Visualizing the Temporal Evolution of Dynamic Networks

Kevin S. Xu
EECS Department
University of Michigan
1301 Beal Avenue
Ann Arbor, MI 48109 USA
xukevin@umich.edu

Mark Kliger
Medasense Biometrics Ltd.
PO Box 633
Ofakim, 87516 Israel
mark@medasense.com

Alfred O. Hero III
EECS Department
University of Michigan
1301 Beal Avenue
Ann Arbor, MI 48109 USA
hero@umich.edu

ABSTRACT

Many developments have recently been made in mining dynamic networks; however, effective visualization of dynamic networks remains a significant challenge. Dynamic networks are typically visualized via a sequence of static graph layouts. In addition to providing a visual representation of the network topology at each time step, the sequence should preserve the "mental map" between layouts of consecutive time steps to allow a human to interpret the temporal evolution of the network and gain valuable insights that are difficult to convey by summary statistics alone. We propose two regularized layout algorithms for visualizing dynamic networks, namely *dynamic multidimensional scaling* (DMDS) and *dynamic graph Laplacian layout* (DGLL). These algorithms discourage node positions from moving drastically between time steps and encourage nodes to be positioned near other members of their group. We apply the proposed algorithms on several data sets to illustrate the benefit of the regularizers for producing interpretable visualizations.

Keywords

Dynamic network, visualization, graph drawing, multidimensional scaling, graph Laplacian

1. INTRODUCTION

The study of networks has emerged as a topic of great interest in recent years, with applications in the social, computer, and life sciences among others. Dynamic networks are of particular interest because networks often grow or evolve over time [21]. Many developments have been made in mining dynamic networks, including the detection of groups or communities and how they evolve over time [27,31,33]. However, the fundamental task of visualizing dynamic networks has remained an open problem. Visualization is an important tool that can provide insights and intuition about networks that cannot be conveyed by summary statistics alone.

Visualizing static networks is a challenge in itself. Static networks are typically represented by graphs, which have no natural representation in a Euclidean space. Many graph drawing or layout algorithms have been developed to create aesthetically pleasing 2-D representations of graphs [16]. Common layout methods for general graphs include force-directed [13, 18] and spectral layouts [19].

Dynamic networks are typically represented by a sequence of graph snapshots; thus, visualizing dynamic networks in 2-D presents an additional challenge due to the temporal aspect. If one axis is used to represent time, then only a single axis remains to convey the topology of the network. Brandes and Corman [8] presented a possible solution to this problem by creating a pseudo-3-D visualization that treats 2-D layouts of each snapshot as layers in a stack. Unfortunately, the resulting visualization is difficult to interpret. The more conventional approach is to present an animated 2-D layout that evolves over time to reflect the current snapshot [2, 6, 22, 26]. A challenge with this approach is to preserve the "mental map" [24] between snapshots so that the transition between frames in the animation can be easily interpreted by a human. In particular, it is undesirable for a large number of nodes to drastically change positions between frames. The mental map can be preserved by a combination of two techniques. The first is to choose node positions at a particular time step so that they are close to their respective positions at neighboring time steps. The second is to use transition frames to smooth the movement between time steps. The latter is usually achieved by creating interpolated transition layouts [6, 26].

The contribution of this paper is in the former area. While interpolation does make an animation more aesthetically pleasing, it does not constrain the successive layouts, so there could be many node movements between time steps. We constrain movements between time steps by adding a temporal penalty to the cost function of a static graph layout method. In addition, we add a group penalty that encourages nodes belonging to the same group to be positioned near each other in the layout, which also helps a human to interpret the network structure. We propose two regularized layout methods, *dynamic multidimensional scaling* (DMDS) and *dynamic graph Laplacian layout* (DGLL) for dynamic networks. DMDS is a regularized version of multidimensional scaling, which is a family of methods for static graph layout that includes the popular Kamada-Kawai (KK) method of force-directed layout [18], while DGLL is a regularized version of spectral layout using the graph Laplacian.

The proposed regularized methods for visualizing dynamic networks can be used together with other graph mining techniques to integrate additional knowledge mined from the

MLG '11 San Diego, CA, USA

network into the visualization. For example, in many real networks, group information is not available a priori. The groups could be learned from the network topology by community detection [27, 31, 33] and then incorporated in the layout via the proposed group regularization. This approach of integrating mining methods with visualization has also been used in [14,30] for visualization of groups in static networks. Vice versa, the regularized visualization can also contribute to graph mining; for example, consider the problem of change detection. Significant node movements between layouts of consecutive time snapshots even in the presence of temporal regularization can indicate changes in the network structure.

To the best of our knowledge, this is the first work which incorporates both temporal and group regularization in MDS and graph Laplacian layout. Existing methods for temporal regularization in MDS [2] and group regularization in Laplacian layout [9] are subsumed by DMDS and DGLL, respectively. The methods for group regularization in MDS and temporal regularization in Laplacian layout proposed in this paper are novel. We apply the proposed algorithms on a selection of dynamic network data sets to demonstrate the contributions of the group and temporal regularizers in creating interpretable visualizations.

2. BACKGROUND

We begin by specifying the notation we shall use in this paper. Time-indexed quantities are indicated using square brackets, e.g. $X[t]$. We represent a dynamic network by a discrete-time sequence of graph snapshots. Each snapshot is represented by a graph adjacency matrix $A[t]$ where $a_{ij}[t] = 1$ if an edge is present between nodes i and j during time step t, and $a_{ij}[t] = 0$ otherwise. We assume all graphs are undirected, so that $a_{ij}[t] = a_{ji}[t]$. For simplicity of notation, we typically drop the time index for all quantities at time step t, i.e. A is assumed to denote $A[t]$.

We refer to a graph layout by a matrix $X \in \mathbb{R}^{n\times s}$, where each row $\mathbf{x}_{(i)}$ corresponds to the s-dimensional position of node i. We are most interested in 2-D visualization ($s = 2$), although the proposed methods can also be applied to other values of s. The ith column of X is denoted by $\mathbf{x}_i$, and the individual entries by x_{ij}. The superscript in $\mathbf{x}_i^{(h)}$ is used to denote the value of $\mathbf{x}_i$ at iteration h. Unless otherwise indicated, the norm operator $\|\cdot\|$ refers to the l_2-norm. The all ones column vector of length n is denoted by 1_n. $\mathrm{tr}(\cdot)$ denotes the matrix trace operator, and $\mathrm{diag}(\cdot)$ creates a diagonal matrix from its vector argument.

