Embedding, Clustering and Coloring for Dynamic Maps

Category: Research

Abstract

We describe a practical approach for visualizing dynamic relational data, or 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.

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

1 Introduction

Maps offer a familiar way to present geographic data (continents, countries, states). Contact graphs, 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 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. 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 visualization for dynamic data and for multiple relationships on the same dataset.

Our motivation comes from population genetics, where one of the major problems is to better understand the genomic and evolutionary factors shaping patterns of human variation and to test 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, using multiple relationships on the same set of objects. Interactively combining these multiple relationships, in an intuitive and visually appealing fashion, will facilitate knowledge discovery and hypothesis testing.

In this paper, we describe a practical approach for visualizing dynamic relational data and multiple relationships on the same data set with the help of the map metaphor. Our main motivation is the biological problem of assigning ancestry, based on several DNA similarity metrics: mitochondrial similarity, autosomal similarity, and NRY (non-recombining portion of the Y chromosome) similarity.[1]

More precisely, 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 suitable placement of nodes and edges, but also with dynamic clustering and dynamic color assignment (for readability and mental map preservation); see Fig. 1.

The main contributions of the paper are the following: first, we describe a heuristic to promote dynamic *cluster stability*; second, we propose an optimal color assignment algorithm to maximize *color stability* between maps; third, we analyze the use of affine transformation to improve *layout stability*. Finally, we briefly describe an interactive tool for DNA visualization that utilizes these results.

2 Related Work

Our work builds on the GMap algorithm [22], 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 [15]. Cortese *et al.* [9] 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 [7] use deformed convex hulls to highlight areas of interest in UML diagrams. Collins *et al.* [8] use "bubblesets," based on isocontours, to depict multiple relations among a set of objects. Simonetto *et al.* [30] 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` [25], where mental map is preserved by fixing node positions. A context-preserving dynamic text data visualization by Cui *et al.* [11], 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).

[1] When DNA is passed from one generation to the next, most of it is mixed by the processes that make each person unique from his or her parents. Some special pieces of DNA, however, remain virtually unaltered as they pass from parent to child. One of these pieces is carried by the Y chromosome, which is passed only from father to son. Another piece, mitochondrial DNA (mtDNA), is passed (with few exceptions) only from mother to child. Since the DNA in the Y chromosome does not mix with other DNA, it is like a genetic surname that allows men to trace their paternal lineages. Similarly, mtDNA allows both men and women to trace their maternal lineages.

