

A Maxent-Stress Model for Graph Layout

Emden R. Gansner, Yifan Hu, and Stephen North, *Member, IEEE*

Abstract—In some applications of graph visualization, input edges have associated target lengths. Dealing with these lengths is a challenge, especially for large graphs. Stress models are often employed in this situation. However, the traditional full stress model is not scalable due to its reliance on an initial all-pairs shortest path calculation. A number of fast approximation algorithms have been proposed. While they work well for some graphs, the results are less satisfactory on graphs of intrinsically high dimension, because some nodes may be placed too close together, or even share the same position. We propose a solution, called the maxent-stress model, which applies the principle of maximum entropy to cope with the extra degrees of freedom. We describe a force-augmented stress majorization algorithm that solves the maxent-stress model. Numerical results show that the algorithm scales well, and provides acceptable layouts for large, nonrigid graphs. This also has potential applications to scalable algorithms for statistical multidimensional scaling (MDS) with variable distances.

Index Terms—Graph drawing, metric embedding, low-dimensional embedding

1 INTRODUCTION

GRAPH drawing using virtual physical models on undirected graphs is among the most common methods of visualizing relationships between objects. This method has succeeded for several reasons: it can be applied to any undirected graph; it often conveys interesting graph properties, such as symmetry and clustering relationships; and some variants can be implemented by scalable methods.

Two virtual physical models are among the most popular. The spring-electrical model [11], [13] treats edges as springs that pull nodes together, and nodes as electrically charged particles that repel each other. High quality, efficient implementations have been proposed [17], [20], [40] based on a multilevel approach and fast force approximation within a suitable space decomposition scheme. They scale to graphs of millions of vertices and edges.

While the spring-electrical model has proven scalable and yields high-quality layouts, it has problems when edges have predefined target lengths. In the spring-electrical model, it is possible to encode edge lengths in the attractive and repulsive forces (e.g., [2, Section 10.1]), but such treatment is not rigorous.

In contrast, the (full) stress model assumes that there are springs connecting vertex pairs of the graph. The energy of this spring system is

$$\sum_{i,j \in V} w_{ij} (\|x_i - x_j\| - d_{ij})^2, \quad (1)$$

where we assume a graph $G = \{V, E\}$, with V the set of vertices and E the set of edges. In (1), d_{ij} is the ideal

distance between vertices i and j , and w_{ij} is a weight factor. A layout that minimizes this stress energy is taken as an optimal layout of the graph.

The stress model has its roots in multidimensional scaling (MDS) [28] which was eventually applied to graph drawing [25], [29]. Note that typically we are given only the ideal distance between vertices that share an edge, which is taken to be unit length for graphs without predefined edge lengths. For other vertex pairs, we define d_{ij} as the length of a shortest path between vertex i and j .

Related to the stress model is the strain model, also known as classical scaling. It relies on the fact that if the edge length can be achieved exactly using a set of node positions, then inner products of the positions can be expressed as the squared and double centered distances [5]. Based on this observation, node positions can be found by an eigendecomposition of a matrix. Because the strain model does not fit distances directly, graph layouts using this model are not as satisfactory as those using the stress model [5], but they can be used as a good starting point for the stress model.

In solving the full stress or strain model, the ideal distances between all pairs of vertices must be calculated. Johnson's algorithm [24] takes $O(|V|^2 \log |V| + |V||E|)$ time, and $O(|V|^2)$ memory. (A slightly faster, but still quadratic, algorithm is also known [34].) For large graphs, these models are computationally too expensive.

A number of strategies [10], [15] have been proposed to approximately minimize the stress or strain model. One notable effort is that of PivotMDS of Brandes and Pich [4]. This is a fast approximation algorithm for solving the strain model, which requires distance calculations from all nodes to only a few chosen nodes. Furthermore, Brandes and Pich suggested using this as a starting point for solving a sparse version of the stress model. The sparse model considers node pairs that are k or less hops away, where $k \ll |V|$. They showed this to be efficient and of good quality for many graphs. However, they noted that the algorithm tends to behave better on graphs of small dimension, and concluded that "further research on reliable sparsification

• The authors are with the AT&T Labs Research, Building 103, 180 Park Ave, Florham Park, NJ 07932.

E-mail: {erg, yifanhu, north}@research.att.com.

Manuscript received 31 May 2012; revised 14 Sept. 2012; accepted 22 Sept. 2012; published online 9 Oct. 2012.

Recommended for acceptance by H. Hauser, S. Kobourov, and H. Qu.

For information on obtaining reprints of this article, please send e-mail to: tvccg@computer.org, and reference IEEECS Log Number TVCGSI-2012-05-0099.

Digital Object Identifier no. 10.1109/TVCG.2012.299.

schemes is needed" [5]. This is because specifying only local distances for node pairs is not sufficient for avoiding the problem often observed in classical scaling and in the high-dimensional embedding algorithm [18], where in nonrigid graphs, such as trees, multiple nodes share the same position.

In this paper, we consider embedding graphs with specified nonunit edge lengths. The limitations of previous work, mentioned above, prompt us to seek an alternative algorithm that scales to large graphs, handles edge lengths well, and does not degrade on nonrigid graphs. Our motivation comes from the observation that, for graph drawing, the ideal distance of each edge is the only information given. To assume that the missing ideal distance between nonneighboring vertices should be the shortest graph-theoretic distance is reasonable, but does add artificial information that is not given in the input. And since it is not practical to calculate all-pairs shortest distances for large graphs anyway, we need some way to resolve the extra degrees of freedom in the node placement.

An interesting possibility is to apply a physics principle which states that, subject to known constraints, a system always settles to a state of maximum entropy. The maximum entropy principle provides the least-biased estimate possible on the given information, one that is "maximally noncommittal with regard to missing information" [23]. This principle is also believed to give rise to aesthetic beauty in nature. In the words of architect Lynn [30], whenever there is a lack of information, nature reverts to symmetry—"symmetry is the absence of information." This principle has been successfully applied in many areas of computational science, such as species distribution modeling [35] and natural language processing [3]. We propose that the same principle can be applied in graph drawing. An optimal layout should be one that attempts to satisfy the given ideal distances as much as possible. Since this itself is not sufficient to determine the layout of all nodes, the remaining degrees of freedom can be resolved through the principle of maximum entropy. We therefore propose a maximum entropy stress model (or maxent-stress model) for drawing graphs with edge distances specified.

The rest of the paper is organized as follows: In Section 2, we discuss related work. Section 3 gives the maxent-stress model, and a way of solving it. Section 4 evaluates our algorithm experimentally. Section 5 presents a summary and topics for further study.

Additional animated examples that illustrate the new algorithm can be found at the paper website <http://www2.research.att.com/~yifanhu/maxent>.

2 RELATED WORK

Embedding high-dimensional data to fit known distances has applications in many fields. Therefore, it is not a surprise that related work can be found not only in graph drawing, but also in other areas, such as machine learning.

2.1 Graph Drawing

Various techniques have been proposed for drawing large graphs with specified edge lengths. PivotMDS [4] was mentioned above. Another technique is to encode edge

lengths in a spring model, and solve it in some scalable manner. The GRIP [14] algorithm takes this approach. GRIP is a multilevel algorithm, with coarsening carried out by maximal independent set based filtration. On coarse levels, a Kamada-Kawai algorithm [25] is applied to each node within a local neighborhood of the original graph, but on the finest level, a localized Fruchterman-Reingold algorithm is used [13]. Because of this last step, the algorithm does not strictly solve a stress model.

Another approach was taken by Khoury et al. [26]. While we cope with the unknown distances between certain nodes by adding a complementary objective function, they showed how to utilize a low-rank SVD of the matrix consisting of distances from k -centers to all vertices to find approximate solutions for the Laplacian system involved in stress majorization. Their approach works efficiently for the special case of $w_{ij} = 1/d_{ij}$ in (1).

Although it essentially ignores edge lengths, the binary stress model of Koren and Cıvırlı [27] is stylistically related to ours, in that the first term attempts to specify edge lengths (as uniformly 0), the second term has the effect of uniformly spacing the nodes. Specifically, in that model, there is a distance of 0 between nodes sharing an edge, and a distance of 1 otherwise, given the model

$$\sum_{\{i,j\} \in E} \|x_i - x_j\|^2 + \alpha \sum_{\{i,j\} \notin E} (\|x_i - x_j\| - 1)^2.$$

Similarly, Noack [32], [33] has proposed the LinLog model and, more generally, the r -PolyLog model,

$$\sum_{\{i,j\} \in E} \|x_i - x_j\|^r - \sum_{i,j \in V} \ln \|x_i - x_j\|,$$

where, in particular, the second term is suggestive of our use of entropy.

There have been other suggestions for filling in missing node-node distance information. For example, Cohen [8] proposed an electrical resistance model for computing these distances. It takes into account multiple pathways between nodes, reducing distances between endpoints when there are more paths. Although this can be useful in emphasizing highly-connected clusters, it provides no help in efficiently computing a layout for large graphs.

