A Maxent-Stress Model for Graph Layout

Emden R. Gansner[*]
AT&T Labs – Research
Florham Park, NJ, USA

Yifan Hu[†]
AT&T Labs - Research,
Florham Park, NJ, USA

Stephen North[‡]
AT&T Labs - Research,
Florham Park, NJ, USA

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 nodes overlap unnecessarily. 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, non-rigid graphs. This also has potential applications to scalable algorithms for statistical multidimensional scaling (MDS) with variable distances.

Keywords: graph drawing; distance scaling; low-dimensional embedding

Index Terms: G.2 [Discrete Mathematics]: Graph Theory—; I.3.3 [Computer Graphics]: Picture/Image Generation—Line and Curve Generation

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 certain interesting properties of graphs, 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 [9, 11] 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 [15, 17, 33] based on a multilevel approach and fast force approximation within a suitable space decomposition scheme. They can scale to graphs of millions of vertices and edges.

While the spring-electrical model has proven to be scalable and to yield 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, but such treatment is not rigorous.

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

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

where 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 root in multidimensional scaling (MDS) [23] which was eventually applied to graph drawing [21, 24]. 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 typically 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. Based on this observation, node positions can be found by an eigen-decomposition 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 [4], 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 takes $O(|V|^2 \log|V| + |V||E|)$ times, and $O(|V|^2)$ memory. For large graphs, these models are computationally too expensive.

A number of strategies [8, 13] have been proposed to approximately minimize the stress or strain model. One notable effort is that of PivotMDS of Brandes and Pich [3]. 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 schemes is needed" [4]. 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 [16], where in non-rigid graphs (e.g., trees), multiple nodes share the same position.

In this paper we are interested in embedding graphs with specified 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 non-rigid 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 non-neighboring 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 the physics principle which states that, subject to known constraints, a system always settles to a state of maximum entropy. This maximum entropy principle provides the least-biased estimate possible on the given information, i.e., it is "maximally noncommittal with regard to missing information" [20]. This principle is also believed to give rise to aesthetic beauty in nature. In the words of architect Greg Lynn [26], when-

[*]e-mail: erg@research.att.com
[†]e-mail: yifanhu@research.att.com
[‡]e-mail: north@research.att.com

ever 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 [29] and natural language processing [2]. 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.

2 Related Work

In machine learning, Chen and Buja [6] 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;

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

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 S if j is among the k nearest neighbors of i. They showed that this model, with careful tuning, outperforms MDS, PCA, LLE, Isomap, and KPCA [25] on image clustering. Their work differs from ours in its motivation and in the energy model used. In addition, they applied their method only to relatively small examples of up to 1956 nodes. Other than the selection of the parameter t, they did not give details on the implementation of the algorithm, so it is difficult to assess its scalability.

Another related model is the binary stress model of Koren and Civril [22]. In that model, there is a distance of 0 between nodes sharing an edge, and a distance of 1 otherwise. The model is then

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

Energy models other than stress or spring-electrical models have also been suggested. Noack [27, 28] proposed the LinLog model and, more generally, the r-PolyLog model,

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

in the context of graph clustering.

The solution of the maxent-stress model requires an initial layout, for which we use PivotMDS [3]. This gives a very fast approximate solution to classical scaling, and provides a good starting point for the stress model. High-dimensional embedding [16] is another fast algorithm, based on distances to k-centers and principal component analysis, that could be used as an initial solution.

Chalmers [5] proposed the first linear-time iterative algorithm for dimensionality reduction in the context of visualization via stochastic sampling, and Ingram et al. [19] proposed Glimmer, a multiscale variant adapted to run efficiently on graphics cards. In these works, a single entry of the distance matrix is assumed to be available in constant time, which is valid for multi-dimensional data, but not for graphs, where graph theoretical distances have to be calculated.

Another attempt at a scalable, distance-sensitive embedding is GRIP [12]. This is a multilevel algorithm, with coarsening carried out by maximal independent set based filtration. On coarse levels, a Kamada-Kawai algorithm [21] 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 [11]. Because of this last step, the algorithm does not strictly solve a stress model.

In Section 1 we argued that symmetry should be used to fill in the missing node-node distance information. Symmetry has long been studied in the context of graph drawing. Purchase [30] found that perceptual symmetry reduces reaction time or errors, though not both. Symmetry has a more positive effect than other aesthetics when it is at the maximum value. Eades and Lin [10] proved that the solution of a "general spring model" can uncover symmetries. This model is very general and includes many energy models as a special case, including the full stress model and Eades' force model [9].

