Extending the Spring-Electrical Model to Overcome Warping Effects

Yifan Hu*
AT&T Labs – Research, 180 Park Ave, Florham Park, NJ 07932

Yehuda Koren†
Yahoo! Research
Haifa 31905, Israel.

ABSTRACT

The spring-electrical model based force directed algorithm is widely used for drawing undirected graphs, and sophisticated implementations can be very efficient for visualizing large graphs. However, our practical experience shows that in many cases, layout quality suffers as a result of non-uniform vertex density. This gives rise to warping effects in that vertices on the outskirt of the drawing are often closer to each other than those near the center, and branches in a tree-like graph tend to cling together. In this paper we propose algorithms that overcome these effects. The algorithms combine the efficiency and good global structure of the spring-electrical model, with the flexibility of the Kamada-Kawai stress model of in specifying the ideal edge length, and are very effective in overcoming the warping effects.

Keywords: Graph drawing, force directed methods, warping effect.

Index Terms: G.2.2 [Discrete Mathematics]: Graph Theory—Graph Algorithms

1 INTRODUCTION

The spring-electrical model [8, 10] is widely used for drawing undirected graphs. It is relatively easy to implement, and when combined with the multilevel approach and suitable data structures (e.g., quad-tree) to approximate long range repulsive forces, is very efficient and generally effective for large graphs [16, 19]. However, it has a limitation that is often overlooked, but seriously hampers its usefulness for real-world applications. It suffers from warping effects in that vertices far away from the center of a layout tends to be closer to each other, and branches in a tree-like graph tend to cling together (see, e.g., Figure 1(a) and Figure 6(b)). These effects are not usually noticeable on small graphs. But when applied to large graphs, particularly those with many nodes of small degrees, such as "small-world" graphs [30], the warping effects can be particularly pronounced, and can degrade the clarity of their drawing, particularly the local details.

Another popular graph drawing method, the stress model [22], is based on realizing given distances between vertices. In this approach, a cost function that is the difference between the physical distance of vertices and their ideal distance is minimized, with the ideal distance determined from the graph theoretical distance among vertices. This model achieves more uniform edge lengths, thus avoiding the aforementioned warping effects. However the calculation of graph theoretical distances among all vertex pairs makes the computational complexity quadratic in the number of vertices. The robustness and efficiency of this approach can be enhanced by the stress majorization technique [14], or by combining it with a multilevel approach [11, 17, 19]. Nevertheless the quadratic complexity means that this method is not suitable for graphs with more than a few thousand vertices.

In this paper we propose two algorithms that overcome the warping effects of the spring-electrical model. Both algorithms act as a post processing step, taking the layout from the spring-electrical model as the input. At the heart of these algorithms is a cost function similar to the stress model, except that calculation of all-pairs shortest path is avoided, and the matrices involved are all sparse, thus the algorithms are computationally efficient.

2 GRAPH DRAWING ALGORITHMS AND WARPING EFFECTS

We use $G = \{V, E\}$ to denote an undirected graph, with V the set of vertices and E edges. Denote by $|V|$ and $|E|$ the number of vertices and edges, respectively. If vertices i and j form an edge, we denote that as $i \leftrightarrow j$, and call i and j neighboring vertices. We denote by x_i the current coordinates of vertex i in two or three dimensional Euclidean space.

The aim of graph drawing is to find x_i for all $i \in V$ so that the resulting drawing gives a good visual representation of the connectivity information between vertices. Two popular methods, the spring-electrical model [8, 10], and the stress model [22], both convert the problem of finding an optimal layout to that of finding a minimal energy configuration of a physical system.

The stress model assumes that there are springs connecting vertices of the graph, with the ideal spring length equal the graph theoretical distance among vertices. The energy of this spring system is

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

where d_{ij} is the graph theoretical distance between vertices i and j, and w_{ij} is a weight factor, typically $1/{d_{ij}}^2$. The layout that minimizes the above stress energy is an optimal layout of the graph according to this model. There are several ways to find a solution of the minimization problem. A force based iterative approach can be used. Alternatively, a stress-majorization technique can be employed, where the cost function (1) is bounded by a series of quadratic functions from above, and the process of finding an optimum becomes that of solving a series of linear systems [14]; see Section 5 for further details. Whichever method is used to solve the model, the graph theoretical distance between all pairs of vertices has to be calculated, making the computational complexity quadratic in the number of vertices. There were attempts to sparsify the stress function by considering only a small portion of the pair-wise distances (e.g., [4, 14]), however our experience with real life networks was that these sparsification techniques often fail to yield good layouts for "small-world" like graphs.