2.2 Machine Learning and Dimensionality Reduction

The machine learning community has contributed a great deal of work recently on the related problem of dimension reduction for high-dimensional data. Unlike our problem, this assumes that distances between all data pairs are given, and the challenge is to efficiently create an embedding that preserves important local structure. Of particular note, Chen and Buja [7] use a force paradigm to define localized versions of MDS stress functions. They assume that local ideal distances are given for some pairs of items, and set the global distances to some very large value. Through algebraic manipulation, a stress energy model based on these distances is then converted into two terms:

$$\sum_{\{i,j\} \in S} (\|x_i - x_j\| - d_{ij})^2 - t \sum_{(i,j) \notin S} \|x_i - x_j\|, \quad (2)$$

the first term being the sparse stress energy, and the second the energy related to distances of node pairs. Here, item $\{i, j\}$ is in the set of vertex pairs S if j is among the k -nearest neighbors of i . They show that this model, LMDS, with careful tuning, outperforms a range of other methods, including MDS, on image clustering. Their work differs from ours in its motivation and in the energy model used. In addition, they applied LMDS only to relatively small examples of up to 1,956 nodes. Other than the selection of the parameter t , they do not give details on the implementation of the algorithm, so it is difficult to assess its scalability. While the motivation for LMDS is really data clustering, we implemented and tested a variant of LMDS but did not find it to be well-suited to graph layout (Section 4.5).

Chalmers [6] proposed the first linear-time iterative algorithm for dimensionality reduction in the context of visualization via stochastic sampling, and Ingram et al. [22] proposed Glimmer, a multiscale variant adapted to run efficiently with hardware GPU acceleration. In these works, a single entry of the distance matrix is assumed to be available in constant time, which is valid for multidimensional data, but not for graphs, where graph theoretical distances must be calculated.

2.3 Symmetry and Aesthetics

In Section 1, we argued that symmetry should be used to fill in the missing node-node distance information. Symmetry is a powerful visual property, and is easily recognized and remembered. It has long been studied in the context of graph drawing. Purchase [36] found that perceptual symmetry reduces response times or errors in various ways. Eades and Lin [12] proved that the solution of a “general spring model” can uncover symmetries. This model is very general and includes many energy models as special cases, including the full stress model and Eades’ force model [11].

3 A MAXIMAL ENTROPY STRESS MODEL

The full stress model assumes that there are springs connecting all vertex pairs of the graph, with the ideal spring lengths defined as the graph-theoretical distances between vertices. The energy of this spring system is given by (1), where d_{ij} is the graph-theoretical distance between vertices i and j , and w_{ij} is a weight factor, typically $1/d_{ij}^2$. A layout that minimizes the stress energy is an optimal layout of the graph according to this model.

As discussed in Section 1, the full stress model has high computational cost because distances between all node pairs must be calculated. We propose to fit only the given edge lengths via a sparse stress model, and to resolve the remaining degrees of freedom via maximization of the entropy of the layout. We denote this entropy $H(x)$. The model we propose is

$$\begin{aligned} & \max H(x) \\ & \text{subject to } \|x_i - x_j\| = d_{ij}, \{i, j\} \in S. \end{aligned} \quad (3)$$

Here, S is the set of vertex pairs that have predefined ideal distances. Typically, S will be the same as E , but could be a superset of E (e.g., the k -neighborhood graph).

This model may be infeasible because it is not usually possible to satisfy all distance constraints simultaneously. Therefore, as a compromise, we try to satisfy the constraints in (3) by minimizing the sum of the distance errors (also known as the sparse stress), while maximizing the entropy. In other words, we wish to solve

$$\min \sum_{\{i,j\} \in S} w_{ij} (\|x_i - x_j\| - d_{ij})^2 - \alpha H(x). \quad (4)$$

Here, $\alpha \geq 0$ is a parameter, and w_{ij} is a weighting factor. Large values of α favor maximizing the entropy, while small values put more emphasis on satisfying the ideal distances.

It remains to define more precisely the “entropy” of a layout. There are notions of point set entropy in image processing and graph entropy based on statistical attributes of nodes, but these do not seem applicable. Instead, we rely on an analogy from physics and consider nodes of a graph as objects in space. To maximize entropy without any constraints, these objects should be evenly dispersed, meaning that, on average, each node should be as far from other nodes as possible. Since some vertex pairs have a predefined ideal distance, we could define the entropy as

$$H(x) = \sum_{\{i,j\} \notin S} \ln \|x_i - x_j\|,$$

which will push vertex pairs as far apart as possible when maximized. Or, more generally, we can set

$$H(x) = -\text{sgn}(q) \sum_{\{i,j\} \notin S} \|x_i - x_j\|^{-q}, \quad q > -2, \quad (5)$$

where $\text{sgn}(q)$ is the sign of real number q . We denote $\|x\|^0 = -\ln \|x\|$, and assumes that $\text{sgn}(0) = 1$. The reason we restrict q to be greater than -2 is that if $q < -2$, the entropy term $H(x)$ will dominate the objective function in (4), and a layout that minimizes (4) is one where all vertices spread out infinitely away from each other. When $q = -2$, the same can happen when w_{ij} is small.

3.1 Force-Augmented Stress Majorization

Next, we turn to showing how the maxent-stress model (4) can be solved while avoiding the cost of the full stress model.

The minimum for (4) is achieved at a stationary point where the gradient vanishes. Thus, taking the derivative of the model with respect to x_i and setting it to zero gives

$$\begin{aligned} & \sum_{\{i,j\} \in S} 2w_{ij} (\|x_i - x_j\| - d_{ij}) \frac{x_i - x_j}{\|x_i - x_j\|} \\ & - \alpha \text{sgn}(q) \sum_{\{i,j\} \notin S} q \frac{x_i - x_j}{\|x_i - x_j\|^{q+2}} = 0, \end{aligned}$$

or, simplifying by setting $\alpha \leftarrow \alpha q/2$

$$\begin{aligned} \sum_{\{i,j\} \in S} w_{ij} (x_i - x_j) &= \sum_{\{i,j\} \in S} \frac{w_{ij} d_{ij} (x_i - x_j)}{\|x_i - x_j\|} \\ &+ \alpha \text{sgn}(q) \sum_{\{i,j\} \notin S} \frac{x_i - x_j}{\|x_i - x_j\|^{q+2}}. \end{aligned} \quad (6)$$

In matrix form, this is

$$L_w x = L_{w,d} x + \alpha b(x), \quad (7)$$

where the weighted Laplacian matrix L_w has elements

$$(L_w)_{ij} = \begin{cases} \sum_{\{i,l\} \in S} w_{il}, & \text{if } i = j \\ -w_{ij}, & \text{if } \{i,j\} \in S \\ 0, & \text{otherwise,} \end{cases}$$

the Laplacian matrix $L_{w,d}$ has elements

$$(L_{w,d})_{ij} = \begin{cases} \sum_{\{i,l\} \in S} w_{il} d_{il} \|x_i - x_l\|, & \text{if } i = j \\ -w_{ij} d_{ij} \|x_i - x_j\|, & \text{if } \{i,j\} \in S \\ 0, & \text{otherwise,} \end{cases}$$

and the vector $b(x)$ has elements

$$b(x)_i = \text{sgn}(q) \sum_{\{i,j\} \notin S} \|x_i - x_j\|^{-q-1} \frac{x_i - x_j}{\|x_i - x_j\|}. \quad (8)$$

Because of (7), the maxent-stress model can be solved in a way similar to stress majorization. This is akin to the Jacobi method, where we use the layout to calculate the right-hand side of (7), then solve the linear system (7) with the known right-hand side. Notice that if $\alpha = 0$, and S is the set of all node pairs, this reduces exactly to the stress majorization algorithm. Thus, the difference in the proposed method for solving the maxent-stress model is the term $b(x)$. Notice that when $q \geq 0$, the term $b(x)_i$, as defined in (8), is the sum of the repulsive forces from other nodes acting on node i , with the force proportional to $1/\|x_i - x_j\|^{q+1}$ along the direction from x_j to x_i . When $q < 0$, the term $b(x)_i$ is the sum of the attractive forces from other nodes acting on node i , with the force proportional to $1/\|x_i - x_j\|^{q+1}$ along the direction from x_i to x_j . For this reason we call this method force-augmented stress majorization. The fact that the augmented term is a sum of attractive forces when $q < 0$ also explains why the algorithm tends to have a clustering effect in this case (Section 4.5).

An alternative way to solve the maxent-stress model is

$$x_i \leftarrow \frac{1}{\rho_i} \sum_{\{i,j\} \in S} w_{ij} \left(x_j + d_{ij} \frac{x_i - x_j}{\|x_i - x_j\|} \right) + \frac{\alpha}{\rho_i} \text{sgn}(q) \sum_{\{i,j\} \notin S} \frac{x_i - x_j}{\|x_i - x_j\|^{q+2}}, \quad (9)$$

where $\rho_i = \sum_{\{i,j\} \in S} w_{ij}$. This simple iterative scheme, which updates the layout node-by-node, is useful for large or dynamic graphs where it would not be practical (or necessary) to solve the linear system accurately.

4 NUMERICAL RESULTS

We implemented the proposed force-augmented stress majorization algorithm using PivotMDS for the initial layout. Overall, the implementation is very similar to a stress majorization solver [15]. However, several further implementation details need to be resolved. First, the repulsive force term (8) involves an almost all-pairs computation, which has the $O(|V|^2)$ complexity that we wish to avoid. Second, we need to choose α in a way that is not dependent on the types or sizes of input graphs.

4.1 Repulsive Force Calculation

To reduce the complexity of the repulsive force calculation, we employ a Barnes-Hut approximation [1], [38], [37] to compute the repulsive forces (8) in $O(|V| \log |V|)$ time with good accuracy. This treats groups of distant vertices as supernodes, using a quadtree data structure (octree in 3D).