We now summarize the static graph drawing methods of MDS and spectral layout using the graph Laplacian, which operate on a single graph snapshot.

2.1 Multidimensional scaling

Multidimensional scaling (MDS) is a family of statistical methods that aim to find an optimal layout $X \in \mathbb{R}^{n\times s}$ such that the distance between $\mathbf{x}_{(i)}$ and $\mathbf{x}_{(j)}$ for all $i \neq j$ is as close as possible to a desired distance δ_{ij}. There are a variety of different cost functions and associated algorithms for MDS; we refer interested readers to [7]. Here we describe the cost function known as stress and its associated majorization algorithm. The stress of a layout X is given by

$$\mathrm{stress}(X) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(\delta_{ij} - \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|\right)^2, \quad (1)$$

where w_{ij} denotes the weight or importance of maintaining the desired distance δ_{ij}. Stress MDS can be used for graph layout by defining a distance metric over the graph. The desired distance δ_{ij} is typically taken to be the length of the shortest path between nodes i and j in the graph. The weights also play a crucial role in the aesthetics of the layout. The KK force-directed layout [18] simply corresponds to choosing $w_{ij} = \delta_{ij}^{-2}$ for $i \neq j$ and $w_{ii} = 0$.

The objective of stress MDS is to find a layout X that minimizes (1). (1) can be decomposed into

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|. \quad (2)$$

Note that the first term of (2) is independent of X. The second term of (2) can be written as $\mathrm{tr}(X^T RX)$ where the $n \times n$ matrix R is given by

$$r_{ij} = \begin{cases} -w_{ij} & i \neq j, \\ \sum_{k\neq i} w_{ik} & i = j. \end{cases} \quad (3)$$

$\mathrm{tr}(X^T RX)$ is quadratic and convex and is easily optimized.

The third term of (2) cannot be written as a quadratic form. It is typically majorized by a convex quadratic function, and the resulting upper bound for the stress is then optimized by differentiation. Define the matrix-valued function $S(Z)$ by

$$s_{ij}(Z) = \begin{cases} -w_{ij}\delta_{ij}/\|\mathbf{z}_{(i)} - \mathbf{z}_{(j)}\| & i \neq j, \\ -\sum_{k\neq i} s_{ik}(Z) & i = j. \end{cases} \quad (4)$$

Then, it is shown in [7,15] that

$$-\mathrm{tr}(X^T S(Z)Z) \geq -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\| \quad (5)$$

so that an upper bound for the stress is

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\delta_{ij}^2 + \mathrm{tr}(X^T RX) - 2\,\mathrm{tr}(X^T S(Z)Z). \quad (6)$$

By setting the derivative of (6) with respect to X to 0, the minimizer of the upper bound is found to be the solution to the equation $RX = S(Z)Z$.

The algorithm for optimizing stress is iterative. Let $X^{(0)}$ denote an initial layout. Then at each iteration h, solve

$$R\,\mathbf{x}_a^{(h)} = S\left(X^{(h-1)}\right)\mathbf{x}_a^{(h-1)} \quad (7)$$

for $\mathbf{x}_a^{(h)}$ for each $a = 1, \ldots, s$. R is rank-deficient because the stress function is translation-invariant. The translation-invariance can be removed by fixing the location of one point, e.g. by setting $\mathbf{x}_{(1)} = 0$, removing the first row and column of R, and removing the first row of $S\big(X^{(h-1)}\big)X^{(h-1)}$ [15]. (7) can then be solved using a standard method for solving linear equations, such as Cholesky factorization. Alternatively, (7) can be solved directly by using the Moore-Penrose pseudoinverse [7]. The iteration can be terminated when

$$\frac{\mathrm{stress}\left(X^{(h-1)}\right) - \mathrm{stress}\left(X^{(h)}\right)}{\mathrm{stress}\left(X^{(h-1)}\right)} < \epsilon, \quad (8)$$

where ϵ is the tolerance of the process.

2.2 Graph Laplacian layout

The family of spectral layout methods involve using the eigenvectors of a matrix representation of the graph. Typically this is the graph Laplacian, obtained from the adjacency matrix by $L = D - A$, where D is the diagonal matrix of node degrees defined by $d_{ii} = \sum_{j=1}^{n} a_{ij}$. The graph Laplacian spectral layout problem can be solved sequentially in each dimension. In dimension i, the problem can be formulated as [19]:

$$\min_{\mathbf{x}_i} \quad \mathbf{x}_i^T L \mathbf{x}_i \tag{9}$$

$$\text{subject to} \quad \mathbf{x}_i^T \mathbf{x}_i = 1 \tag{10}$$

$$\mathbf{x}_i^T 1_n = 0 \tag{11}$$

$$\mathbf{x}_i^T \mathbf{x}_j = 0,\ j = 1, 2, \ldots, i-1. \tag{12}$$

By the definition of the graph Laplacian L, it can be shown that for any vector $\mathbf{y} \in \mathbb{R}^n$,

$$\mathbf{y}^T L \mathbf{y} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} (y_i - y_j)^2. \tag{13}$$

Hence (9) can be viewed as the energy of the layout, and the minimization aims to make edge lengths short [19]. (11) removes the trivial solution $\mathbf{x}_i = 1_n/\sqrt{n}$, which collapses all nodes to a single point. It can also be viewed as removing a degree of freedom in the layout due to translation invariance [4] by setting the mean of $\mathbf{x}_i$ to 0. Since $\mathbf{x}_i$ has zero-mean, (10) can be viewed as a constraint on the variance of the layout. Finally, (12) forces each dimension of the layout to be orthogonal to all previously obtained dimensions to provide as much additional information as possible. By a generalization of the Rayleigh-Ritz theorem [23], the minimizer is $\mathbf{x}_i^* = \mathbf{v}_{i+1}$, the eigenvector corresponding to the $(i+1)$th smallest eigenvalue of L.

