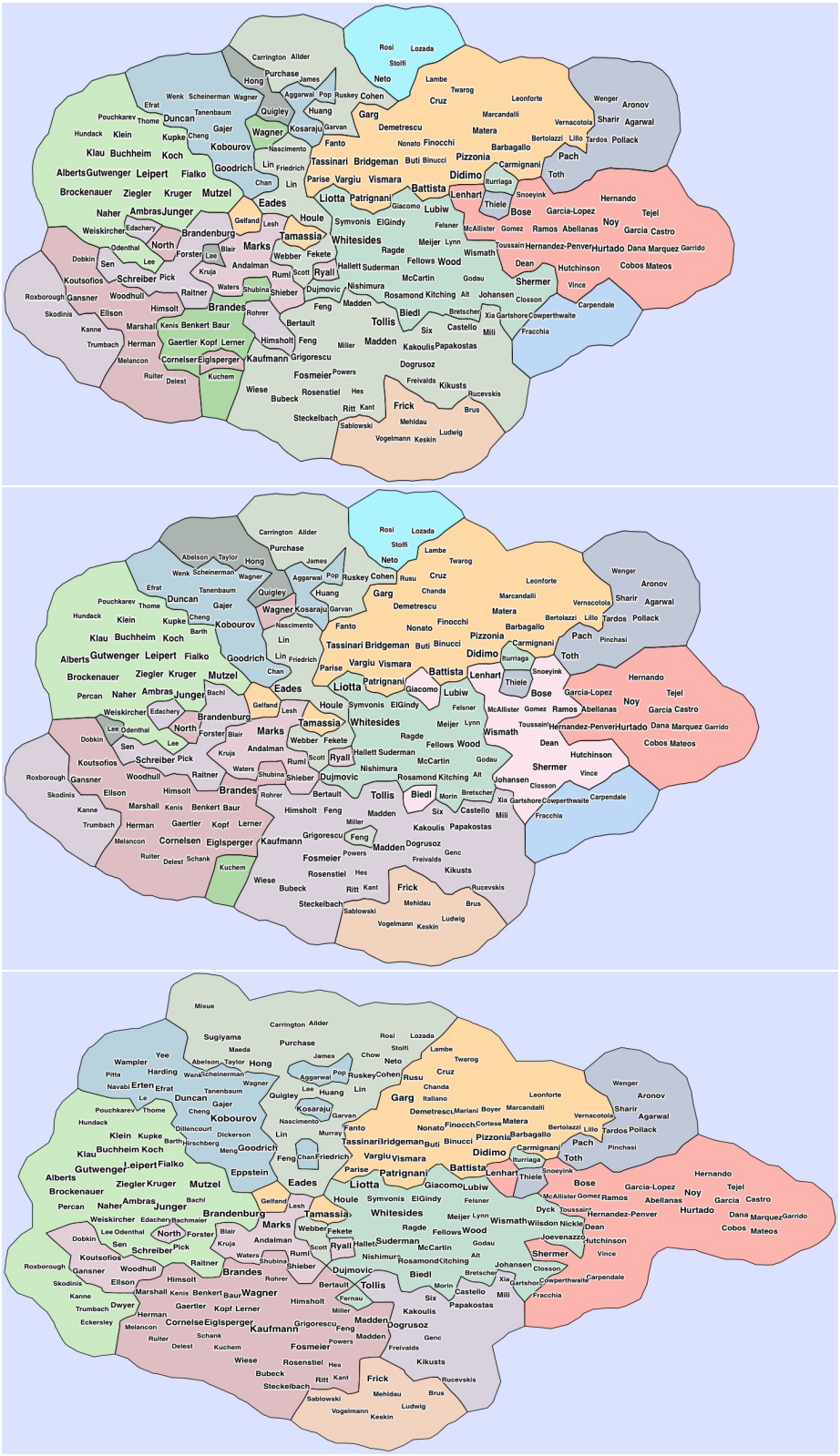


Figure 9: A sequence of GD collaboration graphs ending 2002, 2003 and 2004.

to improve layout stability. We applied these ideas in an interactive tool for DNA visualization.

We modeled the color stability as a maximum weighted matching of a bipartite graph. There are alternative ways to model the cluster stability problem that deserve further investigation. The stable marriage problem could be used to match clusters in C_{old} and C_{new}. Each cluster i in C_{old} has a preference list $\{j_1, j_2, j_3 \dots j_n\}$, of clusters in C_{new}. The cluster j_k is preferred over j_l in the list of i if and only if $w(i, j_k) \geq w(i, j_l)$. The Gale-Shapley algorithm for stable marriage runs in $O(m)$ time [22], however, the algorithm is not symmetric. That is, the solution obtained with respect to C_{old} is not necessarily the same as that obtained with respect to C_{new}. If the two solutions are indeed different, we could compute both and take the better of the two (in terms of numbers of matches).

It is also desirable for the majority of the elements in each cluster to remain in the same cluster. As this property cannot be guaranteed for all clusters, we can instead maximize the number of clusters for which this property holds. Let k be the number of clusters in C_{old}. The objective function to optimize is $Z = \sum_{0 \leq i \leq k} z_i$, where $z_i = 1$ if the majority of the elements in cluster i in the old clustering remains in cluster i in the new clustering, and 0 otherwise. Z may be optimized using a simple greedy algorithm.