Because we approximate the repulsive forces, it may happen that these forces do not sum to zero. This makes the linear system (7) inconsistent because both Laplacians have a row sum of zero. Hence, after the fast force approximation, we normalize them to a sum of zero by adding a constant to the right-hand side.

4.2 Selection of Parameters

We use the conventional weighting factor of $w_{ij} = 1/d_{ij}^2$. For the repulsive force calculation, based on experiments, and following the implementation of `sfdp` (a multilevel force-directed algorithm [20] based on the spring-electrical model, available in `Graphviz` [16]), our implementation uses $q = 0$ except for a graph with many degree-1 nodes (more than 30 percent), where we set $q = 0.8$. The justification is that for such graphs, a weaker repulsive force helps to avoid a “warping effect” [21]. In the graphs tested, $q = 0.8$ only for `btrees` and `1138_bus`. An additional choice of q that resembles LMDS is also investigated and discussed in Section 4.5.

We also need to set the value of α . Since the goal of solving (4) is to satisfy the constraints in (3), we want α to be small. On the other hand, if α is too low initially, then we are essentially solving a conventional sparse stress model, with its problems in handling nonrigid graphs. Therefore, we start with a relatively large α , and gradually reduce it. To make sure that the repulsive force is properly scaled compared with the first term in the right-hand side of (6), we normalize the repulsive force vector $b(x)$ so that it has the same norm as that term. We experimented with several cooling schemes for α , and chose one that works well experimentally. Initially $\alpha = 1$, and we reduce it gradually with $\alpha \leftarrow 0.3 * \alpha$ in five steps, ending with $\alpha_{\min} = 0.008$.

Also, we limit the number of force-augmented stress majorization steps to 50 per setting of α . Thus, in the worst case, the maximum total number of stress majorization steps is 250. We solve the linear system in (7) using the conjugate gradient method, with a tolerance of 0.1 (relative residual), and maximum number of iterations of 10, since we found that it is not necessary to solve each of the intermediate linear systems exactly. We terminate the maxent-stress algorithm when the relative change in the layout, $\|x^{l+1} - x^l\|/\|x^l\|$, is less than 0.001, where x^l is the $2|V|$ -dimensional vector of coordinates for the 2D layout at iteration l of the stress majorization.

We implemented both PivotMDS and our force-augmented stress majorization algorithm in C, compiled with `gcc -O3`. The layout of PivotMDS is used as the initial layout for our algorithm. All results are measured on one core of a 16 core machine with Intel Xeon 2.13 GHz E5506 processors, and 12 GB of memory.

4.3 Experimental Results

We tested the force-augmented stress majorization algorithm for solving the maxent-stress model (hereafter

TABLE 1
Algorithms Tested

Algorithm	Model	Fits distances?
PivotMDS	approx. strain model	Yes/No
PivotMDS(k)	PivotMDS + sparse stress	Yes. k -hops
Maxent(k)	PivotMDS + maxent-stress	Yes. k -hops
sfdp	spring-electrical	No
GRIP	stress on coarser levels, spring-electrical on finest level	No
FSM	full stress model	Yes. All-pairs

denoted as *Maxent*) on a range of graphs. For comparison, we also tested PivotMDS, and PivotMDS with sparse stress majorization. We use PivotMDS(k) to denote PivotMDS, followed by sparse stress majorization on a graph consisting of the original edges, plus edges between vertex pairs of k hops or less. The ideal distance between a vertex pair is the length of the shortest path between them. We define Maxent(k) similarly. Thus, Maxent is essentially Maxent(1). We also consider sfdp as well as an implementation of the full stress model (FSM) that solves (1) using stress majorization. Finally, we include the GRIP multilevel algorithm. As GRIP does not really attempt to fit distances, we used a version that assumes unit distances. We summarize all the tested algorithms in Table 1.

With the exception of graph *gd*, which is an author collaboration graph of the International Symposium on Graph Drawing between 1994 and 2007, the graphs used are from the University of Florida Sparse Matrix Collection [9]. Our selection covers a range of graph sizes, and includes mesh-like and other nonmesh graphs, and graphs from Brandes and Pich's experimental study of distance scaling [5]. Two of the graphs (*commanche* and *luxembourg*) have associated predefined non-unit edge lengths. In our study, a rectangular matrix, or one with an asymmetric pattern, is treated as a bipartite graph. Test graph sizes are given in Table 2.

In the graph renderings, we use a red-to-green-to-blue color scale to encode edge lengths from short to long. Edges shorter than half of the median edge length are red, edges longer than 1.5 times the median are blue, and other edges are colored according to the scale.

We summarize drawings for all graphs tested in Table 3. Following Brandes and Pich [5], each drawing has an associated error chart. In an error chart, the x -axis is the graph distance, and ranges from smallest edge length to the diameter of the graph. It is divided into 31 bins, less if the graph diameter is smaller. The y -axis is the difference between the actual geometric distance in the layout for all pairs of vertices, and the graph distance. The chart shows the median (black line), the 25th and 75th percentiles (gray band) and the min/max errors (gray lines) that fall within each bin. For ease of understanding, we plot graph distance against distance error, instead of graph distance versus actual distance as suggested by Brandes and Pich [5]. Because generating the error chart requires an all-pairs shortest paths calculation, we provide this chart only for graphs $<10,000$ nodes.

While lengths of the edges is the only information specified in an input graph, and therefore it may appear

TABLE 2
Test Graphs

Graph	$ V $	$ E $	Description
<i>gd</i>	464	1311	Collaboration graph
<i>btree</i>	1023	1022	Binary tree
<i>1138_bus</i>	1138	1358	Power system
<i>qh882</i>	1764	3354	Quebec hydro power
<i>lp_ship041</i>	2526	6380	Linear programming
<i>USpowerGrid</i>	4941	6594	US power grid
<i>commanche*</i>	7920	11880	Helicopter
<i>bcsstk31</i>	35586	572913	Automobile component
<i>luxembourg*</i>	114599	119666	Luxembourg street map

Graphs marked * have prespecified nonunit edge lengths. Otherwise, unit edge length is assumed.

that we should only care about the first bins in an error chart, we note that how well we place vertices that are not directly connected to each other is vital to a good graph layout. The full stress model, which attempts to place vertices so that the distance between a pair of vertices in the layout is as close to their graph distance as possible, has been found to give good layouts, even though it has a high computational cost. The error chart gives us a detailed view of how well the layout of other methods approximate the graph distances.

In making the error charts, the layout is first scaled to minimize the full stress, with $w_{ij} = 1/d_{ij}^2$. In other words, we find a scalar s that minimizes

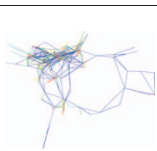
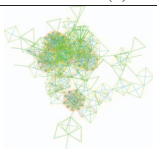
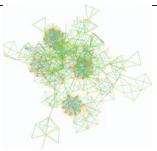
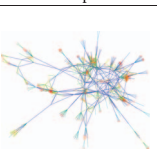
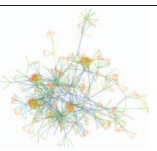
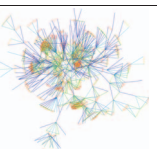
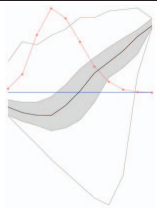
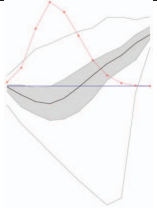
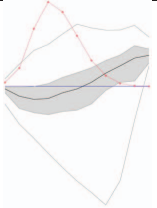
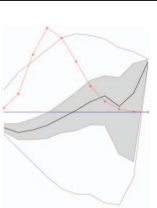
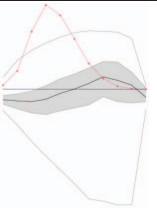
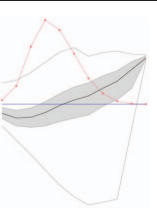
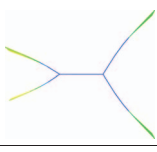
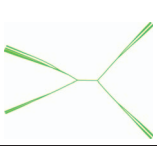
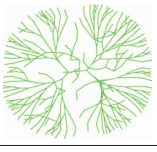
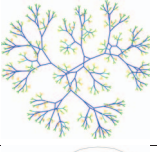
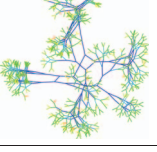
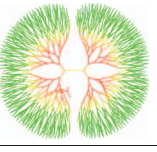
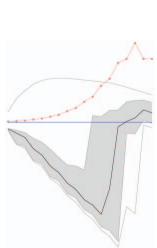
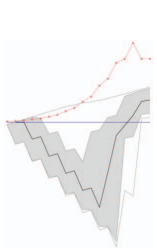
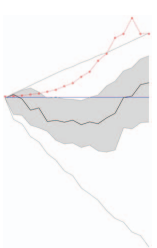
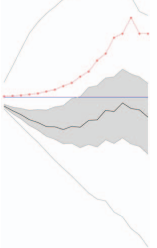
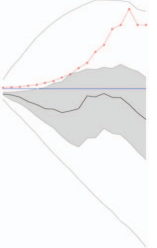
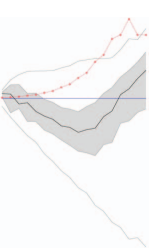
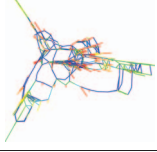
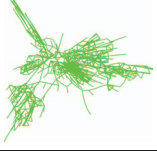
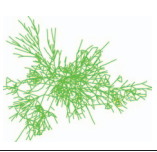
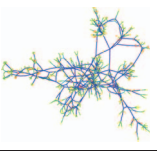
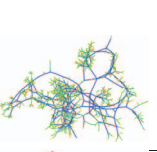
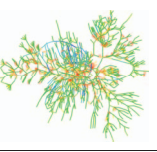
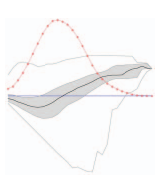
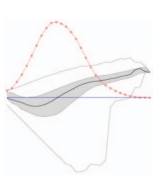
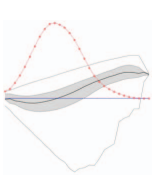
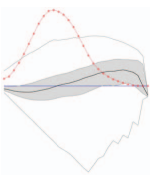
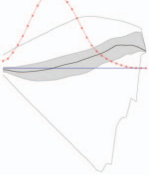
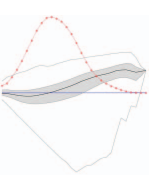