In practice, it has been found that using degree-normalized eigenvectors often results in more aesthetically pleasing layouts [4, 19]. The degree-normalized layout problem differs only in that the dot product in each of the constraints is replaced with the degree-weighted dot product, i.e. (10) to (12) become $\mathbf{x}_i^T D \mathbf{x}_i = 1$, $\mathbf{x}_i^T D 1_n = 0$, and $\mathbf{x}_i^T D \mathbf{x}_j = 0$, respectively. The minimizer is $\mathbf{x}_i^* = \mathbf{u}_{i+1}$, the generalized eigenvector corresponding to the $(i+1)$th smallest generalized eigenvalue of (L, D). A discussion on the merits of the degree normalization can be found in [19].

3. REGULARIZED LAYOUT METHODS

We now introduce the regularized methods DMDS and DGLL for layout of dynamic networks. The two regularizers we use consist of a penalty to keep nodes that are in the same group together and a penalty to prevent a node's position from deviating too far from its position at the previous time step. We assume that group memberships are known (or mined from the network) for at least some nodes. If the group membership of a particular node is not known, no group penalty is applied to that node.

3.1 Dynamic multidimensional scaling

Assume that there are n nodes and k groups. Define the group membership by an $n \times k$ matrix C where

$$c_{ij} = \begin{cases} 1 & \text{node } i \text{ is in group } j \text{ at time step } t, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

We introduce group regularization using the augmented adjacency matrix defined by

$$B = \begin{bmatrix} A & C \\ C^T & 0 \end{bmatrix}, \tag{15}$$

where 0 denotes the $k \times k$ all-zero matrix. The added nodes can be thought of as centroids of the groups, and an edge is added between each node and its associated centroid. Let $\Delta = [\delta_{ij}]$ denote the matrix of shortest-path distances between each pair of nodes in B (including the centroids).

Define the regularized stress of an $(n+k) \times s$ layout matrix X by

$$\begin{aligned} \text{stress}(X) = \frac{1}{2} \sum_{i=1}^{n+k} \sum_{j=1}^{n+k} w_{ij} \left(\delta_{ij} - \left\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\right\|\right)^2 \\ + \beta \sum_{i=1}^{n} e_i \left\|\mathbf{x}_{(i)} - \mathbf{x}_{(i)}[t-1]\right\|^2, \end{aligned} \tag{16}$$

where the weight matrix W is given by

$$w_{ij} = \begin{cases} 0 & i = j \\ \delta_{ij}^{-2} & i, j \le n, i \ne j \\ \alpha \delta_{ij}^{-2} & \text{otherwise,} \end{cases} \tag{17}$$

and the indicator vector $\mathbf{e}$ is given by

$$e_i = \begin{cases} 1 & \text{node } i \text{ was present at time step } t-1, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

The first term of (16) is the usual MDS stress function augmented with group regularization, and the second term is the temporal regularization. α and β are the group and temporal regularization parameters, respectively. Notice that in (17), we have used the KK choice of weights discussed in Section 2.1; however, other weighting schemes can be trivially substituted. Define the diagonal matrix

$$E = \text{diag}(\mathbf{e}). \tag{19}$$

Then the second term in (16) can be written as

$$\begin{aligned} \beta \Big[\operatorname{tr}(X^T E X) + \operatorname{tr}(X^T[t-1] E X[t-1]) \\ - 2 \operatorname{tr}(X^T E X[t-1]) \Big]. \end{aligned}$$

Following the derivation in Section 2.1, for any $(n+k) \times s$ matrix Z, (16) can be majorized by

$$\begin{aligned} \frac{1}{2} \sum_{i=1}^{n+k} \sum_{j=1}^{n+k} w_{ij} \delta_{ij}^2 + \operatorname{tr}(X^T R X) - 2 \operatorname{tr}(X^T S(Z) Z) \\ + \beta \Big[\operatorname{tr}(X^T E X) + \operatorname{tr}(X^T[t-1] E X[t-1]) \\ - 2 \operatorname{tr}(X^T E X[t-1]) \Big], \end{aligned} \tag{20}$$

where R and S are defined in (3) and (4), respectively. (20) is quadratic and convex in X so the minimizer is found by setting the derivative of (20) to 0, resulting in the equation

$$(R + \beta E) X = S(Z) Z + \beta E X[t-1]. \tag{21}$$

This can be solved sequentially over each dimension. As in Section 2.1, we solve this iteratively using the previous iteration as the majorizer, i.e. at iteration h, solve

$$(R + \beta E) \mathbf{x}_a^{(h)} = S(\mathbf{x}_a^{(h-1)}) \mathbf{x}_a^{(h-1)} + \beta E \mathbf{x}_a[t-1]. \tag{22}$$

(22) is solved for each dimension $a = 1, \ldots, s$, and the process is iterated until the convergence criterion in (8) is attained. The first iterate can be taken to be simply the previous layout $\mathbf{x}_a[t-1]$. Unlike in ordinary MDS, the system of linear equations in (22) has a unique solution provided that at least a single node was present at time step $t-1$, because $R + \beta E$ has full rank in this case. Hence there is no need to compute a pseudo-inverse to solve (22) except at the initial time step when no temporal regularization is performed. At all subsequent time steps, a standard method for solving linear equations, such as Cholesky factorization can be used to solve (22).

3.2 Dynamic graph Laplacian layout

Define C, $\mathbf{e}$, and E as in (14), (18), and (19), respectively. The augmented adjacency matrix B is defined slightly differently as

$$B = \begin{bmatrix} A & \alpha C \\ \alpha C^T & 0 \end{bmatrix}. \tag{23}$$

Notice that the group regularization parameter α is directly incorporated into B. Define the augmented diagonal matrix of node degrees D by $d_{ii} = \sum_{j=1}^{n+k} b_{ij}$, and the graph Laplacian by $L = D - B$.