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 formula (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 by $H(x)$. The model we are proposing is:

$$\begin{aligned}&\max H(x)\\&\text{subject to } \|x_i-x_j\|=d_{ij},\ \{i,j\}\in S.\end{aligned}\tag{2}$$

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 (2) 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

$$\min\sum_{\{i,j\}\in S} w_{ij}\left(\|x_i-x_j\|-d_{ij}\right)^2-\alpha H(x).\tag{3}$$

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 exist notions of pointset entropy in image processing and graph entropy based on statistical attributes of nodes, but these do not seem applicable. Rather, 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 in space, 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

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

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

$$H(x)=-\sum_{\{i,j\}\notin S}\left\|x_i-x_j\right\|^{-q},\ q\geq 0,\tag{4}$$

where we denote $\|x\|^0=-ln\|x\|$.

3.1 Force-augmented stress majorization

Next we turn to showing how the maxent-stress model (3) can be solved while avoiding the cost of the full stress model.

The minimum for (3) 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

$$\sum_{\{i,j\}\in S} 2w_{ij}\left(\|x_i-x_j\|-d_{ij}\right)\frac{x_i-x_j}{\|x_i-x_j\|}-\alpha\sum_{\{i,j\}\notin S} q\frac{x_i-x_j}{\|x_i-x_j\|^{q+2}}=0, \tag{5}$$

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

$$\sum_{\{i,j\}\in S} w_{ij}\left(x_i-x_j\right)=\sum_{\{i,j\}\in S}\frac{w_{ij}d_{ij}(x_i-x_j)}{\|x_i-x_j\|}+\alpha\sum_{\{i,j\}\notin S}\frac{x_i-x_j}{\|x_i-x_j\|^{q+2}} \tag{6}$$

In matrix form, this is

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

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

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

and the vector $b(x)$ has elements

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

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 $b(x)_i$, 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. For this reason we call this method force-augmented stress majorization.

An alternative way to solve the maxent-stress model is

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

where $\rho_i=\sum_{\{i,j\}\in S} w_{ij}$. This simple iterative scheme 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 initial layout. Several further implementation details need to be resolved. First, the repulsive force term (8) involves an almost all-pairs computation, which has the $|V|^2$ 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 Barnes-Hut approximation [1, 32, 31] to compute 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).

Note that 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 typical weighting factor of $w_{ij}=1/d_{ij}^2$. For the repulsive force calculation, based on our experimentation, and following the implementation of sfdp (a multilevel force directed algorithm[17] based on the spring-electrical model, and available as part of `Graphviz` [14]), our implementation uses $q=0$ except for a graph with many degree-1 nodes (i.e., more than 30% of the nodes are degree-1 nodes), where we set $q=0.8$. The justification is that for such graphs, a weaker repulsive force helps to avoid a "warping effect" [18]. In the graphs tested, $q=0.8$ only for `btree` and `1138_bus`.

We also need to set the value of α. Since the goal of solving (3) is to satisfy the constraints in (2), 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 non-rigid graphs. Therefore we start with a relatively large α, and gradually reduce it. To make sure 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 $\alpha := 0.3*\alpha$ in 5 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 the force-augmented stress majorization algorithm in C, compiled with `gcc -O3`. 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 denoted as *Maxent*) on a range of graphs. For comparison, we also tested PivotMDS, and PivotMDS plus 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, and for technical reasons, we used a version that assumes unit distances. We summarize all the tested algorithms in Table 1.

The exception of graph `gd`, which is an author collaboration graph of the International Symposium on Graph Drawing between 1994-2007, the graphs used are from the University of Florida Sparse Matrix Collection [7]. Our selection covers a range of graph sizes, and includes mesh-like and other non-mesh graphs, and graphs from Brandes and Pich's experimental study of distance scaling [4]. Two of the graphs (`commanche` and `luxembourg`)

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

Table 2: Test graphs. Graphs marked * have pre-specified non-unit edge lengths. Otherwise, unit edge length is assumed.

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

have associated pre-defined 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 that 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 [4], each drawing has an associated error chart. In an error chart, the x-axis gives the graph distance bins, the y-axis is the difference between the actual geometric distance in the layout and the graph distance. The chart shows the median (black line), the 25 and 75 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 vs. actual distance as suggested by Brandes and Pich [4]. Because generating the error chart requires an all-pairs shortest paths calculation, we provide this chart only for graphs $< 10,000$ nodes.