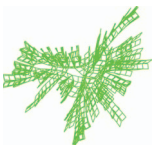

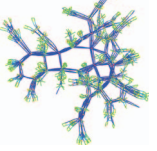
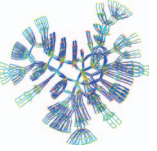
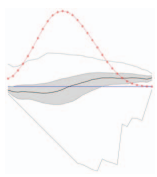
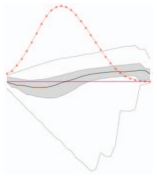
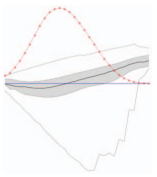
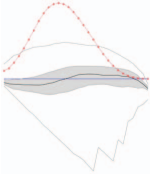
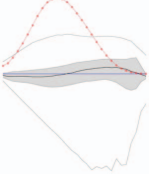
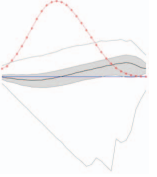
$$\sum_{i,j \in V} w_{ij} (s \|x_i - x_j\| - d_{ij})^2,$$

for an existing layout x . This is done to be fair to methods that do not try to fit layout distance to graph distance for all pairs of vertices; in addition it is necessary because sfdp does not utilize edge length at all. With the error chart, we include a graph distance distribution curve (red), representing the number of vertex pairs in each graph distance bin. This distribution depends on the graph, and is independent of the drawing.

As an example, the error chart for PivotMDS on *btree* (row 2, column 2) shows that, on average, the median line is under the x -axis for small graph distances. This means that the PivotMDS layout under-represents the graph distance between vertex pairs that are a few hops away. This is because it collapses branches of tree-like structures. The leaves of such structures tend to be a few hops away, but are now positioned very near to each other. To some extent the same under-representation of graph distance for vertex pairs that are a few hops away is seen for PivotMDS and PivotMDS(1) on other nonrigid graphs, including *1138_bus*, *btree*, *lp_ship041*, and *USpowerGrid*. Compared with PivotMDS and PivotMDS(1), the median line for Maxent (column 4) does not under-shoot the x -axes as much.

As a side note, these error charts are helpful in understanding the characteristics of other algorithms as well. For example, for most of the graphs, sfdp tends to under-represent vertex pairs with a high graph distance, seen as a dip of the median line past the x -axis for large x . This is likely due to the warping effect [21] of the spring-electrical model, where the length of edges in the layout are longer in the center of the graph and shorter around the periphery. The error charts for GRIP resemble those of sfdp

TABLE 3
Drawings and Error Charts of Algorithms

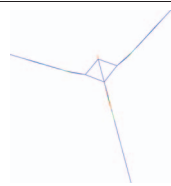

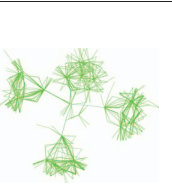
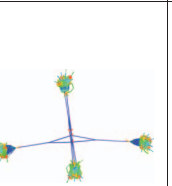
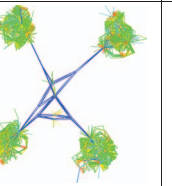
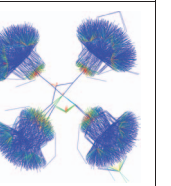
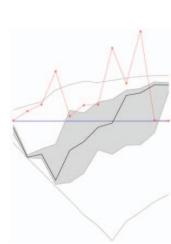
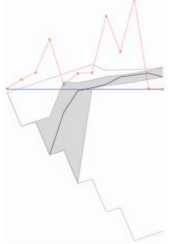
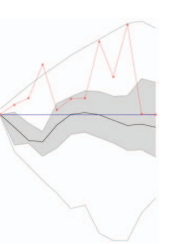
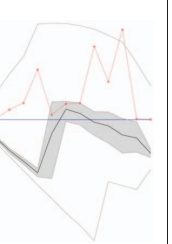
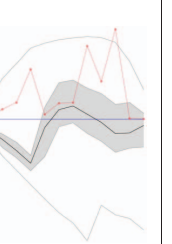
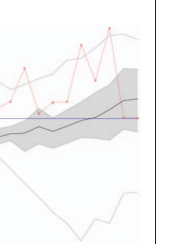
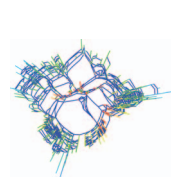

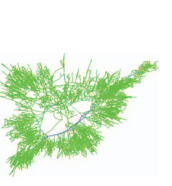
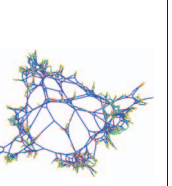
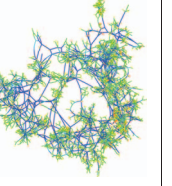
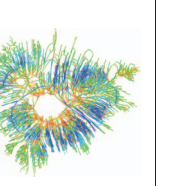
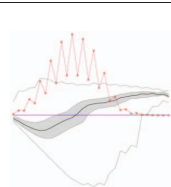
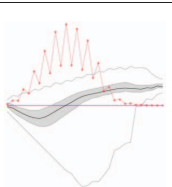
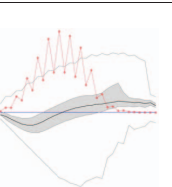
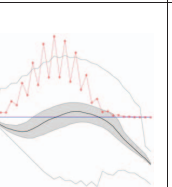
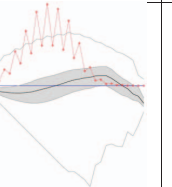
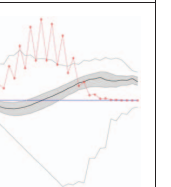
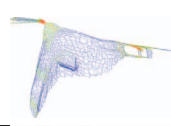
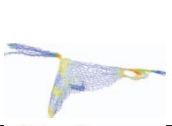
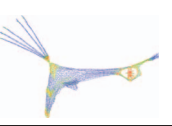
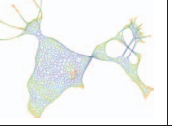
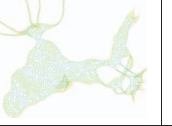

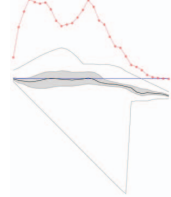
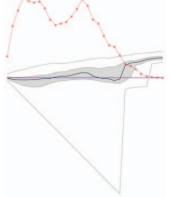
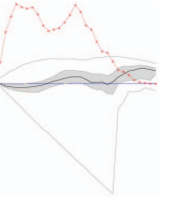
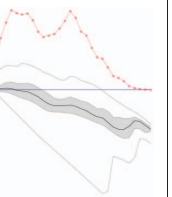
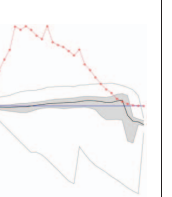
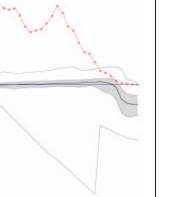
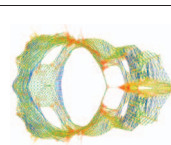
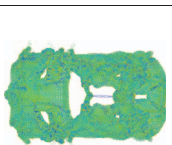
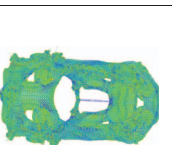
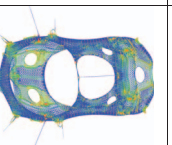
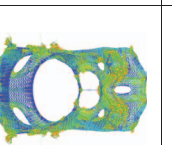
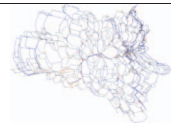
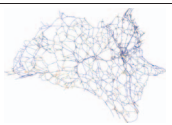
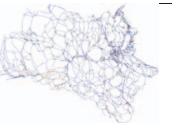
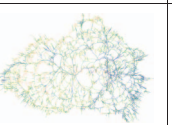
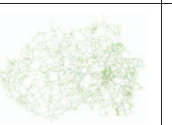
Graph	PivotMDS	PivotMDS(1)	Maxent	sfdp	GRIP	FSM
gd						
						
btree						
						
1138_bus						
						
qh882						
						

in many cases (e.g., btree and lp_ship041), presumably because GRIP applies the Fruchterman-Reingold algorithm on the finest level.