We derive the solution for the degree-normalized Laplacian layout. The unnormalized Laplacian layout can simply be obtained by replacing D with I in the derivation. The regularized Laplacian layout in dimension i is given by the solution to

$$\min_{\mathbf{x}_i} \quad \mathbf{x}_i^T L \mathbf{x}_i + \beta \left\| E\left(\mathbf{x}_i - \mathbf{x}_i[t-1]\right)\right\|_D^2 \tag{24}$$

$$\text{subject to} \quad \mathbf{x}_i^T D \mathbf{x}_i - \mathbf{x}_i^T M \mathbf{x}_i = 1 \tag{25}$$

$$\mathbf{x}_i^T D \mathbf{x}_j - \mathbf{x}_i^T M \mathbf{x}_j = 0,\ j = 1, 2, \ldots, i-1, \tag{26}$$

where $\|\mathbf{x}\|_D^2 = \mathbf{x}^T D \mathbf{x}$, the norm induced by the degree-weighted dot product, and M is defined by

$$M = \frac{D \mathbf{1}_n \mathbf{1}_n^T D}{\mathbf{1}_n^T D \mathbf{1}_n}.$$

Again, the first term of (24) is the usual Laplacian layout cost function augmented with group regularization, and the second term is the temporal regularization. Unlike the unregularized Laplacian layout problem presented in Section 2.2, the trivial solution of the scaled all ones vector is no longer optimal due to the temporal penalty, so we do not restrict the solution to have zero mean. Thus the variance constraint (25) is slightly more complex. (24) can be rewritten as

$$\begin{aligned}\mathbf{x}_i^T L \mathbf{x}_i + \beta \mathbf{x}_i^T E D \mathbf{x}_i - 2\beta \mathbf{x}_i^T E D \mathbf{x}_i[t-1] \\ + \beta \mathbf{x}_i^T[t-1] E D \mathbf{x}_i[t-1]\end{aligned}$$

The final term is independent of $\mathbf{x}_i$, so it can be dropped from the cost function. Due to the presence of the linear term in $\mathbf{x}_i$, the problem cannot be solved using eigenvectors as in the static case. Using the method of Lagrangian multipliers, a necessary condition for the optimizer is given by the equations

$$\begin{aligned}\tilde{L}\mathbf{x}_i - \frac{1}{2}\sum_{j=1}^{i-1} \mu_j (D - M)\mathbf{x}_j &= \beta E D \mathbf{x}_i[t-1] \\ \mathbf{x}_i^T D \mathbf{x}_i - \mathbf{x}_i^T M \mathbf{x}_i &= 1 \\ \mathbf{x}_i^T D \mathbf{x}_j - \mathbf{x}_i^T M \mathbf{x}_j &= 0,\ j = 1, 2, \ldots, i-1,\end{aligned} \tag{27}$$

where

$$\tilde{L} = L + \beta E D - \mu_i (D - M),$$

and $\boldsymbol{\mu} = [\mu_j]_{j=1}^i$ is the vector of Lagrangian multipliers. We propose to solve (27) using Newton's method [3]. Although there any many solution pairs $(\mathbf{x}_i, \boldsymbol{\mu})$ to (27), we find empirically that initializing Newton's method with $\mathbf{x}_i[t-1]$ and randomly chosen $\boldsymbol{\mu}$ allows it to converge to a local minimum near $\mathbf{x}_i[t-1]$, which is a desirable solution because the objective of the temporal regularization is to ensure that $\mathbf{x}_i$ does not deviate too far from $\mathbf{x}_i[t-1]$. At the initial time step, there is no temporal penalty, so the usual spectral layout approach of using generalized eigenvectors of (L, D), as described in Section 2.2, is applied.

3.3 Discussion

The form of the group and temporal penalties are chosen to be consistent with the original cost functions of MDS and Laplacian layout. Consider first the temporal penalty, which has the same form

$$\beta \sum_{i=1}^{n} e_i \left\| \mathbf{x}_{(i)} - \mathbf{x}_{(i)}[t-1] \right\|^2 \tag{28}$$

for both DMDS and unnormalized DGLL[1]. Recall the cost function of unnormalized Laplacian layout is given by (9). By comparing the forms of (13) and (28), it can be seen that the temporal penalty in DGLL corresponds to placing anchor nodes at the previous node positions and minimizing the distance between current node positions and their respective anchors with weight β.

Similarly, by comparing the forms of (1) and (28), it can be seen that the temporal penalty in DMDS also corresponds to placing anchor nodes at the previous node positions. The desired distance between current node positions and their respective anchors is simply taken to be 0 and has weight β. For both DMDS and DGLL, increasing β has the effect of pulling each node towards its previous position.

The group penalty differs slightly between DMDS and DGLL. The group penalty for DGLL can be written as

$$\alpha \sum_{i=1}^{n} \sum_{j=n+1}^{n+k} b_{ij} \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2,$$

where $\mathbf{x}_{(j)}, j > n$ represent the positions of the group centroids. Thus the group penalty in DGLL attempts to pull nodes within a group towards their centroid with weight α. As α increases, this results in nodes in the same group being collapsed onto the same point, while the distance between nodes in different groups increases to satisfy the variance constraint of (25).

On the other hand, the group penalty for DMDS is more complicated. By inserting k centroids with desired distance 1 from all nodes in their respective groups, we have essentially created shortcuts between distant nodes within the same group. That is, the distance between two nodes i, j in the same group is now $\min(2, \tilde{\delta}_{ij})$, where $\tilde{\delta}_{ij}$ denotes the shortest-path distance between i and j before the insertion of the centroids. The insertion of these shortcuts has the effect of lowering the desired distance between nodes in the same group that were previously far apart, which pulls them closer together in the layout. Unlike the group penalty in

[1]For normalized DGLL, replace the l_2-norm with $\|\cdot\|_D$.

DGLL, however, increasing α does not collapse all nodes within a group because it increases the weights of the desired distances between nodes and centroids, which is 1 between a node and the centroid of the group it belongs to. As a result, increasing α tends to give groups a more spherical appearance while pulling nodes in the same group together.

Since the cost functions of DMDS and DGLL encourage different appearances, the decision of which type of layout to use depends on the type of network and user preferences. Kamada-Kawai MDS layouts are often preferred because they discourage nodes from overlapping, and this is true of the group penalty in DMDS as well. On the other hand, if a 1-D layout is desired, so that the entire sequence can be plotted as a time series, node overlap is a lesser concern. For such applications, Laplacian layout may be a better choice.