With the error chart, we also 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. In making the error charts, the layout is scaled to minimize the full stress, with $w_{ij} = 1/d_{ij}^2$.

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 non-rigid graphs, including 1138_bus, btree, lp_ship041 and USpowerGrid. Compared with PivotMDS and PivotMDS(1), the median line for Maxent (column 4) does not undershoot 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 long graph distance, seen as a dip of the median line past the x-axis for large x. This is most likely due to the warping effect [18] 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 peripheral. The error charts for GRIP are seen to resemble those of sfdp 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 bcsstk31, 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 underlining 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 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.

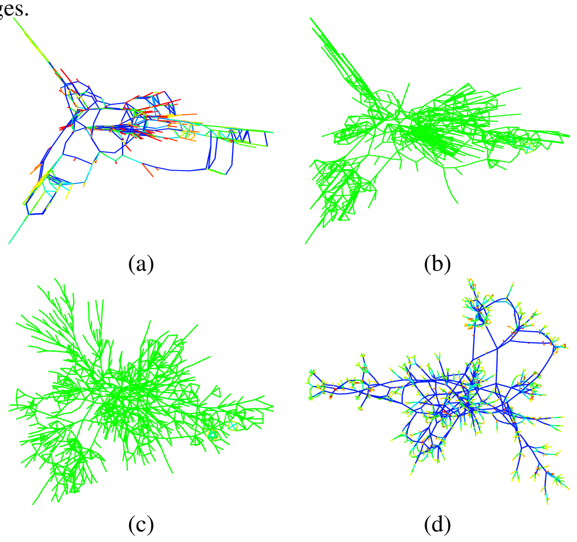


Figure 1: Drawings by (a) PivotMDS, (b) PivotMDS(1), (c) Maxent and (d) sfdp, on the 1138_bus graph

Figure 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 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.

Figure 2 shows layouts for the lp_ship041 graph. The drawing by FSM, seen in Figure 2 (a), identified 4 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 3 arms. PivotMDS(1) does not do much better. PivotMDS(2) is able to expand 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 distance of nodes many hopes 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. Figure 3 (a) 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 regener-

Table 3: Drawings and error charts of algorithms. 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 hour CPU time 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.