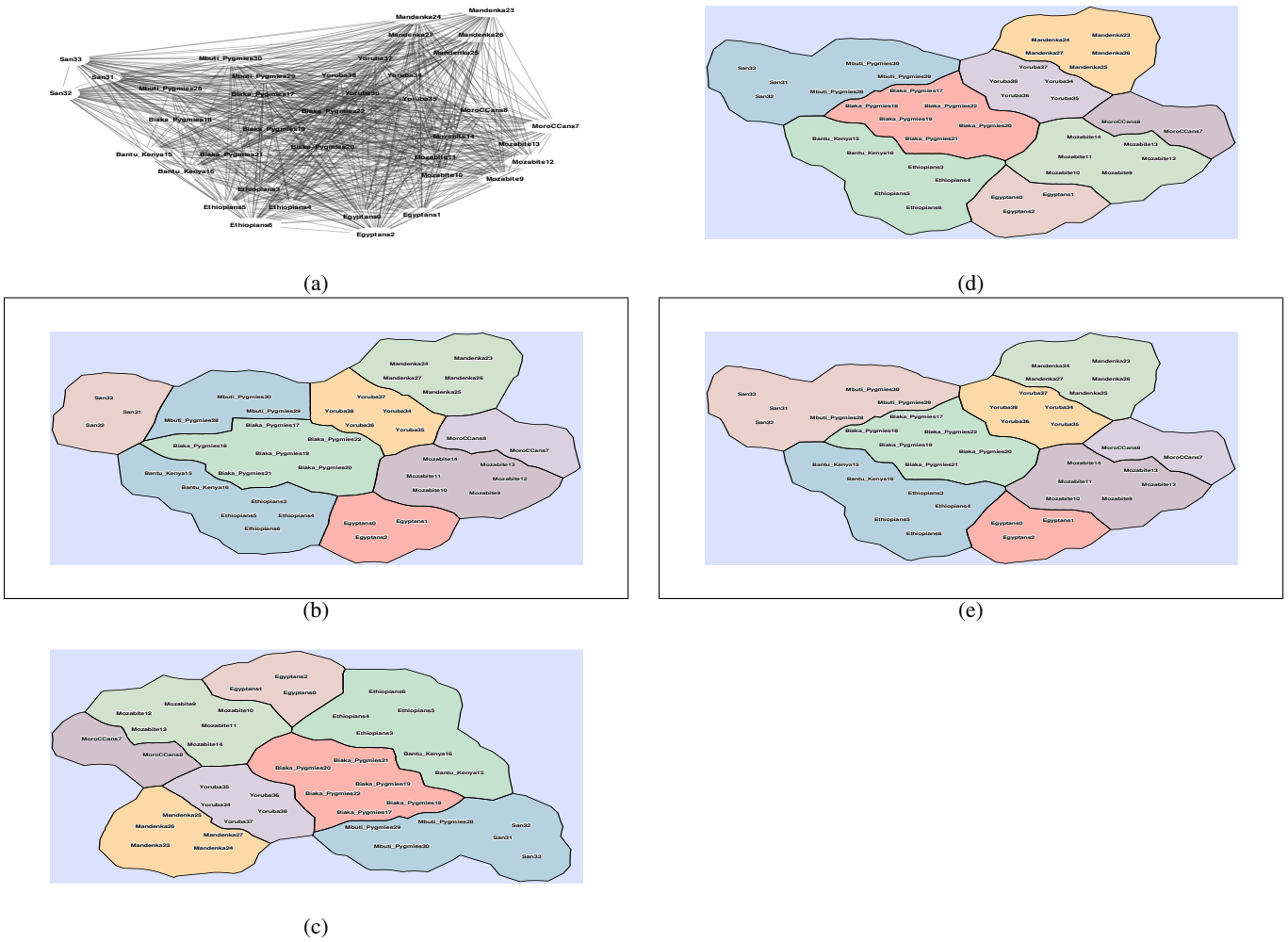

Figure 1: Visualization with dynamic maps requires both layout and color stability. (a): mitochondrial DNA similarity visualized as a graph. (b): mitochondrial DNA similarity visualized as a map. (c): NRY DNA similarity map, layout done independently, making it difficult to compare with (b). (d): NRY DNA similarity map, with layout done to optimize layout stability with regard to (b), which makes it easy to compare nodes, but still difficult to compare clusters. (e): NRY DNA similarity map with optimal layout and color assignment. Now it is easy to compare to (b). For example, we can see that two clusters in the top left of (b) are now merged in a single country.

Holten and van Wijk [21] consider the problem of visual comparison of hierarchically organized data, and propose a method that highlight the differences. Burch and Diehl [6] propose the TimeRadarTree 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 [4], while Herman *et al.* [20] rely on good static layouts. Diehl and Görg [12] 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 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 may not be as important when visualizing the evolution of citation or Internet graphs, but are undesirable when analyzing DNA similarity. Therefore, for example, we could not use techniques for dynamic graph layout, such as anchoring, or artificial edges linking the same node in different time frame [14].

In the area of visualization of DNA data Yang *et al.* [26] study ancestry and ethnic affiliation using DNA genetic markers and make accurate predictions with the help of a Bayesian clustering algorithm. Novembre *et al.* [23] show that there is a very high correspondence between genetic and geographic distance when they characterized the genetic variation in a sample of 3000 European Individuals.