Another decision that needs to be made by the user is the choice of the parameters α and β, which can be tuned as desired to create a meaningful animation. Unlike in supervised learning tasks such as classification, there is no ground truth in visualization so the selection of parameters in layout methods is typically done in an ad-hoc fashion. Furthermore, multiple layouts created by differing choices of parameters could be useful for visualizing different portions of the network or yielding different insights [32]. We explore the effect of changing parameters on the resulting animation in Section 5.1.

4. RELATED WORK

The regularized dynamic network layout algorithms proposed in this paper are designed to achieve two goals: to place nodes belonging to the same group together and to place nodes near their positions at neighboring time steps. The former problem has been investigated, albeit in a static setting, in the related field of supervised dimensionality reduction. The latter problem has been formulated as an objective in several other papers on dynamic network layout.

4.1 Supervised dimensionality reduction

The objective of dimensionality reduction (DR) is to find a mapping $\phi : \mathbb{R}^p \to \mathbb{R}^s$, $p > s$ from a high-dimensional space to a lower-dimensional space while preserving many of the characteristics of the data representation in the high-dimensional space [20]. For example, MDS is a DR method that attempts to preserve pairwise distances between data points. In the supervised DR setting, one also has a priori knowledge of the group structure of the data. Supervised DR methods pose the additional constraint that data points within the same group should be closer together in the low-dimensional space than points in separate groups. Notice that this is the same group constraint we pose in our regularized layout algorithms.

Witten and Tibshirani [32] proposed a supervised version of MDS (SMDS) that optimizes the following cost function over X:

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij} - \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|)^2 + \alpha \sum_{i,j:y_j>y_i}(y_j - y_i)\sum_{a=1}^{s}\left(\frac{\delta_{ij}}{\sqrt{s}} - (x_{ja} - x_{ia})\right)^2, \quad (29)$$

where y_i is an ordinal value denoting the group membership of data point i. Notice that the first term in (29) is the ordinary MDS stress with $w_{ij} = 1$ for all i, j, while the second term provides the group regularization. α controls the trade-off between the two terms.

The key difference between the SMDS group penalty and the DMDS group penalty proposed in this paper is in the way groups are treated. SMDS assumes that groups are labeled with an ordinal value that allows them to be ranked, and the form of the group penalty in (29) does indeed tend to rank groups in $\mathbb{R}^s$ by encouraging $x_{ja} > x_{ia}, a = 1, \ldots, s$ for all $i, j : y_j > y_i$. On the other hand, our proposed group penalty of adding k centroid nodes to the graph and connecting them to their respective group members treats group labels as categorical. Thus, our proposed group penalty does not rank groups in $\mathbb{R}^s$ but simply pulls nodes belonging to the group together.

Another related method for supervised DR is classification constrained dimensionality reduction (CCDR) [9], which is a supervised version of Laplacian eigenmaps [4]. CCDR optimizes the following cost function over (X, Y):

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2 + \alpha\sum_{l=1}^{k}\sum_{i=1}^{n}c_{il}\|\mathbf{y}_{(l)} - \mathbf{x}_{(i)}\|^2,$$

where C is as defined in (14), $\mathbf{x}_{(i)}$ denotes the position of the ith data point in $\mathbb{R}^s$, and $\mathbf{y}_{(l)}$ denotes the position of the lth centroid in $\mathbb{R}^s$. The solution [9] is given by the matrix of generalized eigenvectors $U = [\mathbf{u}_2, \ldots, \mathbf{u}_{s+1}]$ of (L, D) where the graph Laplacian L and diagonal degree matrix D are calculated with respect to the augmented adjacency matrix B defined in (23). The first n rows of U correspond to the positions of the data points X, and the last k rows correspond to the positions of the centroids Y.

Notice that the group penalty in CCDR is identical to the group penalty in DGLL. Indeed, DGLL can be viewed as an extension of CCDR to time-varying data. Although the addition of the temporal regularization due to the anchoring presence of the previous layout $X[t-1]$ prevents the DGLL layout problem from being solved by eigendecomposition, it discourages large node movements between time steps in order to preserve the mental map.

4.2 Layout of dynamic networks

There have been several previous studies on the problem of laying out dynamic networks while preserving the mental map between time snapshots. As mentioned in the introduction, the problem separates into two areas: choosing node positions at successive time steps so that node movement is minimized and generating transition layouts that interpolate between layouts of consecutive time snapshots. The former is more closely related to the contribution of this paper.

Moody et al. [26] proposed to generate dynamic layouts using a static layout method such as KK and to initialize at each time step using the layout generated at the previous time step. The idea of this approach is to anchor the nodes initially so that the entire layout does not get rotated. The anchoring differs from the layout methods proposed in this paper, which penalize changes in node positions over time and can be thought of as anchoring all iterations rather than just the initial iteration. Experimentally, we find that solely anchoring the initialization is insufficient at preventing drastic node movements over time (see Section 5 for examples).

Baur and Schank [2] proposed a temporally regularized MDS algorithm that uses the following localized update rule

at each iteration h for each node i at each time step t:

$$x_{ia}^{(h)} = \frac{\tilde{x}_{ij}^{(h-1)} + \beta(e_i x_{ia}[t-1] + f_i x_{ia}[t+1])}{\sum_{j \neq i} w_{ij} + \beta(e_i + f_i)}, \qquad (30)$$

where

$$\tilde{x}_{ij}^{(h-1)} = \sum_{j \neq i} w_{ij} \left(x_{ja}^{(h-1)} + \delta_{ij} \frac{x_{ia}^{(h-1)} - x_{ja}^{(h-1)}}{\|\mathbf{x}_{(i)}^{(h-1)} - \mathbf{x}_{(j)}^{(h-1)}\|} \right),$$

and $\mathbf{f}$ is the indicator vector defined by

$$f_i = \begin{cases} 1 & \text{node } i \text{ is present at time step } t+1, \\ 0 & \text{otherwise.} \end{cases}$$

This algorithm was used in [22] for visualizing similarities in journal content over time. It was shown in [2] that the localized update of (30) also optimizes the temporally regularized stress function in (16) with $k = 0$, i.e. without a group penalty. Hence the proposed DMDS layout method can be viewed as a generalization of the method of [2] with a group penalty. Note that (30) is an off-line update because it uses both the node positions at time steps $t-1$ and time steps $t+1$ to compute the node position at time step t, whereas DMDS can be used in the on-line scenario.