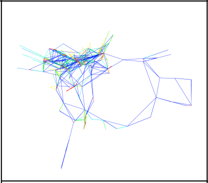
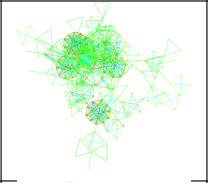
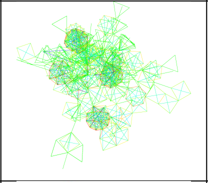
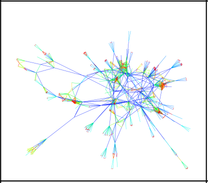
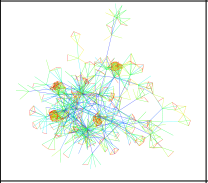
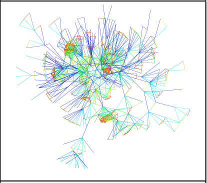
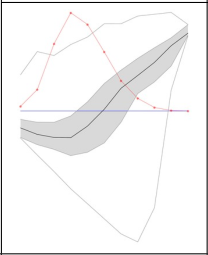
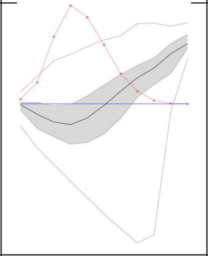
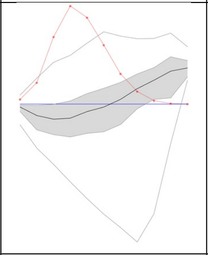
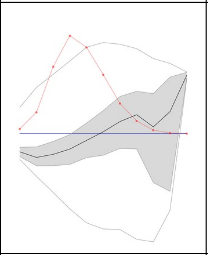
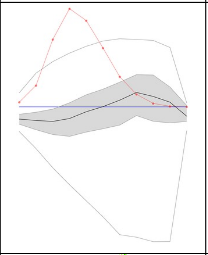
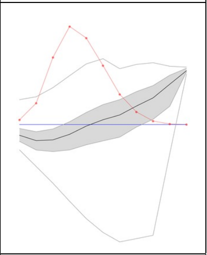
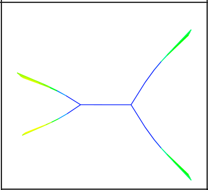
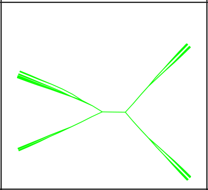
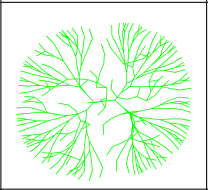
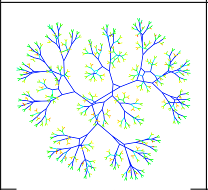
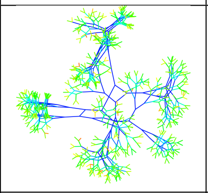
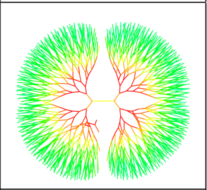
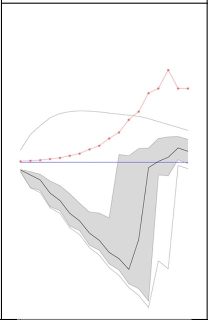
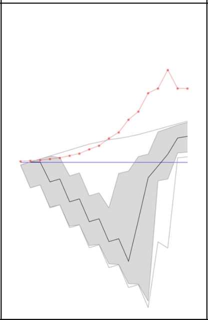
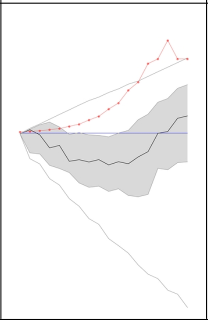
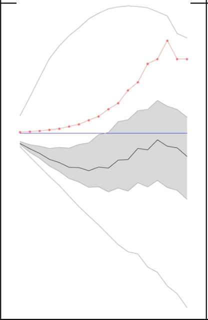
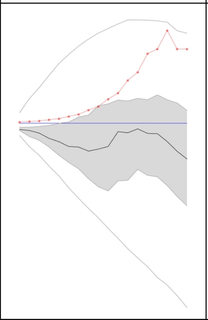
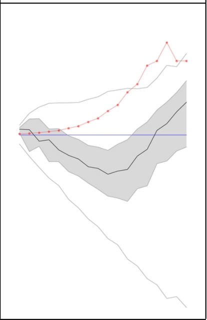
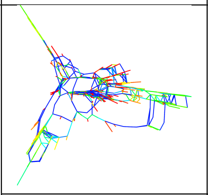
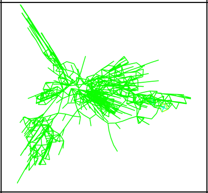
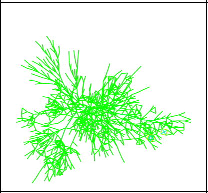
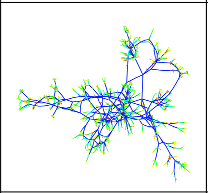
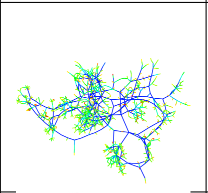
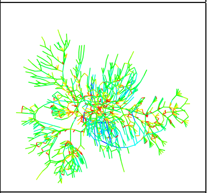
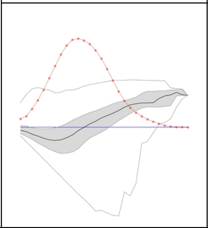
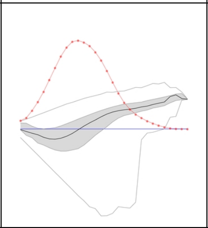