3 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 autosomal), 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 Figure 1 show the nature of the problem. In Figure 1 (a), 39 subjects are embedded in 2D space based on a Mitochondrial DNA similarity measure, using multi-dimensional scaling. Figure 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 [5]. In Figure 1 (c), we re-embedded the subjects using a different similarity metric (NRY DNA similarity), independent of Figure 1 (b). Compared with Figure 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 Figure 1 (b) and Figure 1 (c). In Figure 1 (d), the embedding of NRY DNA similarity is done to minimize the difference to that of the embedding based on Mitochondrial DNA 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 Figure 1 (b). Finally, in Figure 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 amount of weight given to each of the two metric 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 Figure 1, namely: *node 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).

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, Figure 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. Figure 2 (a) is clustered using the total import/export values among countries as edge weights. Figure 2 (b) is clustered using the total import/export fractions[2] among countries as edge weights. 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. Interestingly, the idea to color a map of Europe and Africa so as to indicate European colonies was used over a hundred years ago in a poster for the German Shipping Company Woermann-Linie; see Fig. 3.

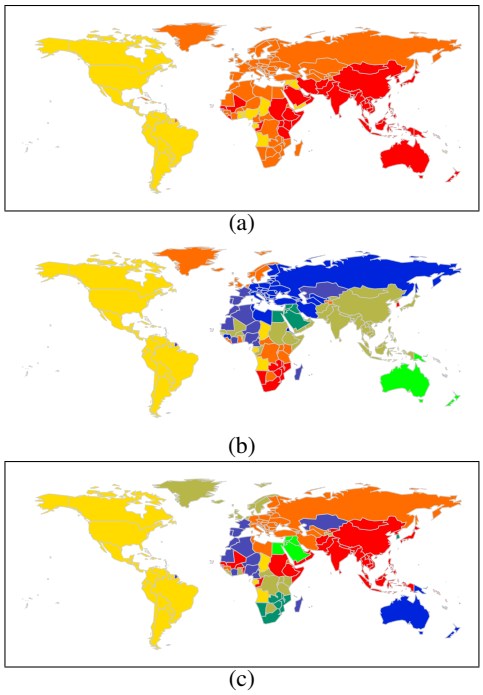

Figure 2: World trade maps: (a) clustering of countries based on total import/export values between countries; (b) clustering of countries based on total import/export fractions; (c) same clustering as (b), but with color stability.

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

4 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.

4.1 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, clustering algorithms are either divisive or agglomerative. Agglomerative algorithm 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

[2]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.

Figure 3: A map of Europe and Africa from a poster for the Woermann-Linie. Note that the colors of countries in Europe match the colors of the colonies in Africa: Mozambique and Portugal, Germany and Namibia, France and Senegal, Belgium and Congo, etc.

is modularity, defined as a value between -1 and 1 that measures the density of links inside communities compared to links between communities [27, 28]. For a weighted undirected graph $G = (V, E)$ the modularity is defined as follows:

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

Here $A_{i,j}$ is the weight of the edge between i and j, where $A_{i,j}$ represents the similarity between i and j (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 $\delta-$ 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.

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] (and made it available as the `Graphviz` function `cluster`).

4.2 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 do not result in suboptimal clustering.

One measure of cluster similarity given by Rand [29] is defined on the basis of node-pair clustering. Let C and C' be two clusterings of a graph G. Let S_{11} denote the set of pairs that are clustered together in both clusterings, and S_{00} denotes the set of pairs that are in different clusters. Then the Rand distance measure is given by

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

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. While this is a natural metric, we are not aware of a natural way to combine the modularity measure of a clustering and structural Rand distance to the previous clustering. Instead, we propose a simple heuristic to promote cluster stability.

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 [19]. Our implementation uses several agglomerative iterations, where clustering of the previous iteration is used for pre-clustering. We begin with each node as a singleton. We consider merging only node pairs which belong to the same cluster in the pre-clustering. 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 the adjacencies and edge weights adjusted appropriately. We proceed iteratively with the contracted graph as input.