The spring-electrical model [8] represents the graph drawing problem by a system of electrically charged vertices attracted to each other by springs; vertices also repel each other via electrical forces. Here, following [10], the attractive spring force exerted on vertex i from its neighbor j is proportional to distance between these two vertices,

$$F_a(i,j) = \frac{\left\| x_i - x_j \right\|^2}{K}, \; i \leftrightarrow j,$$

*e-mail: yifanhu@research.att.com

†Work done while this author was at AT&T Labs. e-mail:yehuda@yahoo-inc.com

where K is a parameter related to the nominal edge length of the final layout. The repulsive electrical force exerted on vertex i from any vertex j is inversely proportional to the distance between these two vertices,

$$F_r(i,j) = -\frac{K^2}{\|x_i - x_j\|},\ i \neq j. \tag{2}$$

The spring-electrical model can be solved by starting from a random layout, calculating the combined attractive and repulsive forces on each vertex, and moving the vertices along the direction of the force for a certain step length. This process is repeated, with the step length decreasing every iteration, until the layout stabilizes. This procedure can be enhanced by an adaptive step length updating scheme [5, 19], and usually works well for small graphs.

For large graphs, this simple iterative procedure is not sufficient to overcome the many local minima that often exist in the space of all possible layouts. Instead, multilevel approach has to be used [29]. Furthermore, nested space partitioning data structure is needed to approximate the all-to-all electrical force [2, 25, 28] so as to reduce the quadratic complexity to $O(|V|\log|V|+|E|)$. Combining these two powerful techniques resulted in efficient implementations of the spring-electrical model [16, 19] that are capable of handling graphs of millions of vertices and edges [18].

Even though the spring-electrical model has the advantage that it can be made to work efficiently on very large graphs, it does suffer from *warping effects* (also known as peripheral effects [19]), in that vertices on the out-skirt of a drawing tend to be closer to each other, and branches in a tree-like graph often cling to each other, see, e.g., Figure 1(a) and Figure 6(b). This is particularly pronounced for graphs with a large diameter, and can degrade the clarity of their drawing, particularly the local details. This effect was illustrated in [19] with a simple line graph of 100 vertices: when laid out using a spring-electrical algorithm, the edges at the center are longer than edges at the two ends, with the ratio between the longest and the shortest edges equals 2.72:1.

Little recorded efforts was found in the literature that attempt to overcome the warping effect of the spring-electrical model. One strategy [19] is to use an alternative repulsive force model,

$$F_r(i,j) = -\frac{K^{1+p}}{\|x_i - x_j\|^p},\ i \neq j. \tag{3}$$

When $p = 1$, this is the same repulsive force as (2); but when $p > 1$, this gives a weaker repulsive force for vertices that are far apart, yet a stronger repulsive force for vertices that are close to each other. Using this simple strategy, for the same line graph, it was found [19] that the ratio between the longest and the shorted edges reduces to 1.33:1 for $p = 2$ and 1.06:1 for $p = 3$.

In practice, with a suitable choice of p, this strategy is remarkably effective for many graphs. The difficulty lies however in deciding a suitable value of p: if p is chosen too large, the long range repulsive force becomes too weak and parts of the graph can fold into each other. Figure 1(a) shows the result of applying a multilevel spring-electrical algorithm with the standard force model (2) to the graph `qh882` ($|V| = 882, |E| = 1533$) [7]. It is clear from the figure that while the global structure of the graph is captured nicely, locally, some vertices are too close to each other. For example, vertices at the tip of a branch are much closer to each other than those in the middle. The branches at the tips cling to each other, due to the strong long range repulsive force from far away vertices. If we use the alternative force model (3) with $p = 1.8$, we get a better drawing (Figure 1 (b)). Now we can see the ladder like local structure much clearer.

However, it is difficult to know in advance what the final layout would look like for a given value of p, and too large a value can cause long range repulsive forces to be so weak that parts of the graph fold into each other, as seen in Figure 1 (c) when $p = 4$. Furthermore, there is no control of the length of each individual edge, what comes out of the iterative process is what we get.