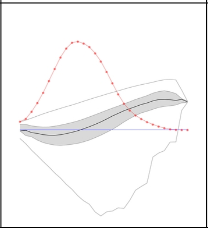
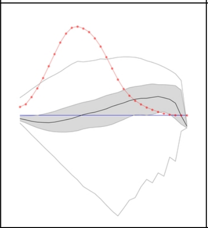
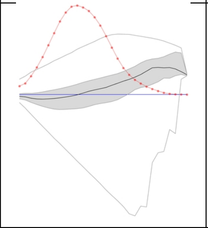
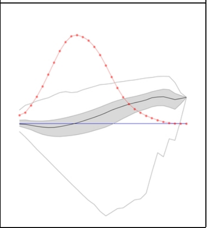
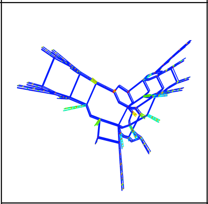
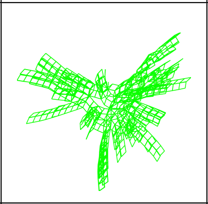
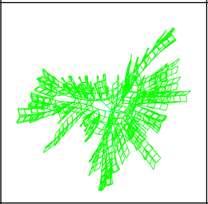
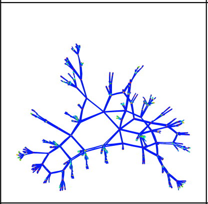
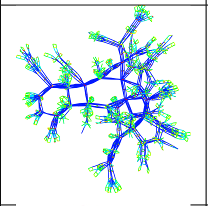
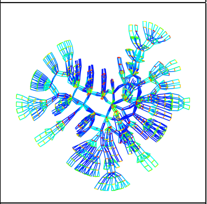
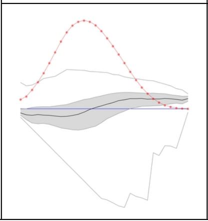
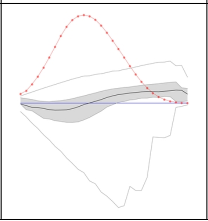
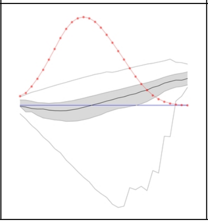
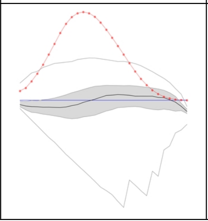
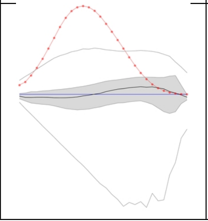
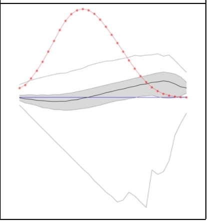
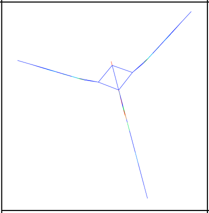
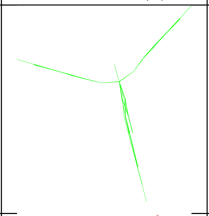
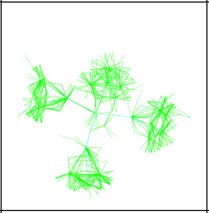
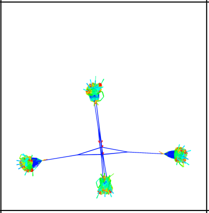
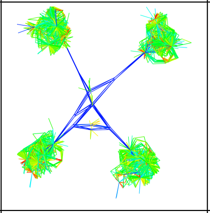
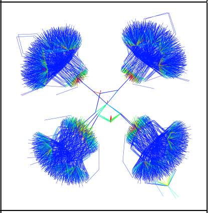
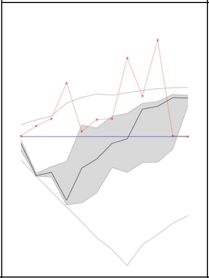
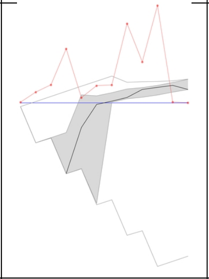
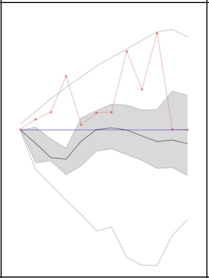
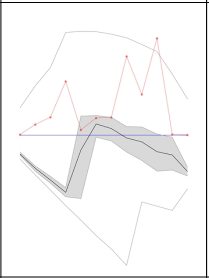
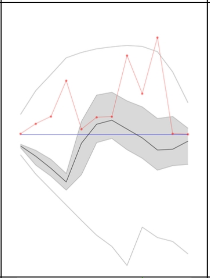
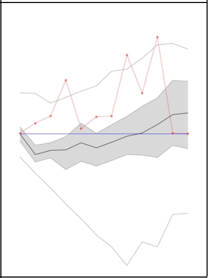
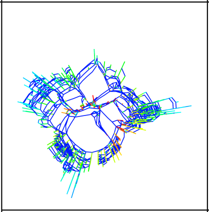
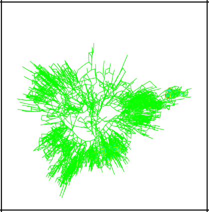
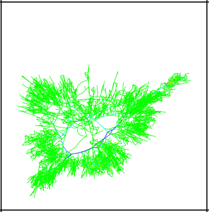
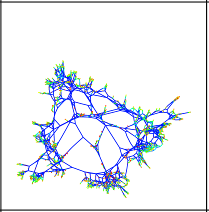