Other methods for layout of dynamic networks have also been proposed [11, 12]. TGRIP [11] is a modified force-directed layout method with added edges between vertices present at multiple time steps. The user-selected weights of these added edges control the amount of temporal regularization in the layouts. The method of Frishman and Tal [12] is also a modified force-directed layout. It is an on-line method that uses pinning weights to previous node positions to achieve temporal regularization and multi-scale computation to improve scalability. However, these methods do not incorporate any sort of group regularization to encourage nodes from the same group to be placed together in the layout, unlike the methods proposed in this paper.

5. EXPERIMENTS

We demonstrate the the proposed DMDS and DGLL layout methods on three dynamic network data sets. Several snapshots of the resulting visualizations are presented. The full, animated visualizations over all time steps can be found in the supplementary material, available on-line [1].

Summary statistics from applying both DMDS and DGLL on each of the data sets are presented in Tables 1 and 2, respectively, and are discussed in the individual subsections. When group regularization is used, $\alpha = 1$ unless otherwise specified. Similarly, when temporal regularization is used, $\beta = 1$ unless otherwise specified. We define three measures of layout quality: *snapshot cost*, *temporal distance*, and *group distance*. The snapshot cost measures how well the current layout coordinates fit the current graph snapshot. The snapshot cost for DMDS is taken to be that static MDS stress defined in (1). From (9), the snapshot cost for DGLL is the layout energy given by $\mathrm{tr}(X^T L X)$. The temporal distance is a measure of the amount of node movement between consecutive layouts, and the group distance represents how well the current layout reflects the group structure. We take the temporal distance to be the squared Frobenius norm distance between node positions at consecutive time steps $\|X - X[t-1]\|_F^2$ and the group distance to be the sum of the squared pairwise Euclidean distances between all pairs of nodes within a group, summed over all groups. The costs and distances displayed are appropriately normalized (either by the number of nodes or pairs of nodes, depending on the quantity) so they are comparable across different data sets.

Data set	Regularizer	Stress	Temporal	Group
SBM	Both	0.174	**0.247**	**2.49**
	Temporal	0.173	0.307	4.41
	Group	0.160	0.672	3.51
	None	**0.153**	1.07	6.10
Newcomb	Both	0.112	**0.107**	**3.22**
	Temporal	0.106	0.116	3.49
	Group	0.102	0.493	3.69
	None	**0.0953**	0.681	4.27
MIT	Both	0.169	**0.369**	**4.36**
	Temporal	0.113	0.861	7.59
	Group	0.159	1.00	4.58
	None	**0.0956**	3.06	9.28

Table 1: Mean normalized costs and distances of DMDS layouts for all data sets. The smallest quantity in each column for each data set is bolded.

Data set	Regularizer	Energy	Temporal	Group
SBM	Both	1.82	**0.135**	**2.99**
	Temporal	1.89	0.179	10.1
	Group	1.48	7.01	8.56
	None	**1.39**	6.94	20.9
Newcomb	Both	3.11	**0.332**	30.4
	Temporal	2.97	0.365	30.4
	Group	2.19	22.7	**28.4**
	None	**2.12**	32.7	29.3
MIT	Both	0.465	**0.530**	10.5
	Temporal	0.431	2.26	14.6
	Group	0.288	8.26	**10.2**
	None	**0.253**	13.6	19.3

Table 2: Mean normalized costs and distances of DGLL layouts for all data sets. All entries are $\times 10^{-3}$.

As expected, the regularized layouts have lower group and temporal distance than the unregularized layouts, which is achieved by choosing node positions with a slightly higher snapshot cost. The improvement in temporal distance is lower for DMDS than for DGLL. This is due to the initialization of the unregularized MDS using the previous node positions, as discussed in Section 3.3, which has some effect but not as much as the temporal regularization in DMDS. Also, note that using both regularizers often results in better performance both in group and temporal distance, but not always because the two penalties could oppose each other, which we discuss at the end of Section 5.1.

5.1 Stochastic block model

In this experiment, we generate simulated networks using a stochastic block model (SBM) [17]. An SBM creates networks with k groups, where nodes in a group are stochastically equivalent, i.e. the probability of an edge between nodes i and j is dependent only on the groups to which i and j belong. An SBM is completely specified by the set

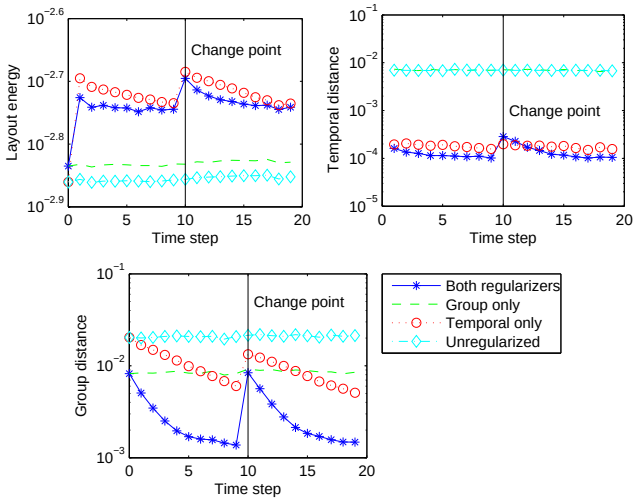

Figure 1: Costs and distances of DGLL layouts in SBM simulation at each time step.

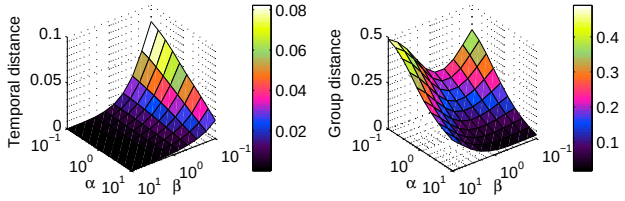

Figure 2: Mean temporal and group distances in SBM simulation as functions of α and β.

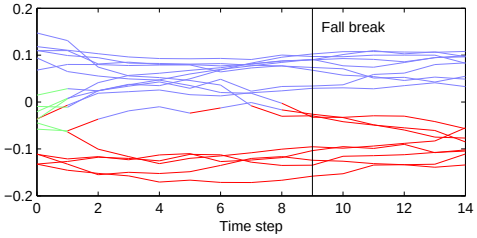

Figure 3: Time plots of 1-D DGLL layouts of Newcomb's fraternity, colored by group.

of probabilities $\{p_{cd}, c = 1, \ldots, k, d = c, c+1, \ldots, k\}$, which specify the probability of an edge between any particular node in group c and any particular node in group d.