The stress model does not suffer from the warping effects, because the cost function encodes the ideal spring length. However as discussed, this model is not suitable for very large graphs. Therefore our motivation is to devise algorithms that overcome the warping effects of the spring-electrical model, without destroying the efficiency and good global structure that can be achieved with the model. We accomplish this by taking advantage of the fine control of edge length offered by the stress model, while avoiding its quadratic complexity. We propose two models in the following, both assume that an initial layout is available. The layout reveals global structures well, even though it suffers from the warping effect.

3 A Localized Stress Model

The layout from a spring-electrical algorithm tends to reveal the global structure of the graph very well. Therefore our first intuition is that perhaps we can take the layout, and fine tune it locally as a post processing step. One way of achieving this is to impose an ideal edge length by minimizing a cost function $\sum_{(i,j)\in P}\left(\|x_i - x_j\| - d_{\mathrm{ij}}\right)^2$ for pairs of vertices (i,j) in a set P that will be defined shortly. Here d_{ij} is the ideal distance between vertices i and j, to be specified. We assume that spring-electrical model has produced a globally good layout, so we want to refrain each vertex i from deviating too much from its current position, ${x_i}^0$. Therefore to this cost function we can add a penalty function, which results in:

$$\sum_{(i,j)\in P} w_{\mathrm{ij}}\left(\|x_i - x_j\| - d_{\mathrm{ij}}\right)^2 + \sum_{i\in V}\lambda_i\left\|x_i - {x_i}^0\right\|^2. \tag{4}$$

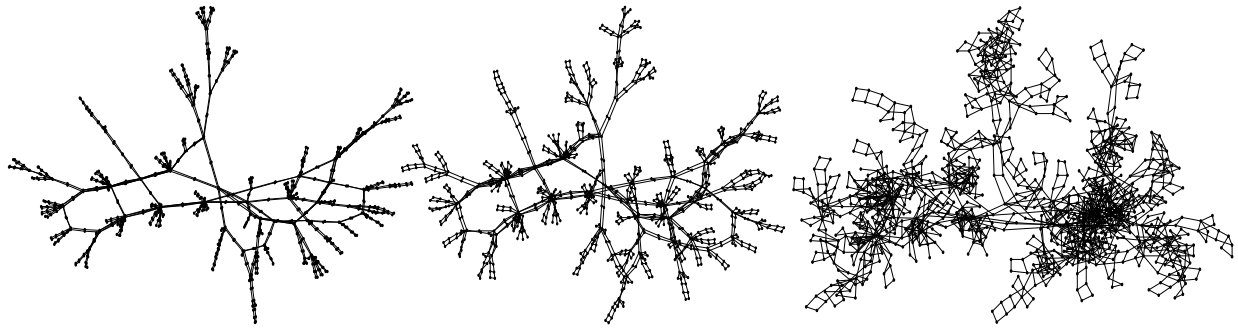
We denote this localized stress model (LSM). This objective of keeping vertices close to their original position is known as anchoring [3, 9, 23]. In the above model, λ_i is a penalty parameter specifying the penalty we impose on vertex i for moving away from the current position. If $\lambda_i = 0$ and P is the set of all pairs $\{(i,j)|i \neq j,\ i,j \in V\}$, then the above is exactly the stress model. However for efficiency sake we localize the model by using a much smaller set P. A small set P, such as $P = E$, would be very efficient, but it only imposes a distance requirement on neighboring vertices, therefore does not help in solving the problem of branches clinging to each other as seen in Figure 1(a) and Figure 6(b). On the other hand, a larger set P that includes most of the vertex pairs results in a high computational complexity. To balance quality with efficiency, we set P to be the vertex pairs with a graph theoretical distance of no more than 2. This does mean that for a graph with large 2-neighborhoods, for example a graph that contains a large star-like structure, the model still has a high complexity.