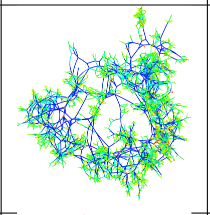
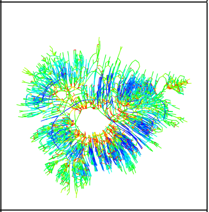
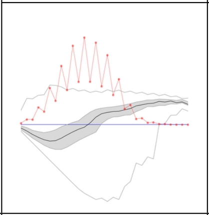
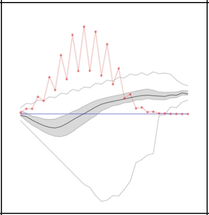
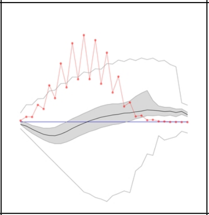
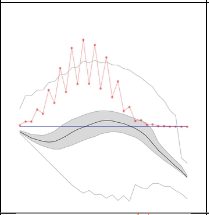
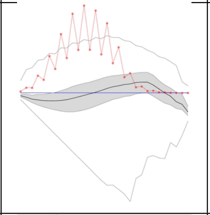
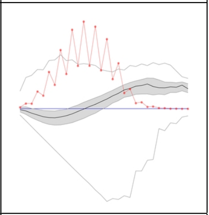
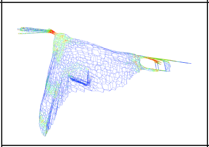
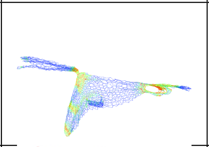
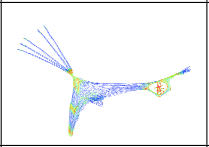
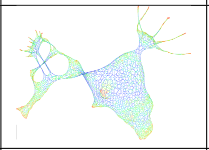
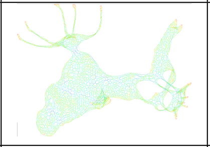
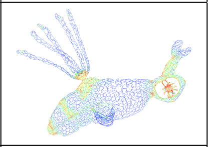
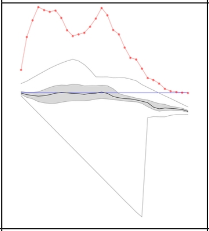
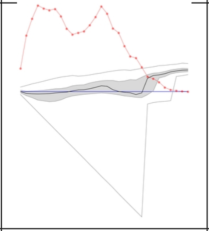
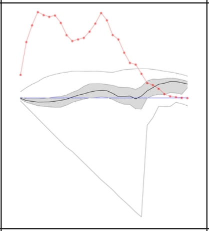
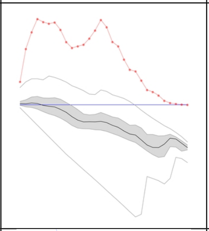
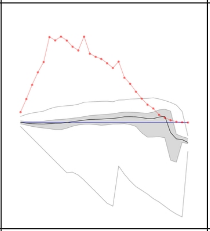
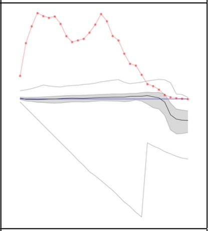
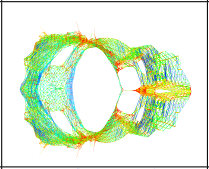
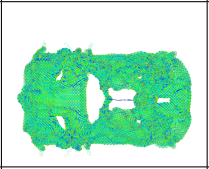
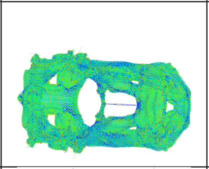
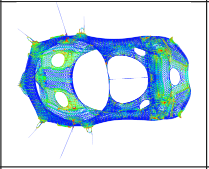
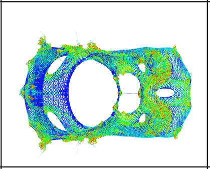
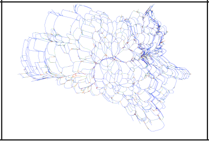
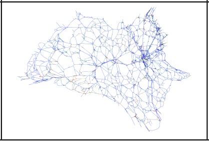
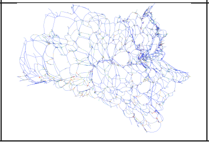
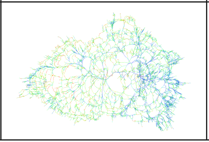
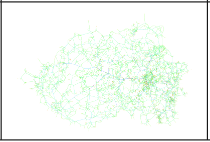
Graph	PivotMDS	PivotMDS(1)	Maxent	sfdp	GRIP	FSM
gd						
						