In the following, we highlight drawings for a few graphs. In Table 3, in the row for bcsst.k31, we see that PivotMDS(1) and Maxent give more or less the same layout, and both are

qualitatively not far from that of PivotMDS. This is expected because many graphs with an underlying mesh structure have a low intrinsic dimension, and PivotMDS alone can often give a good layout. Inspecting the color of edges in the drawings, we note that PivotMDS(1) and Maxent are dominated by green edges, indicating that the specified

TABLE 3
(continued)

Graph	PivotMDS	PivotMDS(1)	Maxent	sfdp	GRIP	FSM
lp_ship041						
						
USpowerGrid						
						
commanche						
						
bcsstk31						-
Luxembourg						-

In an error chart, X is the target distance bin, Y is the difference between layout distance and target distance. The chart shows median (black line), 25 and 75 percentile (gray band), and min/max errors (gray lines), as well as error distribution (red line). A limit of 10 CPU-hours was imposed and “-” denotes runs that did not finish within that time, or ran out of memory. In the drawings, a red-to-green-to-blue color palette is used to encode edge lengths from short to long.

edge length (in this case 1) is largely respected. PivotMDS has more edge length variation. For comparison, we see that sfdp produces a drawing with even more edge length variation, seen as regions of blue for long edges, and small regions of red for short edges.

Fig. 1 shows layouts for the 1138_bus graph. For this graph, PivotMDS collapsed many of the branches in the tree-like structures. This is a known problem with algorithms such as PivotMDS or high-dimensional embedding, for which “hairs” in tree-like structures cannot be differentiated by only

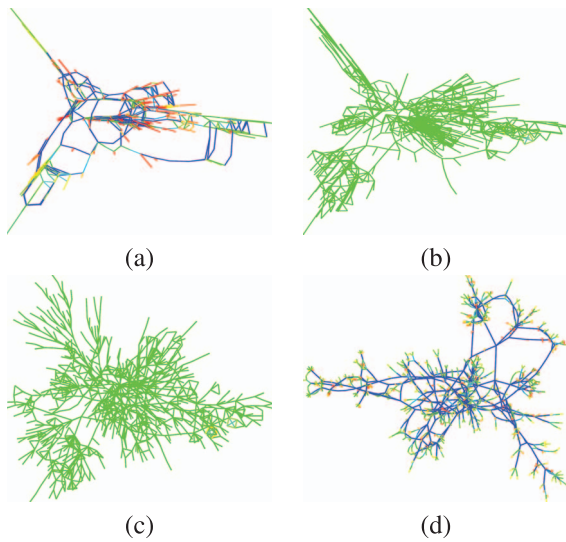


Fig. 1. Drawings by (a) PivotMDS, (b) PivotMDS(1), (c) Maxent, and (d) sfdp, on the 1138_bus graph.

considering distances from the k -centers. PivotMDS(1) expands some of the branches, but still shows a collapsing effect. Maxent expands these branches further, showing more details. Both PivotMDS(1) and Maxent provide more consistent target edge lengths, as indicated by the dominance of green, compared with the sfdp layout.

Fig. 2 shows layouts for the 1p_ship041 graph. The drawing by FSM, seen in Fig. 2a, emphasizes four clusters sparsely connected with each other. PivotMDS suffers from the same problem as with 1138_bus, where branches in the tree-like structures are collapsed, leaving only a skeleton of three arms. PivotMDS(1) does not do much better. PivotMDS(2) expands the clusters, because the ideal distance between nodes of up to 2 hops are now specified. Nevertheless, its drawing overlaps two clusters with each other, because to separate these would require specifying ideal distances of nodes many hops away. Compared to the corresponding PivotMDS(k) drawings, Maxent and Maxent(2) give better overall layouts of this graph.

So far, the graphs we have considered are without known coordinates. Fig. 3a shows a graph, *commanche*, with known coordinates, representing a helicopter. We can use the coordinates to compute edge lengths and see how well the graph can be regenerated. PivotMDS collapses the rotors of the helicopter. Both PivotMDS(1) and PivotMDS(2) have difficulty in separating the rotors. Maxent and Maxent(2) are able to show a better overall structure, although PivotMDS(2) gives more local details of the mesh.

Table 4 lists the CPU time used by these methods on a range of graphs. Maxent(k) usually takes more time than PivotMDS(k), because of its extra repulsive force calculation, but it still scales to relatively large graphs. On the largest graph, *luxembourg*, Maxent(k) is faster than PivotMDS(k), although not as fast as sfdp, showing that there is still a room for improvement in the implementation of the force-augmented maxent algorithm. Both sfdp and GRIP scale well to large graphs. For comparison, we also include the CPU time for FSM. Clearly, CPU time for FSM increases very quickly with graph size, and it does not scale to large graphs.

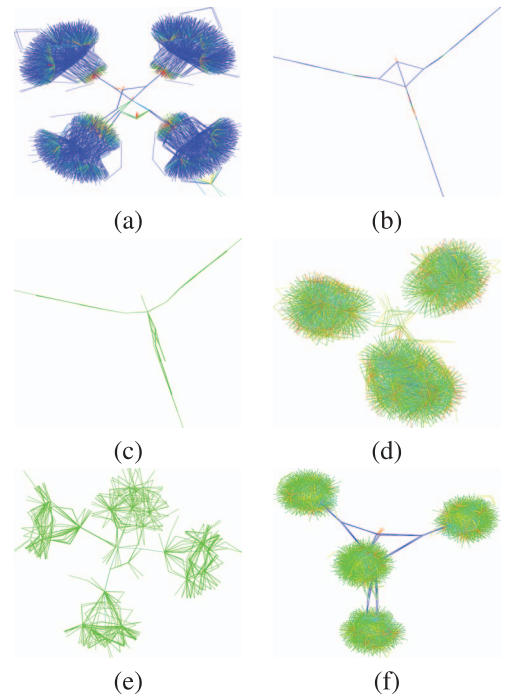


Fig. 2. Drawings by (a) FSM, (b) PivotMDS (c) PivotMDS(1), (d) PivotMDS(2), (e) Maxent, and (f) Maxent(2) on the 1p_ship041 graph.

4.4 Further Quantitative Evaluations

While visually comparing drawings made by different algorithms is informative, and may give an overall impression of the characteristics of each algorithm, such inspection is subjective. Ideally, we would prefer to rely on a quantitative measure of performance. However, such a measure is not easy to devise. For example, if we use sparse stress as our measure, PivotMDS(k), that minimizes sparse stress, is likely to come out best, despite its shortcomings. In the following, we attempt to measure layout quality using three quantitative measures: full stress, neighborhood

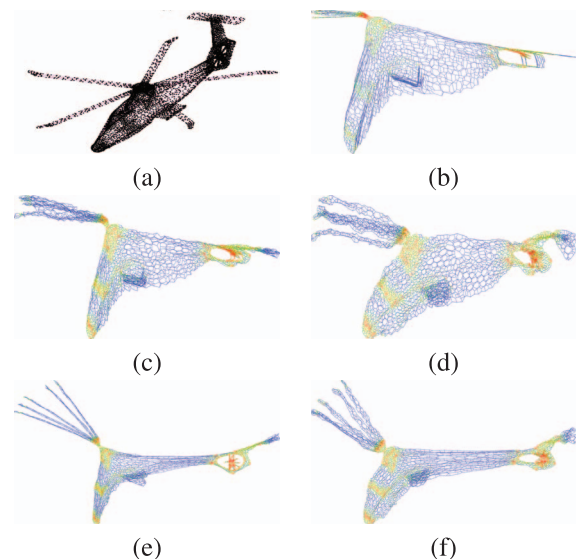


Fig. 3. Original graph (a), and drawings by (b) PivotMDS, (c) PivotMDS(1), (d) PivotMDS(2), (e) Maxent, and (f) Maxent(2) on the *commanche* graph.

TABLE 4
CPU Time (in Seconds) for PivotMDS, PivotMDS(1), PivotMDS(2), Maxent, Maxent(2), sfdp, and FSM

Graph	PivotMDS	PivotMDS(1)	PivotMDS(2)	Maxent	Maxent(2)	sfdp	GRIP	FSM
gd	0.3	0.3	0.5	0.8	1.1	0.2	0.1	2.3
btree	1.1	1.1	1.2	2.7	2.1	1	0.1	10
1138_bus	0.1	0.19	0.3	2.1	1.4	1.2	0.2	16
qh882	0.1	0.3	0.5	2.2	2.4	0.9	0.2	39
lp_ship041	0.1	0.1	2.2	2.2	8.2	2.0	0.2	58
USpowerGrid	0.1	0.9	1.4	6.5	6.4	3.7	0.4	272
commanche	0.2	0.9	5.0	9.0	10.5	5.6	0.7	1025
bcsstk31	2.4	21.6	71.0	102	258	23.8	22	-
luxembourg	2.4	630	806	209	266	78.7	66.9	-

A limit of 10 CPU-hours is imposed and “-” is used to denote runs that could not finish within that time, or ran out of memory.

inconsistency, and precision of neighborhood preservation. The latter two measures compare the neighborhood structure in layout space with that in the original graph space.