We also need to decide what is the ideal distance d_{ij} for a pair of vertices. There are a number of possibilities, for example, we could set d_{ij} to be proportional to the graph theoretical distance (1 or 2). Alternatively, we could set it to be related to the local average edge length. After some experiments, we find that setting d_{ij} to a power of the current distance,

$$d_{\mathrm{ij}} = \left\|{x_i}^0 - {x_j}^0\right\|^t,\ t < 1 \tag{5}$$

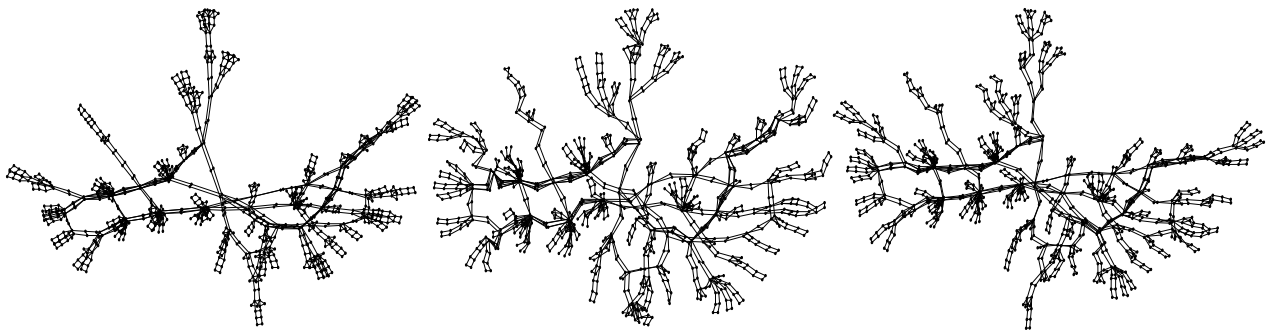
works well. this is because when $\|{x_i}^0 - {x_j}^0\|$ is large, the power function makes it smaller, while when $\|{x_i}^0 - {x_j}^0\|$ is small, the power function makes it larger, therefore overall the power function evens out the variation in the distance.

It is important to scale the above ideal distance to the proper unit, as explained later in equation (8).



(a) (b) (c)

Figure 1: Applying a spring-electrical algorithm with different repulsive force models on `qh882` with $|V| = 882$ and $E| = 1533$. (a) With the standard force model (2). (b) With the alternative force model (3) ($p = 1.8$). (c) With the alternative force model (3) ($p = 4$).



(a) (b) (c)

Figure 2: Applying post processing algorithms to the layout of Figure 1(a). (a) LSM. (b) The Delaunay triangulation based PGM. (c) Relative neighborhood graph based PGM.

4 A Proximity Graph Based Model

Our experience with the localized stress model LSM suggests that it tends to work well for mesh like graphs. On the other hand, for sparse networks with many degree-1 nodes, LSM tends to be too conservative in expanding the layout to fill up the available white space, due to the penalty term. If we reduce the penalty by using smaller penalty parameters λ_i, the final layout can deviate from the initial layout so significantly that many of the nice features of the initial layout are lost. The reason is that the penalty term $\lambda_i \left\| x_i - {x_i}^0 \right\|^2$ imposes a stringent constraint on the position of the vertex with reference to its current position. A more flexible, yet adequate constraint would be to maintain the relative vertex positions. We like to set up a "scaffolding" structure so that while vertices can move around, their relative positions are maintained. This scaffolding is constructed using a proximity graph. We note that the idea of maintain proximity is not new, and can dates at least back to Lyons et al. [23], where a Voronoi diagram is generated for the purpose of node overlap removal. But the idea has not been used for the purpose of improving the force directed algorithm. We note that in [12, 20], a proximity stress model was used for node and edge label overlap removal, where a sparse stress model based on Delaunay triangulation is applied. The difference to the work here is that the ideal edge length was calculated based on the amount of overlaps, and the model is solved repeatedly until overlaps are removed.

A proximity graph is a graph derived from a set of points in the space: points that are "neighbors" tend to form an edge in the proximity graph. There are several ways to create a proximity graph [21], we work with two of these:

- The Delaunay triangulation (DT): two points are neighbors in the DT if and only if there exist a sphere passing through these two points, and no other points lie in the interior of this sphere.
- The relative neighborhood graph (RNG): two points x_i and x_j are neighbors in RNG if and only if no point x_k is both closer to x_i than x_j and closer to x_j than x_i. RNG is a spanning subgraph of DT.