btree						
						
1138_bus						
						
qh882						
						

Table 3: (continued) Drawings and error charts of algorithms. 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 hour CPU time 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.

Graph	PivotMDS	PivotMDS(1)	Maxent	sfdp	GRIP	FSM
lp_ship04l						
						
USpowerGrid						
						
commanche						
						
bcsstk31						-
Luxembourg						-

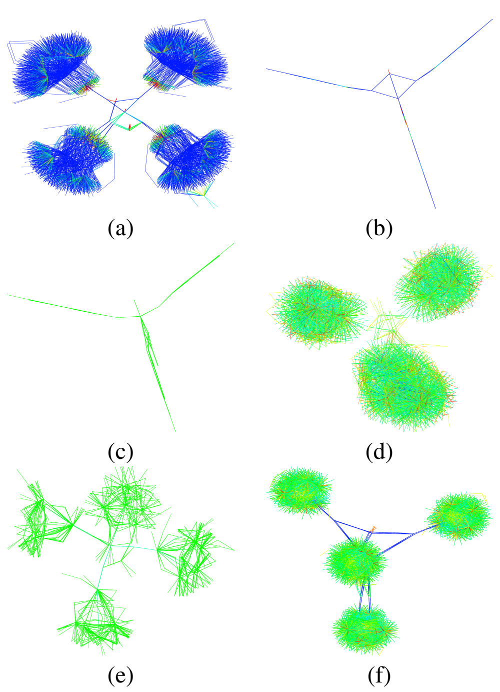


Figure 2: Drawings by (a) FSM, (b) PivotMDS (c) PivotMDS(1), (d) PivotMDS(2), (e) Maxent and (f) Maxent(2) on the `lp_ship04l` graph.

ated. 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.

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. As a compromise, we propose to measure full stress, as defined by (1), with $w_{ij} = 1/d_{ij}^2$. Bear in mind that this measure naturally favors the full stress model. Table 4 gives the full stress measure achieved by each algorithm. Because it is expensive to calculate all-pairs shortest paths, we restrict experimental measurement to graphs with less 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 and GRIP yield high full stress on `commanche`, because both assume unit edge length on a non-unit edge length input graph.

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% higher than Maxent, but the drawing given by Maxent(2) (Figure 3 (f)) does not seem any worse than that of Maxent (Figure 3 (e)). We conjecture that because we use the maximal entropy principle to cope 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.

Table 5 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 room for improvement in the implemen-

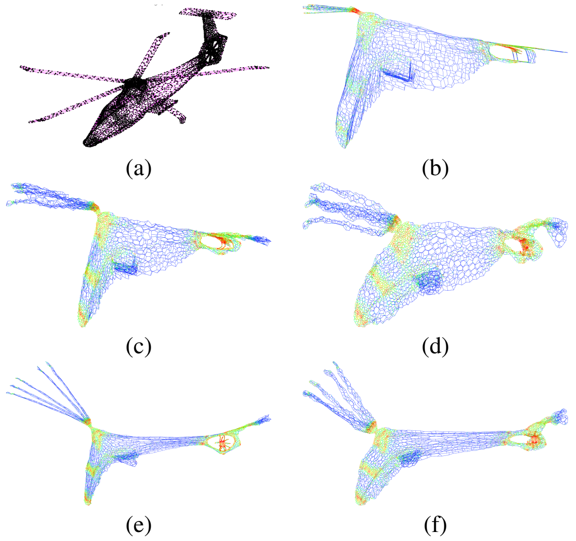


Figure 3: Original graph (a), and drawings by (b) PivotMDS, (c) PivotMDS(1), (d) PivotMDS(2), (e) Maxent and (f) Maxent(2) on the `commanche` graph.

tation of the force-augmented maxent algorithm. Both sfdp and GRIP scale well to large graphs. For comparison, we also include the CPU time for the FSM. Clearly CPU time for FSM increases very quickly with the size of the graph, and it does not scale to large graphs.

5 Discussion and Conclusions

This study 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 non-rigid graphs.

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 (2) by making distance satisfaction the top priority, even though (2) 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). This might be combined with a multilevel scheme, which would do away with the need for an initial global layout. We have implemented a prototype which seems promising.

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

We would like to thank Ulrik Brandes for his lecture at PacificVis 2011 on PivotMDS, and subsequent discussions.