4.4.1 Measuring Full Stress

One imperfect measure of quality is full stress, as defined by (1), with $w_{ij} = 1/d_{ij}^2$. This measures how the embedding fits the graph theoretical distances. Notice that with $w_{ij} = 1/d_{ij}^2$ the full stress (1) can be written as

$$\sum_{i,j \in V} (\|x_i - x_j\|/d_{ij} - 1)^2. \quad (10)$$

Therefore, the full stress is a measure of how the ratio between euclidean and graph theoretical distances deviate from 1. Note that this penalizes the case when $\|x_i - x_j\| \gg d_{ij}$, but under-penalizes the case when $\|x_i - x_j\| \ll d_{ij}$ (because the most a $\|x_i - x_j\| = 0$ term can contribute to the stress is 1). Furthermore, this measure naturally favors the full stress model based algorithm FSM.

Table 5 gives the full stress measure achieved by each algorithm. In addition, it gives a score for each method (last row, smaller is better) defined as the average ratio between the full stress and the smallest stress. Because it is expensive to calculate all-pairs shortest paths, we restrict experimental measurement to graphs with fewer than 10,000 nodes. From the table we can see that, as expected, FSM is the best, because it tries to optimize this measure. PivotMDS and sfdp end up with higher full stress. As expected, sfdp yields high full stress on commanche, because it assumes unit edge length on a nonunit edge length input graph. Surprisingly, GRIP gives the second smallest full stress among all algorithms.

Maxent often gives lower full stress than PivotMDS(1), although it is not clear why Maxent(2) tends to gives higher full stress than Maxent. For example, on commanche, Maxent(2) gives full stress that is almost 60 percent higher than Maxent, but the drawing given by Maxent(2) (Fig. 3f) does not seem any worse than that of Maxent (Fig. 3e). We conjecture that because we use the maximal entropy principle to deal with the extra degrees of freedom, the edge length information is all it takes for Maxent to work well, and it is not necessary to have additional information for pairs of vertices that do not form edges. On the other hand, PivotMDS clearly benefits from such extra information.

4.4.2 Measuring Neighborhood Inconsistency

Often, in embedding high-dimensional data into a lower dimension, one is interested in preserving the neighborhood structure. In such a situation, replicating the distance between objects becomes a secondary concern.

As an example, imagine a graph where each node is a movie. Based on some recommender algorithm, an edge is added between two movies if the algorithm predicts that a user who likes one movie would also like the other, with the length of the edge defined as the distance (dissimilarity) between the two movies. The graph is sparse because only movies that are strongly similar are connected by an edge. For a visualization of this data to be helpful, we need to embed this graph in 2D in such a way that, for each node (movie), nodes in its neighborhood in the layout are very likely to be similar to this node. This would allow the user to explore movies that are more likely of interest to her by examining, in the visualization, the neighborhoods of the movies she knew and liked.

Because the recommender algorithm may not be accurate, even if two movies are not directly connected, there is

TABLE 5
Full Stress Measure for PivotMDS, PivotMDS(1), PivotMDS(2), Maxent, Maxent(2), sfdp, and FSM

Graph	PivotMDS	PivotMDS(1)	PivotMDS(2)	Maxent	Maxent(2)	sfdp	GRIP	FSM
gd	19384	15073	15506	12327	13011	12752	11051	9734
btree	130190	109713	74946	63524	70260	75518	82341	60226
1138_bus	77834	64630	56368	44797	50533	56225	51668	40030
qh882	147114	119615	125694	102654	104139	125999	105382	84477
lp_ship041	666532	769495	436130	363024	469808	632422	458384	250707
USpowerGrid	1123582	932395	892356	1017798	923927	1156193	1038001	701831
commanche	2305010	1547432	1203406	1545418	2464612	3690581	967418	653869
avg. ratio/best	2.23	1.88	1.51	1.41	1.71	2.18	1.40	1

The last row gives the average ratio over the best, defined as the stress divided by the smallest stress for all methods, averaged over seven graphs. Smaller is better.

TABLE 6
Neighborhood Inconsistency (NI) for PivotMDS, PivotMDS(1), PivotMDS(2), Maxent, sfdp, and FSM

Graph	PivotMDS	PivotMDS(1)	PivotMDS(2)	Maxent	sfdp	GRIP	FSM
gd	2627	2108	2676	2227	2137	2441	2655
btree	4816	4795	5585	4083	3096	5209	9663
1138_bus	4879	5246	5952	3786	2988	4222	3536
qh882	8813	10186	13238	8499	5856	7608	6538
lp_ship041	17600	39734	38554	16944	17374	28589	18597
USpowerGrid	31511	37778	46249	32721	21912	43236	33190
commanche	210131	200682	189179	148337	132389	126402	134316
avg. ratio/best	1.44	1.67	1.89	1.25	1.01	1.46	1.48

The last row gives the average ratio over the best, defined as the NI of a method divided by the smallest NI for all methods, averaged over seven graphs. Smaller is better.

still a probability that they are similar, except that the probability diminishes if they are many hops away. Therefore, for each movie, we have a probability distribution where every other movie is given a probability of being similar to it. In the embedding space, we can also define a probability distribution based on how far other movies are from this movie, in euclidean distance. An optimal embedding is one that matches these two distributions as closely as possible. This approach was first proposed by Hinton and Roweis [19], and is described formally as follows.

In the graph space, we define the probability that vertex j is similar to vertex i as

$$p_{ij} = e^{-\frac{d_{ij}^2}{\sigma_i}}, j \neq i. \quad (11)$$

Here, we take d_{ij} to be the graph-theoretical distance between the two nodes, and σ_i is a scaling parameter dictating how quickly the probability decays with the distance. The probability is further normalized so that $\sum_{j:j \neq i} p_{ij} = 1$.

In the embedding space, in a similar fashion, we define the probability that vertex j is a neighbor of vertex i as

$$q_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\sigma_i}}, j \neq i. \quad (12)$$

A good embedding should make these two probability distributions match as closely as possible. The difference between these two distributions can be measured using the following symmetrized Kullback-Leibler (KL) divergence (Note that Hinton and Roweis [19] originally proposed using the asymmetric version of the KL divergence, but subsequent work [31], [39] used the symmetrized version.)

$$NI = 0.5 \sum_i \sum_j \left(p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + q_{ij} \log \left(\frac{q_{ij}}{p_{ij}} \right) \right). \quad (13)$$

We call this quantity the *neighborhood inconsistency* (NI). The smaller the value, the better the embedding matches the original data in terms of neighborhood structure. We give the NI measure of the algorithms for graphs with less than 10,000 nodes in Table 6. The scaling parameter σ_i has to be chosen carefully. If it is set too small, then larger values of d_{ij} would make (11) negligible, therefore the NI measure would effectively only consider a few closest neighbors. On the other hand, a tiny value of σ_i would make far away neighbors equally important as close neighbors. If we assume a simple case of K closest neighbors of being similar to i with equal probability $1/K$,

and the rest with probability zero, then the entropy for the distribution $\{p_{ij} | j : j \neq i\}$ is $\log(K)$. Therefore, we choose σ_i so that the entropy of the probability distribution equals $\log K$. Here, K is known as the perplexity, or the effective number of local neighbors. Following Hinton and Roweis [19], who use K in the range of 10-25, we set $K = 20$.

For ease of comparison, in the last row of Table 6 we also give a score for each method (smaller is better), defined as the ratio between the NI for the method on a graph, and the smallest NI over all methods for that graph, averaged over all graphs. From Table 6, we can see that PivotMDS sometimes has smaller NI than PivotMDS(1), probably because NI is an objective function that, to some extent, characterizes how well the embedding depicts the cluster structure of the data, and is used in the field of machine learning for the purpose of visual classification. Often, eigenvector-based layout methods, such as PivotMDS, without further refinement, characterize the cluster structures well, but with a lack of details (multiple vertices may be placed at the same location). According to NI, Maxent is generally better than PivotMDS and PivotMDS(1), and comparable to FSM and GRIP. Surprisingly, sfdp has the smallest or the second smallest NI measures. Again, we hypothesize that sfdp performs well due to the fact force-directed algorithms such as sfdp tend to show cluster structures well [33].

4.4.3 Measuring Precision of Neighborhood Preservation

While the NI measure (13) is computationally simple and gives us a single number for comparing neighborhood inconsistency, it is not easy to interpret intuitively. To help us to understand and validate NI, in the following we look at the *precision of neighborhood preservation*, a more concrete measure.

We are interested in answering the question: if we see vertices nearby in the embedding, how many of these are actually also neighbors in the graph space? We define the precision of neighborhood preservation as follows: For each vertex i , K neighboring vertices of i in the layout are chosen. These K vertices are then checked to see if their graph distance is less than a threshold $d(K)$, where $d(K)$ is the distance of the K th closest vertex to i in the graph space. The percentage of the K vertices that are within the threshold, averaged over all vertices i , is taken as the precision. Note that precision (the fraction of retrieved instances that are relevant) is a well-known concept in information retrieval. Chen and Buja [7] use a similar concept called *LC metacriteria*.