5 Dynamic Map Coloring

Color stability, that is, using the same color for countries on the two maps that shares 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 $d(C_{old}, C_{new})$ be the number of nodes that undergo clustering change.

$$d(C_{old}, C_{new}) = \sum_{u \in V} \delta(C_{old}(u), C_{new}(u)); \quad (3)$$

$\delta(u, v) = 1$ 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 $d(C_{old}, \Pi(C_{new}))$. For example let $C_{old} = \{1, 1, 2, 2, 3\}$ be the clusters assigned to the 5 nodes v_1, v_2, v_3, v_4, v_5; let $C_{new} = \{2, 2, 1, 3, 4\}$ be the new clustering in which v_3 and v_4 split into two clusters. Clearly $d(C_{old}, C_{new}) = 0$. 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 $d(C_{old}, \Pi(C_{new})) = 4$.

The problem can be modeled with maximum weighted matching (MWM) of a bipartite graph. The corresponding bipartite graph G_C has node set $\{1, 2, \ldots, |C_{old}|\} \times \{1, 2, \ldots, |C_{new}|\}$. 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}.

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

$\phi(i, j, u) = 1$ 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 $d(C_{old}, \Pi(C_{new}))$. The MWM for bipartite graphs can be found using the Hungarian algorithm [24]. For bipartite graphs, an efficient implementation of the Hungarian algorithm using Fibonacci heaps [16] 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 [17] for MWM can be implemented in $O(n^{\frac{3}{2}} \log(n|V|))$. In practice, the number of clusters is typically small and our implementation of the Hungarian algorithms is fast enough.

6 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 [14]. 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.

We decided to preserve the mental map via a Procrustes transformation of the coordinates of the nodes, after an MDS layout, initialized with the node positions from the previous iteration. The Procrustes transformation is a well known technique [10], 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. Figure 4 shows the pipeline of how the transformation is applied to a sequence of graphs.

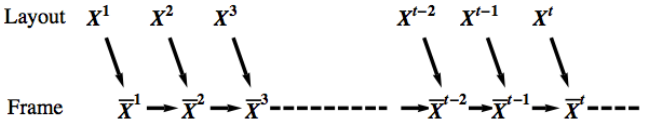

Figure 4: The affine transformation pipeline. X^t denotes the layout at time t. This layout is transformed to fit the previous frame $\bar{X}^{t-1}$ as close as possible to give the current frame $\bar{X}^t$, and the frames are what the user sees.

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 and affine 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 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

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

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

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

□

The same theorem applies to affine scaling. 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 the second 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 $s = 0.5$.

6.1 Evaluating layout stability

In order to analyze the effect of different layout stability approaches, we compare the trajectories of a set of randomly selected nodes in the example from Fig. 1. Fig. 5 (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. 5 (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. 5 (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.

We consider a sequence of frames using the last two strategies, as shown in Fig. 6. Note that the difference between the left and right sequences in Fig. 6 increases with each iteration, i.e., each successive pair of frames shows a greater difference than its predecessor. This is because the changes due to the affine transformations 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 last strategy was the best one, its trajectories being 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), and 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.

6.2 An additional example

Figure 7 shows the landscape of authors that submit to the International Symposium on Graph Drawing (GD) in each of the 3 years from 1999. Nodes are authors and two authors are connected by an edge if they co-authored a paper. The edge weight is the number of co-authored papers. MDS is used to place the nodes in the graph, and GMap is used to highlight clusters based on modularity clustering. By applying the layout, cluster and color stability strategies proposed in this paper, we can obtains maps that allow us to easily compare author clusters evolution over time.

7 The Dynamic Maps System

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 100 and let M_1 and M_2 be the two input matrices. We compute the weighted matrix $M_w = \frac{(w \times M_1) + ((100-w) \times M_2)}{100}$. 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: Trajectories of randomly selected nodes with three different layout stability methods. (Top) independent layout with average distance traveled 21.41; (Middle) layout initialized with positions from the previous frame with average distance traveled 13.19; (Bottom) initialized positions and Procrustes transformation, with average distance traveled 8.43.