We generate 20 independent samples from a 30-node 4-group SBM with parameters $p_{ii} = 0.6$ and $p_{ij} = 0.3$ for all $i \neq j$. Each sample corresponds to a graph snapshot at a single time step. The group memberships are randomly assigned at the initial time step. At $t = 10$, 1/4 of the nodes are randomly re-assigned to different groups to simulate a change in the network structure. We create layouts of the network using normalized DGLL with $\alpha = \beta = 1$.

In Figure 1, we plot the variation over time of the snapshot cost, the temporal distance, and the group distance. The quantities are averaged over 100 simulation runs. As expected, the snapshot cost is higher for the regularized layouts than for the static (unregularized) layout. Notice that temporal regularization allows the temporal distance to decrease significantly, which is crucial in order to preserve the mental map over time. Finally, it can be seen that group regularization results in lower group distance, as expected, but temporal regularization also results in lower group distance after several time steps have elapsed in this experiment. This is because the SBM parameters are held constant from time steps 0 to 9 and from time steps 10 to 19, so that the group structure can be revealed by temporal regularization alone once enough time samples have been collected. These findings apply also to the DMDS layouts, as summarized in Table 1.

We demonstrate the effect of varying the regularization parameters in Figure 2. We generate layouts using 10 choices each of α and β, uniformly distributed on a log scale between 0.1 and 10. As expected, the temporal distance drops for increasing β, but notice also that for low values of β, increasing the group penalty also decreases the temporal distance. This is a sensible result because nodes can move significantly over time but must remain close to the group centroid. The result is different when it comes to group distance. As expected, increasing α decreases group distance; however, for low values of α, a moderate value of β provides the lowest group distance. This is also a sensible result in this experiment, as discussed previously, because the SBM parameters are held constant before and after the change point. An extremely low β is detrimental because it does not constrain node positions enough, while an extremely high β is also detrimental because it places too much weight on the initial time step.

From this experiment we can see that there is a coupled effect between group and temporal regularization, and that a combination of both penalties can sometimes result in better performance with respect to both temporal and group distance. However, it is important to note that this is not always true. For example, if a node changes group between two time steps, then the group and temporal regularization can oppose each other, with the temporal penalty attempting to pull the node towards its previous position and the group penalty attempting to pull the node towards its current centroid, which could be quite far from the node's previous position. This is reflected by the spike in temporal and group distances at $t = 10$ in Figure 1 when both regularizers are used.

5.2 Newcomb's fraternity

This data set was collected by Nordlie and Newcomb [28, 29] as part of an experiment on interpersonal relations. 17 incoming male transfer students at the University of Michigan were housed together in fraternity housing. Each week, the participants ranked their preference of the other individuals in the house, in private, from 1 to 16, where 1 indicates highest preference. Data was collected over 15 weeks in a semester, with one week of data missing during week 9, corresponding to Fall break.

The rank data was first symmetrized by taking the average of participant i's preference for participant j and vice versa. Then, graph snapshots were created by connecting each participant to his 3 most preferred neighbors. No group information is known a priori, so the group structure is learned using the AFFECT evolutionary spectral clustering algorithm [33]. We first create layouts by normalized DGLL with $\alpha = \beta = 1$. In Figure 3, we show a time plot of 1-D

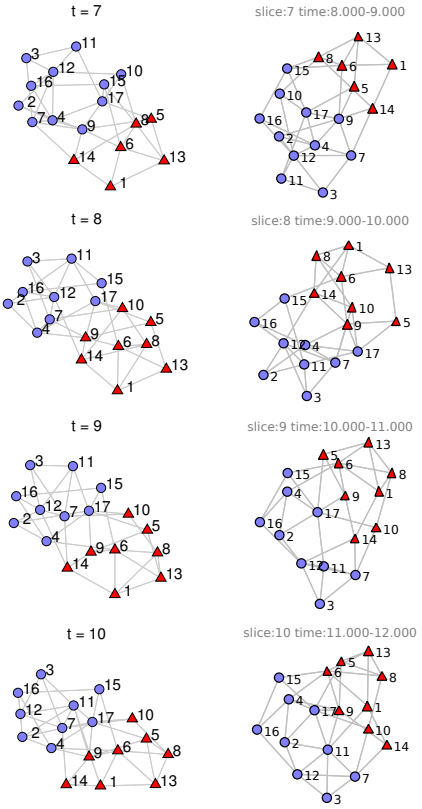

Figure 4: Comparison of proposed DMDS layout (left) with anchored KK layout (right) of Newcomb's fraternity at four time steps (animation on-line [1]).

layouts, where the color of a line segment between time steps t and $t+1$ denotes the group membership of the node at time step t. While a 1-D layout does a poor job of conveying the topology of the network, some temporal trends can be seen. For example, three nodes appear to switch from the blue to the red group around Fall break.

The Newcomb data was also analyzed by Moody et al. [26] using the method of anchoring the initial iteration as discussed in section 3.3. In Figure 4, we present a side-by-side comparison of several snapshots from the DMDS layout and the anchored KK layout, generated using SoNIA [5]. The anchored KK layout is equivalent to performing ordinary MDS at each time step using the weights discussed in Section 2.1 and the previous node positions as the initialization. In both layouts, it can be seen that nodes 9, 10, and 14 break away from the blue group and join the red group. This change is detected by the clustering procedure and is reflected in the animation, available on-line [1]. However, there is a lot of node movement between time snapshots 7 to 9 in the anchored KK layout and is reflected in the high temporal distance of the unregularized MDS layout in Table 1. The DMDS layout has much less node movement between snapshots, achieved by a small increase in the static stress.