References

[1] J. Barnes and P. Hut. A hierarchical O(NlogN) force-calculation algorithm. *Nature*, 324:446–449, 1986.

Table 4: Full stress measure for PivotMDS, PivotMDS(1), PivotMDS(2), Maxent, Maxent(2), sfdp and FSM. Smaller is better.

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

Table 5: CPU time (in seconds) for PivotMDS, PivotMDS(1), PivotMDS(2), Maxent, Maxent(2), sfdp and FSM. A limit of 10 hour CPU time is imposed and "-" is used to denote runs that could not finish within that time, or ran out of memory.

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	-

[2] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), 1996.

[3] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Proc. 14th Intl. Symp. Graph Drawing (GD '06)*, volume 4372 of *LNCS*, pages 42–53, 2007.

[4] U. Brandes and C. Pich. An experimental study on distance based graph drawing. In *Proc. 16th Intl. Symp. Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 218–229. Springer-Verlag, 2009.

[5] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of the 7th conference on Visualization '96*, VIS '96, pages 127–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[6] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the Americal Statistical Association*, 104:209–219, 2009.

[7] T. A. Davis and Y. F. Hu. University of Florida Sparse Matrix Collection. *ACM Transaction on Mathematical Software*, 38, 2011. `http://www.cise.ufl.edu/research/sparse/matrices/`.

[8] V. de Solva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 721–728. MIT Press, 2003.

[9] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[10] P. Eades and X. Lin. Spring algorithms and symmetry. In *COCOON*, pages 202–211, 1997.

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

[12] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A fast multidimensional algorithm for drawing large graphs. *LNCS*, 1984:211 – 221, 2000.

[13] E. R. Gansner, Y. Koren, and S. C. North. Graph drawing by stress majorization. In *Proc. 12th Intl. Symp. Graph Drawing (GD '04)*, volume 3383 of *LNCS*, pages 239–250. Springer, 2004.

[14] E. R. Gansner and S. North. An open graph visualization system and its applications to software engineering. *Software - Practice & Experience*, 30:1203–1233, 2000.

[15] S. Hachul and M. Jünger. Drawing large graphs with a potential field based multilevel algorithm. In *Proc. 12th Intl. Symp. Graph Drawing (GD '04)*, volume 3383 of *LNCS*, pages 285–295. Springer, 2004.

[16] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. *J. Graph Algorithms and Applications*, 8(2):195–214, 2004.

[17] Y. F. Hu. Efficient and high quality force-directed graph drawing. *Mathematica Journal*, 10:37–71, 2005.

[18] Y. F. Hu and Y. Koren. Extending the spring-electrical model to overcome warping effects. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 129–136. IEEE Computer Society, 2009.

[19] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel mds on the gpu. *IEEE Trans. Vis. Comput. Graph.*, 15(2):249–261, 2009.

[20] E. T. Jaynes. Information theory and statistical mechanics. *Physical Reviews Series II*, 106(4):620–630, 1957.

[21] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[22] Y. Koren and A. Çivril. The binary stress model for graph drawing. In *Proc. 16th Intl. Symp. Graph Drawing (GD '08)*, volume 5417 of *LNCS*, pages 193–205. Springer-Verlag, 2008.

[23] J. B. Kruskal. Multidimensioal scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[24] J. B. Kruskal and J. B. Seery. Designing network diagrams. In *Proceedings of the First General Conference on Social Graphics*, pages 22–50, Washington, D.C., July 1980. U. S. Department of the Census. Bell Laboratories Technical Report No. 49.

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

[26] G. Lynn. Calculus in architecture. `http://www.ted.com/index.php/talks/greg_lynn_on_organic_design.html`, 2005.

[27] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11, 2007.

[28] A. Noack. Modularity clustering is force-directed layout. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 79, 2009.

[29] S. J. Phillips, R. P. Anderson, and R. E. Schapire. Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, 190(3-4):231–259, 2006.

[30] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proc. 5th Intl. Symp. Graph Drawing (GD '01)*, volume 1353 of *LNCS*, pages 248–261. Springer-Verlag, 1997.

[31] A. Quigley. *Large scale relational information visualization, clustering, and abstraction*. PhD thesis, Department of Computer Science and Software Engineering, University of Newcastle, Australia, 2001.

[32] D. Tunkelang. *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, Carnegie Mellow University, 1999.

[33] C. Walshaw. A multilevel algorithm for force-directed graph drawing. *J. Graph Algorithms and Applications*, 7:253–285, 2003.