The approach we take is to first form a proximity graph, either a DT or a RNG. Then we merge the proximity graph with the original graph, to form a new graph $G' = \{V, E'\}$, whose edges include both original edges and edges from the proximity graph. We then minimize a cost function

$$\sum_{(i,j)\in E'} w_{\mathrm{ij}} \left(\left\| x_i - x_j \right\| - d_{\mathrm{ij}} \right)^2, \tag{6}$$

with d_{ij} the ideal distance between vertices i and j, defined in (5).

Notice that DT is a planar graph, therefore has no more than $3|V| - 3$ edges. Hence G' has the same number of vertices, and no more than $3|V| - 3$ extra edges, compared with the original graph.

Furthermore, G', as a spanning supergraph of DT, is rigid[1], therefore provides a good scaffolding that constrains the relative positions of the vertices and helps to preserve the good global structure of the spring-electrical layout. Hereafter we shall denote the proximity graph based model (6) PGM.

5 Numerical Evaluation

In this section we will evaluate LSM and PGM on some graphs. Our baseline algorithm is a multilevel spring-electrical algorithm [19], hereafter denoted as SFDP (Scalable Force Directed Placement). SFDP checks the input graph, and automatically switches to the force model (3) with $p = 1.8$ if at least 30% of the nodes are of degree 1. We use the mesh generator Triangle [26, 27] for triangulation. All results are generated on a 64-bit Linux machine with a Intel Xeon 3.20 GHz CPU and 8 GB of memory, using gcc compiler version 3.4.6.

5.1 Stress majorization

Both LSM and PGM, defined in Equations (4) and (6) respectively, have similar forms to the stress model defined by Equation (1), therefore we solve them by applying the stress majorization technique [14], which was demonstrated to be a robust algorithm for finding the minimum of (1). Consider the cost function of LSM:

$$\begin{aligned} f(x) &= \sum_{(i,j)\in P} w_{\mathrm{ij}} \left(\|x_i - x_j\| - d_{\mathrm{ij}}\right)^2 + \sum_{i\in V} \lambda_i \left\|x_i - x_i^0\right\|^2 \\ &= \sum_{(i,j)\in P} \left(w_{\mathrm{ij}}\|x_i - x_j\|^2 - 2d_{\mathrm{ij}} w_{\mathrm{ij}}\|x_i - x_j\| + w_{\mathrm{ij}} {d_{\mathrm{ij}}}^2\right) \\ &\quad + \sum_{i\in V} \lambda_i \|x_i - x_i^0\|^2 \end{aligned}$$

All the terms above are either constant, linear, or quadratic with regard to x, except the second one. Using the Cauchy-Schwartz inequality, $\left(x_i - x_j\right)^T \left(y_i - y_j\right) \le \left\|x_i - x_j\right\| \|y_i - y_j\|$, we can bound the cost function by

$$\begin{aligned} & g(x,y) \\ = & \sum_{(i,j)\in P} \left(w_{\mathrm{ij}}\|x_i - x_j\|^2 - 2d_{\mathrm{ij}} w_{\mathrm{ij}} \frac{\left(x_i - x_j\right)^T \left(y_i - y_j\right)}{\|y_i - y_j\|} + w_{\mathrm{ij}} {d_{\mathrm{ij}}}^2\right) \\ & + \sum_{i\in V} \lambda_i \|x_i - x_i^0\|^2, \end{aligned}$$

with the bound tight when $y = x$. The idea of stress majorization is to minimize a sequence of quadratic function $g\left(x, y^k\right)$, with $y^0 = x^0$ the initial layout, and subsequent y^k the result of minimizing $g\left(x, y^{k-1}\right), k = 1, 2, \ldots$.

The minimum of the quadratic function $g(x,y)$ is derived by setting $\partial_{x_i} g(x,y) = 0$, giving

$$\left(L_w + \Lambda\right) x = L_{w,d}\, y + \Lambda x^0 \tag{7}$$

where the weighted Laplacian matrix L_w has elements

$$\left(L_w\right)_{\mathrm{ij}} = \begin{cases} \sum_{(i,l)\in P} w_{\mathrm{il}}, & i = j \\ -w_{\mathrm{ij}}, & i \ne j \end{cases}$$

and matrix $L_{w,d}$ has elements

$$\left(L_{w,d}\right)_{\mathrm{ij}} = \begin{cases} \sum_{(i,l)\in P} w_{\mathrm{il}}\, d_{\mathrm{il}} / \|y_i - y_l\|, & i = j \\ -w_{\mathrm{ij}}\, d_{\mathrm{ij}} / \left\|y_i - y_j\right\|, & i \ne j \end{cases}$$