We would also like to be able to combine more than two similarity metrics, to allow for more general ways to combine multiple similarity metrics.

We were able to use our system in a generic dynamic data setting. The TV data set and the research collaboration data set help illustrate the utility of cluster, color and layout stability, when dealing with dynamic data. Parts of the system can also be used for comparing and contrasting geographic maps, as illustrated with the world trade maps and with the "connected states" maps, computed based on commuting and communication patterns. We hope that a formal evaluation of the effectiveness of our method would allow us to weigh more carefully the balance between cluster stability, color stability, and layout stability.

Acknowledgments

We thank Michael Hammer and Nirav Merchant from the Bio5 Institute for introducing us to the DNA-similarity problem and for their patience in explaining the various similarity metrics and the underlying biology.

We thank DeDe Paul from AT&T Labs and Francesco Calabrese from IBM Research Dublin for assistance with data for the maps of the Connected States of America, and for allowing us to use their images in Fig. 6.

References

[1] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Stat. Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[2] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. In *Proc. of the IEEE INFOVIS'02*, pages 145–151, 2002.

[3] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *Proc. of the WG'07*, pages 121–132, 2007.

[4] U. Brandes and M. Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In M. J. van Kreveld and B. Speckmann, editors, *Graph Drawing*, volume 7034 of *Lecture Notes in Computer Science*, pages 99–110. Springer, 2011.

[5] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In *Proc. of the 5th Symp. on Graph Drawing (GD)*, pages 236–247, 1998.

[6] C. Brewer. ColorBrewer - Color Advice for Maps. `www.colorbrewer2.org`.

[7] M. Burch and S. Diehl. Timeradartrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum*, 27(3):823–830, 2008.

[8] H. Byelas and A. Telea. Visualization of areas of interest in software architecture diagrams. In *Proc. of the ACM Symp. on Software Visualization (SoftVis'06)*, pages 105–114, 2006.

[9] F. Calabrese, D. Dahlem, A. Gerber, D. Paul, X. Chen, J. Rowland, C. Rath, and C. Ratti. The connected states of america: Quantifying social radii of influence. In *Proc. of the SocialCom/PASSAT*, pages 223–230. IEEE, 2011.

[10] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 554–560, New York, NY, USA, 2006. ACM.

[11] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE TVCG*, 15(6):1009–1016, 2009.

[12] P. F. Cortese, G. D. Battista, A. Moneta, M. Patrignani, and M. Pizzonia. Topographic visualization of prefix propagation in the internet. *IEEE TVCG*, 12:725–732, 2006.

[13] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000.

[14] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu. Context-preserving, dynamic word cloud visualization. *Computer Graphics and Applications*, 30:42–53, 2010.

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

[16] S. Diehl and C. Görg. Graphs, they are changing. In *Proc. of the 10th Symp. on Graph Drawing (GD)*, pages 23–30, 2002.

[17] J. Ellson, E. Gansner, E. Koutsofios, S. North, and G. Woodhull. Graphviz and dynagraph–static and dynamic graph drawing tools. *Graph Drawing Software*, pages 127–148, 2003.

[18] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. V. Yee. GraphAEL: Graph Animations with Evolving Layouts. In *Proc. of the 11th Symp. on Graph Drawing*, pages 98–110, 2003.

[19] S. I. Fabrikant, D. R. Montello, and D. M. Mark. The distance-similarity metaphor in region-display spatializations. *IEEE Computer Graphics & Application*, 26:34–44, 2006.

[20] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, July 1987.

[21] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18:1013–1036, October 1989.

[22] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[23] R. Görke, P. Maillard, C. Staudt, and D. Wagner. Modularity-driven clustering of dynamic graphs. In *Proc. of the 9th Symp. on Experimental Algorithms*, pages 436–448, 2010.

[24] D. Greene and P. Cunningham. Spectral co-clustering for dynamic bipartite graphs. In *Workshop on Dynamic Networks and Knowledge Discovery (DyNAK'10)*, 2010.

[25] Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[26] D. Holten and J. J. van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum*, 27(3):759–766, 2008.

[27] Y. Hu, E. Gansner, and S. Kobourov. Visualizing Graphs and Clusters as Maps. *IEEE Computer Graphics and Applications*, 99(1):54–66, 2010.

[28] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 263–272, 2008.

[29] J. Novembre et al. Genes mirror geography within Europe. *Nature*, 456(7218):98–101, 2008.

[30] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[31] D. Mashima, S. G. Kobourov, and Y. F. Hu. Visualizing dynamic data with maps. In *Proc. of the IEEE Pacific Visualization Symposium*, pages 155–162, 2011.

[32] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Vis. Lang. Comput.*, 6(2):183–210, 1995.

[33] N. Yang et al. Examination of ancestry and ethnic affiliation using highly informative diallelic DNA markers: application to diverse and admixed populations and implications for clinical epidemiology and forensic medicine. *Human genetics*, 118(3):382–392, 2005.

[34] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004.

[35] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, Feb 2004.

[36] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. of the Amer. Stat. Assoc.*, pages 846–850, 1971.

[37] P. Simonetto, D. Auber, and D. Archambault. Fully automatic visualisation of overlapping sets. *Computer Graphics Forum*, 28:967–974, 2009.