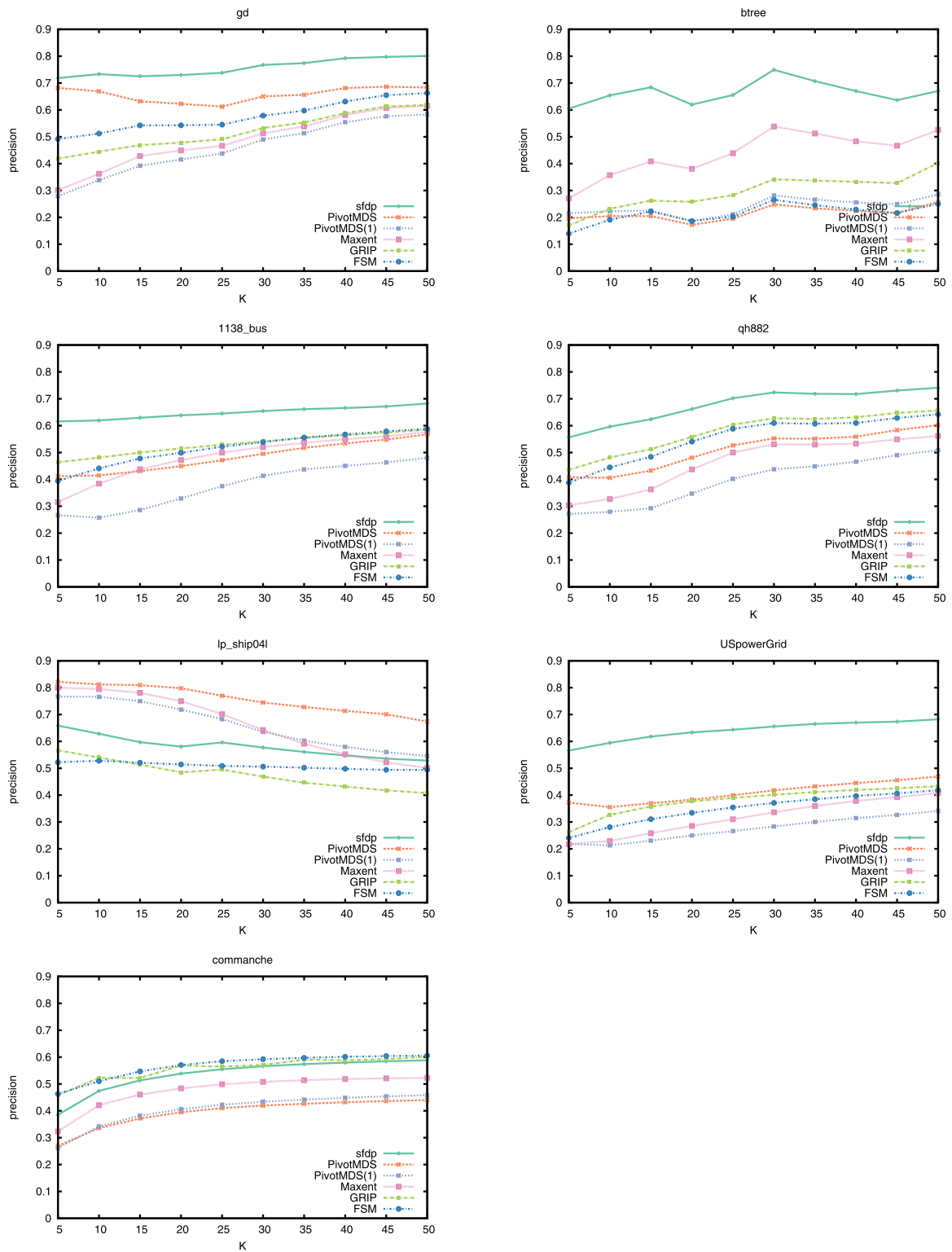


Fig. 4. Precision of neighborhood preservation of the algorithms, as a function of K . For each vertex i , K -nearest neighbors of vertex i in the layout is chosen. These K vertices are then checked to see if their graph distance is less than a threshold $d(K)$, where $d(K)$ is the graph distance of the K th closest vertex to i in the graph space. The percentage of the K vertices that are within the threshold, averaged over all vertices i , is taken as the precision. The higher the precision, the better.

Fig. 4 gives the precision as a function of K . From the figure, it is seen that, in general, sfdp has the highest, or nearly the highest precision, except for lp_ship041. This is consistent with our findings when using NI as a measure of neighborhood quality. PivotMDS(1) tends to have low precision. The precision of other algorithms,

including Maxent, tends to be between these two extremes. One outlier is lp_ship041. For this case, PivotMDS(1) has high precision. In light of the drawing of this graph in Fig. 2, this may be related to the fact that the drawing by PivotMDS(1), like that of PivotMDS, collapses the clusters.

FSM has low precision for *btree*. We believe this can be explained when looking at the drawings for *btree* in Fig. 3. The FSM drawing utilizes space well, but each leaf i of the tree tends to be very close to the leaves of other branches. These leaves have a high graph distance to i , making the precision lower. This is the result of FSM under-penalizing the under-prediction of large graph distances, as discussed before.

Overall, neighborhood inconsistency and precision of neighborhood preservation are two quantitative ways in which to gauge aspects of the algorithms not well captured by the full stress objective function. We believe they offer useful and complementary measures of layout quality for distance-based embedding. Based on these, and the full stress objective function, Maxent is found to offer a compromise between distance preservation and neighborhood preservation, with the advantages of being more scalable than the full stress model, and of giving better drawings and neighborhood preservation than PivotMDS(k) for nonrigid graphs.

4.5 The Effect of the q Parameter

As explained in Section 4.2, in our implementation we use $q = 0$ except for graphs with many degree-1 nodes, where we set $q = 0.8$. We note that if $q = -1$, the entropy function (5) becomes

$$H(x) = \sum_{\{i,j\} \notin S} \|x_i - x_j\|, \quad (14)$$

and the repulsive force equation (8) becomes

$$b(x)_i = \sum_{\{i,j\} \notin S} \frac{x_i - x_j}{\|x_i - x_j\|}. \quad (15)$$

With (14), our energy model (4) is essentially the same as LMDS of Chen and Buja [7] (2), except that our solution procedure calls for annealing α gradually to zero, instead of a fixed α . In addition, we can still utilize the Barnes-Hut approximation to efficiently compute the sum of the unit vectors in (15). Because of this resemblance to LMDS, we are interested to see whether this choice of q gives us a good graph drawing algorithm.

We call the algorithm that corresponds to $q = -1$ Maxent(1, -1). Fig. 5 gives the drawing for some graphs using Maxent(1, -1). It seems that because of the stronger repulsive forces in (15) that do not decay with the distance, nodes can cling to each other, and Maxent(1, -1) gives drawings that resemble those of PivotMDS and PivotMDS(k). For example, its drawing of the *btree* makes the branches stay close to each other, just like the drawings by PivotMDS and PivotMDS(k) in Fig. 3. This strong clustering effect may be desirable for the purpose of classification in machine learning applications, but makes it harder to differentiate nodes and edges for the purpose of graph visualization. In general, we found that a smaller q value (e.g., $q < 0$) increases clustering effect at the expense of local details. A very large q value (e.g., $q \gg 0$) increase fitting of distances along edges, at the expense of obfuscating global structure.

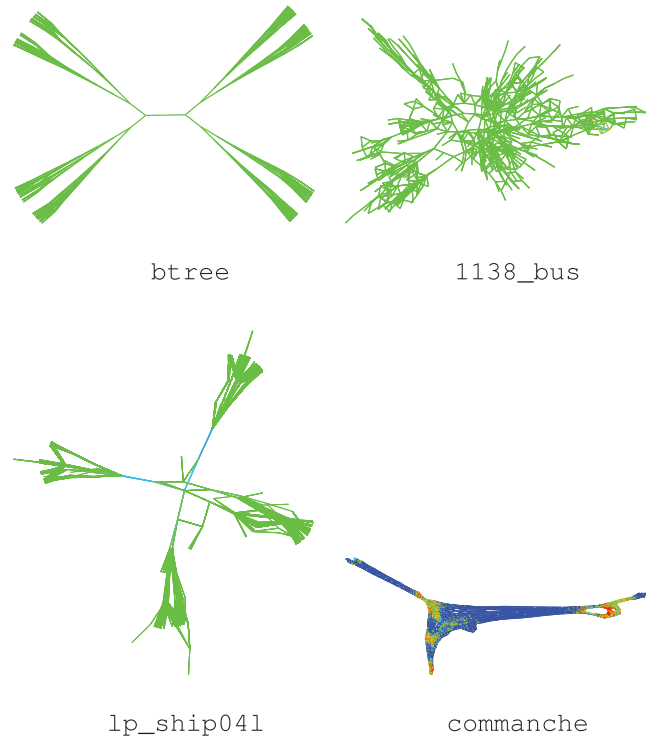


Fig. 5. Drawings by Maxent(1, -1).

5 DISCUSSION AND CONCLUSIONS

This paper proposed the maxent-stress model for graph embedding, with the objective of satisfying input edge lengths, while resolving the remaining degrees of freedom with the principle of maximal entropy. The proposed method does not require an all-pairs shortest path calculation, as needed by the full stress model, and is therefore more scalable. Compared with other scalable stress models such as PivotMDS, the proposed method also does not degrade as much on nonrigid graphs.

We introduced the concepts of neighborhood inconsistency and precision of neighborhood preservation to gauge aspects of the algorithms not well captured by the full stress. We believe these are useful additional measures of layout quality for distance-based embedding. Based on these measures, the maxent-stress model is found to offer a compromise between distance preservation and neighborhood preservation.