In (7), Λ is a diagonal matrix, with the i-th diagonal entry λ_i. Therefore the problem of finding a minima of $g(x,y)$ becomes that of solving the linear system (7), with the left hand side matrix fixed and sparse (provided that P is a small subset of all possible vertex pairs). In fact when all penalty parameters λ_i are positive, the matrix $L_w + \Lambda$ is diagonally dominant, so an iterative procedure such as the preconditioned conjugate gradient method [15] should converge quickly on the linear system. We use a diagonal preconditioner, and terminate the conjugate gradient algorithm if the relative 2-norm residual for (7) is less than 0.01. A tighter tolerance is not necessary because the solution of (7) constitutes an intermediate step of the stress majorization. The resulting solution x^k is used to substitute for the y in (7) and the linear system solved again until $\left\|x^{k+1} - x^k\right\| / |V| < \varepsilon$. We used $\varepsilon = 0.001$.

The proximity graph model (6) can be similarly solved using the stress majorization procedure, except here $P = E'$, and terms related to the penalty parameters vanish.

For LSM, it is particularly important that we scale the ideal distance suitably. The scaling factor s is chosen to minimize the initial stress $f\left(x^0\right)$,

$$\left(\sum_{(i,j)\in P} w_{\mathrm{ij}} \left(\left|{x_i}^0 - {x_j}^0\right| - s d_{\mathrm{ij}}\right)^2 + \sum \lambda_i \|{x_i}^0 - {x_i}^0\|^2\right)' = 0$$

or

$$s = \frac{\sum_{(i,j)\in P} w_{\mathrm{ij}} \left\|{x_i}^0 - {x_j}^0\right\| d_{\mathrm{ij}}}{\sum_{(i,j)\in P} w_{\mathrm{ij}} {d_{\mathrm{ij}}}^2}. \tag{8}$$

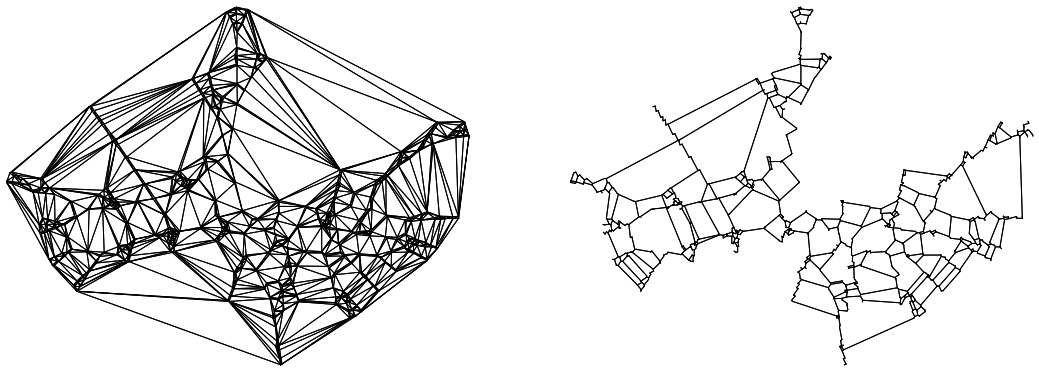
5.2 Numerical results

We have experimented with different values of t in the ideal distance formula $d_{\mathrm{ij}} = \left\|{x_i}^0 - {x_j}^0\right\|^t$, and found that $t = 0.4$ works well for LSM. We set the penalty parameters $\lambda_i = 0.05$. Figure 2(a) shows the result of LSM on graph `qh882`. As can be seen, LSM improves upon the original drawing (Figure 1(a)) by revealing more details of the graph, such as the ladder structure. At the same time, it does not deviate from the original drawing very much.