The embedding of the graph was done using `Graphviz` [13]. 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 [22], available as part of `Graphviz` provides most of the geographic map functionality. Movies and an image gallery of the system can be found at `http://freeproject.us/dynamap`.

8 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 suggested a heuristic to address the challenge of dynamic cluster stability, proposed a strategy to preserve color stability, and analyzed the use of affine transformations to improve layout stability. We applied these results 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 $\{j_1, j_2, j_3 \cdots j_n\}$, where j_k's are the 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 $O(m)$ [18], however, the algorithm is not symmetric. That is, the solution obtain with respect to C_{old} need not be 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, however, this property cannot be guaranteed for all clusters. Another optimization criterion then could be to 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 el-

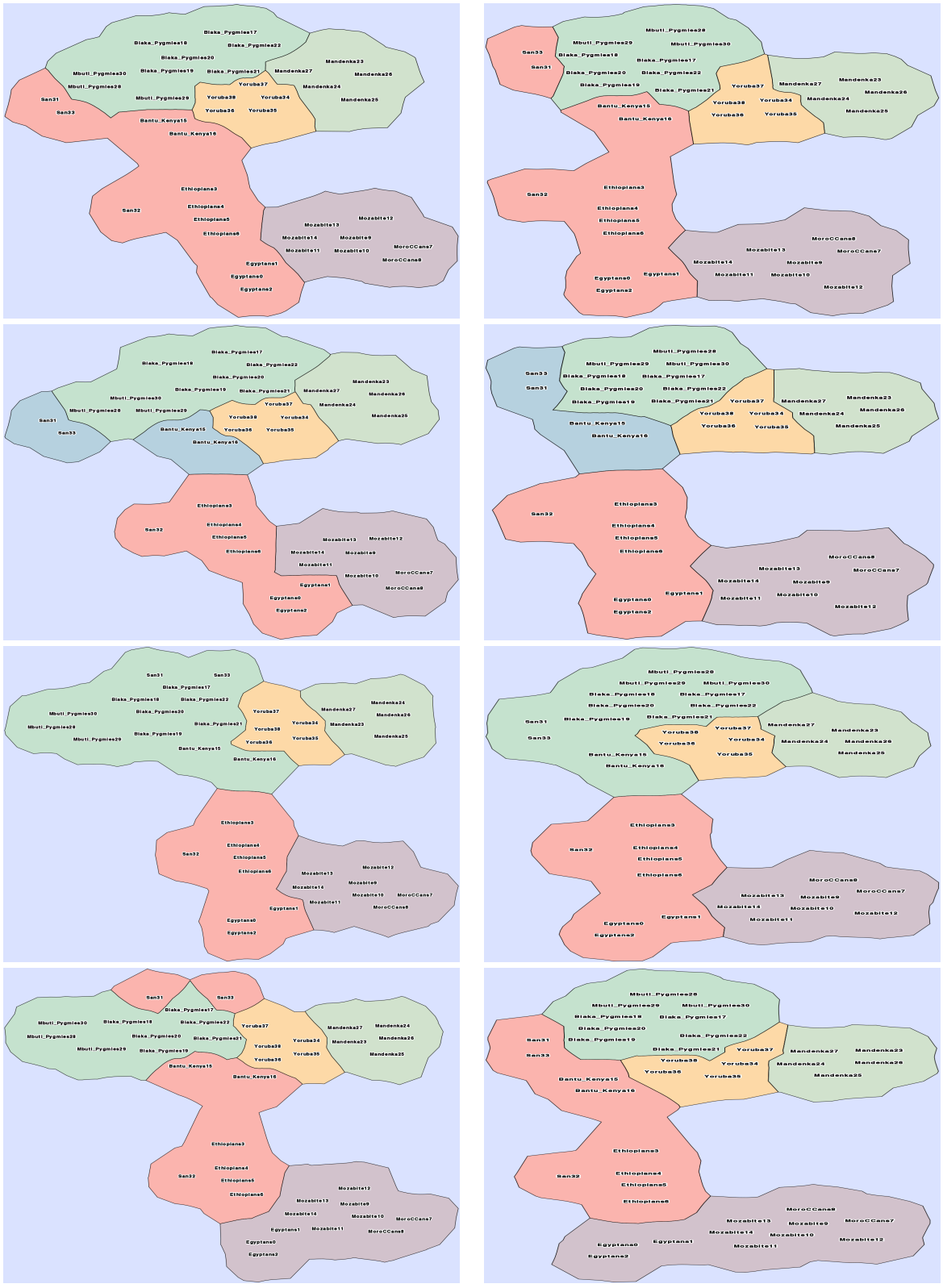

Figure 6: (Left) a sequence of frames without affine transformation; (Right) a sequence of frames with affine transformation. Note that the nodes exhibit far less jitter in the right sequence than in the one on the left.

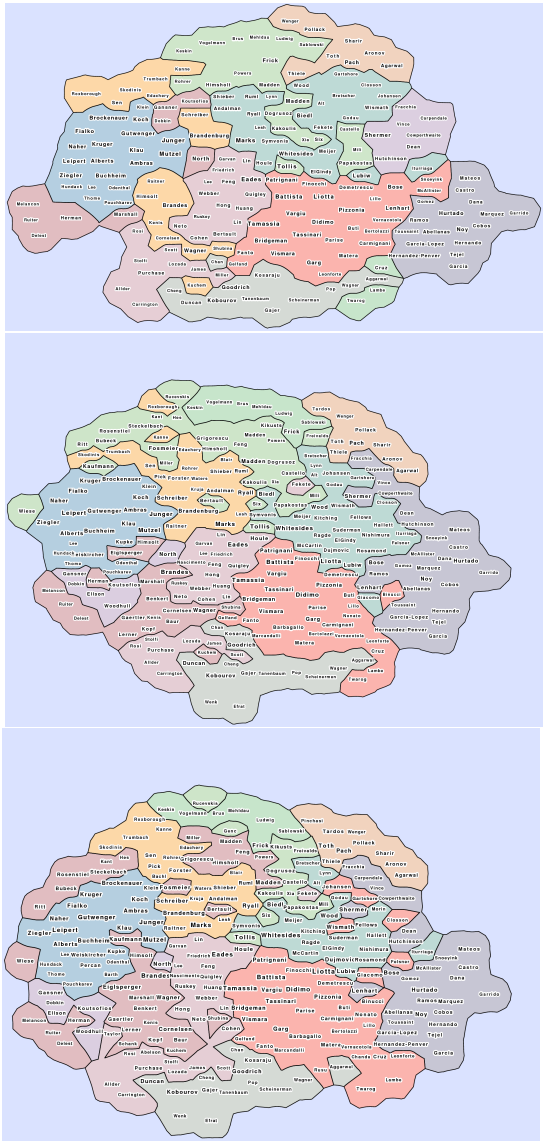

Figure 7: A sequence of GD collaboration graphs: 2000, 2001 and 2002.

ements 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 are currently using the system with the help of forbearing biologists who are using multiple DNA-similarity metrics. In addition to a formal evaluation of the effectiveness of our method, we would also like to be able to combine more than 2 similarity metrics, to allow for more general ways to combine multiple similarity metrics, and to deploy the system in a generic dynamic data setting.

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 *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 *WG'07*, pages 121–132, 2007.

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

[5] C. Brewer. ColorBrewer - Color Advice for Maps. `www.colorbrewer.org`.

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

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

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

[9] 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.

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

[11] 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.

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

[13] 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.

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

[15] 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.

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

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

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

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

[20] 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.

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

[22] Y. Hu, E. Gansner, and S. Kobourov. Visualizing Graphs and Clusters as Maps (PDF). *IEEE Computer Graphics and Applications*, 99(1), 2010.

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

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

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

[26] 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.

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

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

[29] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. of the American Statistical Association*, pages 846–850, 1971.

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