The maxent-stress model incorporates a parameter α that controls the strength of the repulsive forces. During iterative solution, this parameter should gradually be reduced toward zero. We proposed a scheme to reduce it geometrically. We are still experimenting with refinements to this schedule. It is also not clear how to analyze the convergence of the maxent-stress algorithm. Ideally, we would like to solve the true maximum entropy model (3) by making distance satisfaction the top priority, even though (3) contains constraints that may be infeasible.

While we applied the force-augmented stress majorization algorithm proposed in this paper to solve the maxent-stress model, another potential way to solve the model is using a variant of a pure force-directed algorithm (9).

Embedding high-dimensional data to fit known distances has potential applications not only in graph drawing, but also in machine learning. We would like to investigate the use of the proposed model in such problems, and compare it with established approaches such as LLE and Isomap.

ACKNOWLEDGMENTS

The authors thank Ulrik Brandes for his PacificVis 2011 lecture on PivotMDS, and subsequent discussions. They are also grateful to the anonymous reviewers whose comments helped us to significantly improve the paper. A preliminary version of this paper appeared in the Proceedings of the fifth IEEE PacificVis Symposium.

REFERENCES

- [1] J. Barnes and P. Hut, "A Hierarchical O(NlogN) Force-Calculation Algorithm," *Nature*, vol. 324, pp. 446-449, 1986.
- [2] G.D. Battista, P. Eades, R. Tamassia, and I.G. Tollis, *Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
- [3] A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra, "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39-71, 1996.
- [4] U. Brandes and C. Pich, "Eigensolver Methods for Progressive Multidimensional Scaling of Large Data," *Proc. 14th Int'l Symp. Graph Drawing (GD '06)*, pp. 42-53, 2007.
- [5] U. Brandes and C. Pich, "An Experimental Study on Distance Based Graph Drawing," *Proc. 16th Int'l. Symp. Graph Drawing (GD '08)*, pp. 218-229, 2009.
- [6] M. Chalmers, "A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data," *Proc. Seventh Conf. Visualization (VIS '96)*, pp. 127-131, 1996.
- [7] L. Chen and A. Buja, "Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Drawing, and Proximity Analysis," *J. Am. Statistical Assoc.*, vol. 104, pp. 209-219, 2009.
- [8] J.D. Cohen, "Drawing Graphs to Convey Proximity: An Incremental Arrangement Method," *ACM Trans. Computer-Human Interaction*, vol. 4, no. 3, pp. 197-229, <http://www.cise.ufl.edu/research/sparse/matrices/>, Sept. 1997.
- [9] T.A. Davis and Y.F. Hu, "University of Florida Sparse Matrix Collection," *ACM Trans. Math. Software*, vol. 38, pp. 1-18, 2011.
- [10] V. de Silva and J.B. Tenenbaum, "Global versus Local Methods in Nonlinear Dimensionality Reduction," *Proc. Advances in Neural Information Processing Systems 15*, pp. 721-728, 2003.
- [11] P. Eades, "A Heuristic for Graph Drawing," *Congressus Numerantium*, vol. 42, pp. 149-160, 1984.
- [12] P. Eades and X. Lin, "Spring Algorithms and Symmetry," *Proc. Int'l Conf. Computing and Combinatorics (COCOON)*, pp. 202-211, 1997.
- [13] T.M.J. Fruchterman and E.M. Reingold, "Graph Drawing by Force Directed Placement," *Software - Practice and Experience*, vol. 21, pp. 1129-1164, 1991.
- [14] P. Gajer, M.T. Goodrich, and S.G. Kobourov, "A Fast Multidimensional Algorithm for Drawing Large Graphs," vol. 1984, pp. 211-221, 2000.
- [15] E.R. Gansner, Y. Koren, and S.C. North, "Graph Drawing by Stress Majorization," *Proc. 12th Int'l. Symp. Graph Drawing (GD '04)*, pp. 239-250, 2004.
- [16] E.R. Gansner and S. North, "An Open Graph Visualization System and Its Applications to Software Engineering," *Software - Practice & Experience*, vol. 30, pp. 1203-1233, 2000.
- [17] S. Hachul and M. Jünger, "Drawing Large Graphs with a Potential Field Based Multilevel Algorithm," *Proc. 12th Int'l Symp. Graph Drawing (GD '04)*, pp. 285-295, 2004.
- [18] D. Harel and Y. Koren, "Graph Drawing by High-Dimensional Embedding," *J. Graph Algorithms and Applications*, vol. 8, no. 2, pp. 195-214, 2004.
- [19] G.E. Hinton and S.T. Roweis, "Stochastic Neighbor Embedding," *Proc. Neural Information Processing Systems (NIPS)*, pp. 833-840, S. Becker, S. Thrun, and K. Obermayer, eds., 2002.
- [20] Y.F. Hu, "Efficient and High Quality Force-Directed Graph Drawing," *Math. J.*, vol. 10, pp. 37-71, 2005.
- [21] Y.F. Hu and Y. Koren, "Extending the Spring-Electrical Model to Overcome Warping Effects," *Proc. IEEE Pacific Visualization Symp.*, pp. 129-136, 2009.
- [22] S. Ingram, T. Munzner, and M. Olano, "Glimmer: Multilevel MDS on the GPU," *IEEE Trans. Visualization Computer Graphics*, vol. 15, no. 2, pp. 249-261, Mar./Apr. 2009.
- [23] E.T. Jaynes, "Information Theory and Statistical Mechanics," *Physical Rev. Series II*, vol. 106, no. 4, pp. 620-630, 1957.
- [24] D.B. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," *J. ACM*, vol. 24, no. 1, pp. 1-13, Jan. 1977.
- [25] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters*, vol. 31, pp. 7-15, 1989.
- [26] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger, "Drawing Large Graphs by Low-Rank Stress Majorization," *Proc. Eurographics/IEEE TVCG Symp. Visualization*, 2012.
- [27] Y. Koren and A. Çivril, "The Binary Stress Model for Graph Drawing," *Proc. 16th Int'l Symp. Graph Drawing (GD '08)*, pp. 193-205, 2008.
- [28] J.B. Kruskal, "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis," *Psychometrika*, vol. 29, pp. 1-27, 1964.
- [29] J.B. Kruskal and J.B. Seery, "Designing Network Diagrams," *Proc. First General Conf. Social Graphics*, pp. 22-50, July 1980.
- [30] G. Lynn, "Calculus in Architecture," http://www.ted.com/index.php/talks/greg_lynn_on_organic_design.html, 2005.
- [31] L.v. Maaten and G. Hinton, "Visualizing Data Using t-sne," *J. Machine Learning Research*, vol. 9, pp. 2579-2605, 2008.
- [32] A. Noack, "Energy Models for Graph Clustering," *J. Graph Algorithms and Applications*, vol. 11, pp. 453-480, 2007.
- [33] A. Noack, "Modularity Clustering is Force-Directed Layout," *Physical Rev. E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 79, 2009.
- [34] S. Pettie, "A New Approach to All-Pairs Shortest Paths on Real-Weighted Graphs," *Theoretical Computer Science*, vol. 312, no. 1, pp. 47-74, Jan. 2004.
- [35] S.J. Phillips, R.P. Anderson, and R.E. Schapire, "Maximum Entropy Modeling of Species Geographic Distributions," *Ecological Modelling*, vol. 190, nos. 3/4, pp. 231-259, 2006.
- [36] H.C. Purchase, "Which Aesthetic Has the Greatest Effect on Human Understanding?" *Proc. Fifth Int'l Symp. Graph Drawing (GD '97)*, pp. 248-261, 1997.
- [37] A. Quigley, "Large Scale Relational Information Visualization, Clustering, and Abstraction," PhD thesis, Dept. of Computer Science and Software Eng., Univ. Newcastle, Australia, 2001.
- [38] D. Tunkelang, "A Numerical Optimization Approach to General Graph Drawing," PhD thesis, Carnegie Mellon Univ., 1999.
- [39] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization," *J. Machine Learning Research*, vol. 11, pp. 451-490, 2010.
- [40] C. Walshaw, "A Multilevel Algorithm for Force-Directed Graph Drawing," *J. Graph Algorithms and Applications*, vol. 7, pp. 253-285, 2003.



Emden R. Gansner received the PhD degree in mathematics from MIT in 1978. He joined Bell Laboratories in 1980 and was a distinguished member of Technical Staff in the Software Engineering Research Department. He is currently a lead member of Technical Staff in the Information Visualization Research Department at AT&T Labs Research. He is an author of the Graphviz graph drawing system. His research interests include graphs (drawing, theory, algorithms), information visualization, graphical user interfaces, and algebraic combinatorics.



Yifan Hu received the BS and MS degrees in applied mathematics from Shanghai Jiao-Tong University, in 1981 and 1985, and the PhD degree in optimization from Loughborough University, United Kingdom, in 1992. He is a principal member of Technical Staff in the Information Visualization Department at AT&T Labs Research. He is a contributor to the Graphviz graph drawing system. His research interests include numerical and combinatorial

algorithms, information visualization, and data mining.



Stephen North received the PhD degree in computer science from Princeton University. He is the director of Information Visualization Research at AT&T Labs Research in Florham Park, New Jersey, and the AT&T Fellow since 2008. He has served as a program chair or co-chair of IEEE Info Vis, IEEE Pacific Visualization, IEEE VizSEC, and the International Graph Drawing Symposium. He is interested in scale, automation, and applied algorithms in informa-

tion visualization. He is one of the authors of Graphviz. He is a member of the IEEE and the ACM.

► **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.**