5.3 MIT Reality Mining

The MIT Reality Mining data set [10] was collected as part of an experiment on inferring social networks by using

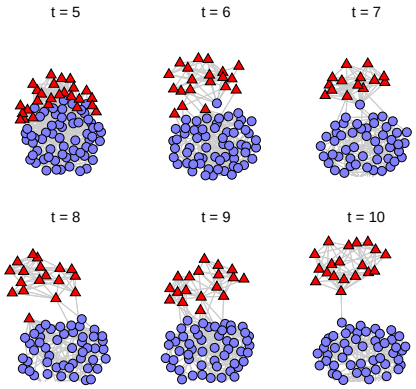

Figure 5: DMDS layout of MIT Reality Mining data at six time steps (animation on-line [1]). Blue nodes denote colleagues working in the same building, and red nodes denote incoming students.

cell phone as sensors. 94 students and staff at MIT were given access to smart phones that were monitored over two semesters. The phones were equipped with Bluetooth sensors, and each phone recorded the Media Access Control addresses of nearby Bluetooth devices at five-minute intervals. Using this proximity data, we construct a sequence of graph snapshots where each participant is connected to the 5 participants he or she was in highest proximity to during a time step. We divide the data into time steps of one week, resulting in 46 time steps between August 2004 and June 2005. From the MIT academic calendar [25], we know the dates of important events such as the beginning and end of school terms. We also know that 26 of the participants were incoming students at the university's business school, while the rest were colleagues working in the same building.

Using the affiliations as groups, we apply both DMDS and DGLL to the dynamic network. The layouts at six time steps for DMDS with $\alpha = \beta = 1$ are shown in Figure 5. Node labels are not displayed to reduce clutter in the figure. We encourage readers to view the animation on-line [1] to get a better idea of the temporal evolution of this network. $t = 5$ corresponds to the first week of classes. Notice that the two groups are slightly overlapped at this time step. As time progresses, the group of incoming students separates quite clearly from the colleagues working in the same building. This result suggests that the incoming students are spending more time in proximity with each other than with the remaining participants, which one might expect as the students gain familiarity with each other as the semester unfolds. The benefit of the regularization can be seen once again from the costs and distances in Tables 1 and 2.

6. CONCLUSIONS

In this paper we proposed two methods for generating layouts of dynamic networks over time. The proposed methods DMDS and DGLL incorporate group and temporal regularization to encourage nodes in the same group to be positioned near each other and to discourage nodes from straying too far from their previous position. We demonstrated the necessity of the regularizers for preserving the mental map

in multiple experiments. The proposed methods generalize existing approaches for temporal regularization in MDS and group regularization in graph Laplacian layout.

In this paper, we chose small networks of less than 100 nodes for demonstrative purposes. An important area for future work concerns visualization of large dynamic networks containing upwards of thousands of nodes. Even when equipped with temporal and group regularization, dynamic layouts of large networks may be confusing for a human to interpret. The integration of additional graph mining methods into visualization algorithms could be useful in dealing with these large networks. In addition, there is the issue of improving the scalability of the proposed algorithms, which can be accomplished by using multi-scale methods as in [12], which we plan to investigate in the future.

7. ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation grant CCF 0830490. Kevin Xu was partially supported by an award from the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] Supplementary material. Available from: `http://tbayes.eecs.umich.edu/xukevin/visualization_mlg_2011`.

[2] M. Baur and T. Schank. Dynamic graph drawing in Visone. Technical report, 2008.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. Wiley, 2006.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computat.*, 15(6):1373–1396, 2003.

[5] S. Bender-deMoll and D. A. McFarland. SoNIA - social network image animator. Available from: `http://www.stanford.edu/group/sonia/`.

[6] S. Bender-deMoll and D. A. McFarland. The art and science of dynamic network visualization. *J. Social Struct.*, 7(2):1–38, 2006.

[7] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling*. Springer, 2nd edition, 2005.

[8] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Inform. Visualiz.*, 2(1):40–50, 2003.

[9] J. A. Costa and A. O. Hero III. Classification constrained dimensionality reduction. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005.

[10] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proc. Nat. Acad. Sci.*, 106(36):15274–15278, 2009.

[11] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. Exploring the computing literature using temporal graph visualization. In *Proc. Conf. Visualiz. Data Anal.*, 2004.

[12] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Trans. Visualiz. Comput. Graphics*, 14(4):727–740, 2008.

[13] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[14] E. R. Gansner, Y. Hu, S. Kobourov, and C. Volinsky. Putting recommendations on the map – visualizing clusters and relations. In *Proc. 3rd ACM Int. Conf. Recommend. Sys.*, 2009.

[15] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proc. 12th Int. Symp. Graph Drawing*, 2004.

[16] I. Herman, G. Melançon, and M. S. Marshall. Graph visualisation and navigation in information visualisation: A survey. *IEEE Trans. Visualiz. Comput. Graphics*, 6(1):24–43, 2000.

[17] P. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[18] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31(12):7–15, 1989.

[19] Y. Koren. On spectral graph drawing. In *Proc. 9th Int. Comput. Combinat. Conf.*, 2003.

[20] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[21] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007.

[22] L. Leydesdorff and T. Schank. Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. *J. American Soc. Inform. Sci. Techol.*, 59(11):1810–1818, 2008.

[23] H. Lütkepohl. *Handbook of Matrices*. Wiley, 1997.

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

[25] MIT academic calendar 2004-2005. Available from: `http://web.mit.edu/registrar/www/calendar0405.html`.

[26] J. Moody, D. McFarland, and S. Bender-deMoll. Dynamic network visualization. *American J. Sociol.*, 110(4):1206–1241, 2005.

[27] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[28] T. M. Newcomb. *The acquaintance process*. Holt, Rinehart and Winston, 1961.

[29] P. G. Nordlie. *A longitudinal study of interpersonal attraction in a natural group setting*. PhD thesis, University of Michigan, 1958.

[30] J. Parkkinen, K. Nybo, J. Peltonen, and S. Kaski. Graph visualization with latent variable models. In *Proc. 8th Workshop Mining Learn. Graphs*, 2010.

[31] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proc. 14th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2008.

[32] D. M. Witten and R. Tibshirani. Supervised multidimensional scaling for visualization, classification, and bipartite ranking. *Computat. Statist. Data Anal.*, 55(1):789–801, 2011.

[33] K. S. Xu, M. Kliger, and A. O. Hero III. Adaptive evolutionary clustering. *Submitted*, 2011. Available from: `http://arxiv.org/abs/1104.1990`.