Figure 2(b) shows the results of applying PGM, with the Delaunay triangulation as the proximity graph. Here we used distance formula $d_{\mathrm{ij}} = \left\|{x_i}^0 - {x_j}^0\right\|^{0.6}$ (0.6 is our default setting for PGM). Compared with Figure 2(a), this drawing utilizes more available space. For example, the branch at the top is now expanded and can be seen more clearly. The downside of PGM is that some branches which were straight in the original drawing now become twisted, such as the branch at the bottom of Figure 1(a). The reason is that the Delaunay triangulation can create edges that link far away vertices together. Figure 3(a) shows a Delaunay triangulation of the vertices in Figure 1(a). In this Delaunay triangulation there is a particularly long edge linking the bottom most vertex with one to the left. In addition there are four long edges that link the bottom branch to a vertex to the right. Since we set the ideal distance of PGM to be the 0.6 power of the current distance, the bottom branch is "pulled" to the left and right, which explains the kink of this branch seen in Figure 2(b).

We also experimented with relative neighborhood graph based PGM. Here, for computational convenience and efficiency, following [13], we generate an approximation to the relative neighborhood graph by starting from a Delaunay triangulation, and removing edges between vertices i and j if there is a vertex k adjacent to i or j such that $\left\|x_i - x_j\right\| > \min\left\{\|x_i - x_k\|, \left\|x_j - x_k\right\|\right\}$. Because we only check neighboring vertices, the result is a superset of the true

[1] Here a graph is defined as rigid, if a layout of the graph is not flexible in 2D. A layout is flexible if there exists a non-trivial continuous deformation (rotation and translation are considered trivial deformations) from this layout to another, such that edge lengths are preserved.



(a) (b)

Figure 3: Proximity graphs using the layout of Figure 1(a). (a) The Delaunay triangulation. (b) Relative neighborhood graph.

relative neighborhood graph. This relative neighborhood graph for the layout in Figure 1(a) is seen in Figure 3(b).

While the Delaunay triangulation is rigid, from Figure 3(b) we can see that relative neighborhood graph is not rigid. If we imagine that edges are rods and vertices are joints, the top branch of Figure 3(b) is clearly flexible and can swing to the left or right. This non-rigidity means that realizable layout that minimizes the cost function (4) is not unique, and is subject to deformation. We found that for some non-rigid graphs, parts of the layout after applying relative neighborhood graph based PGM can fold into each other. For graph `qh882`, however, this proved not to be a real issue and the resulting layout in Figure 2(c) is reasonable.

Because of the many long edges that exist in the Delaunay triangulation, for mesh like graphs, DT based PGM is not suitable as it tends to destroy the symmetry that exist in the drawing of the spring-electrical model (Figure 4). Relative neighborhood based PGM tends to suffer less from this problem, and together with LSM are suitable for such graphs, as seen in Figure 5 where these two algorithms are applied to the `dwt_1005` graph [7].

For sparser graphs, however, the long edges in the Delaunay triangulation proved very effective in pulling out branches that cling to each other, thus utilize the empty space. Figure 6(a) shows the result of our baseline spring-electrical algorithm SFDP on the `USA.ncol` graph. This graph is a spanning tree taken from a web crawl graph, generated by Bill Cheswick of AT&T Labs, similar to those seen in [6]. The drawing is pleasing and highlights the tree nature of this graph. However, there are a lot of white spaces that are not utilized. This is a common problem for the spring-electrical model, and is also seen in other implementations. Figure 6(b) shows the result of applying the FM^3 algorithm [16] with the default settings. Here the warping effects, with the branches cling tightly together, is also obvious.

If we now apply LSM and PGM to the layout of Figure 6(a), we get Figure 7. Clearly both LSM and PGM improve space utilization. For such sparse graphs, it is beneficial to augment the original edge set with the Delaunay edges, thus making the graph rigid and enabling the stress function to preserve relative position well. Accordingly Figure 7(b), which is based on PGM, is probably superior to Figure 7(a).

Both LSM and PGM are relatively cheap computationally. Table 1 shows the CPU time for laying out a graph, and the post processing time. For graphs that do not contain any large 2-neighborhood (`dwt_1005` and `qh882`), LSM is faster because the penalty term makes the linear system involved in stress majorization more strongly diagonally dominant, hence conjugate gradient algorithm converges in less iterations, and computational cost per iteration is also small. However for the `USA.ncol` graph, LSM is slower. This is because the graph contains a large star structure (seen as the dense round clump near the center of Figure 6(a)). This makes the 2-neighborhood graph within this structure a complete graph, hence even though conjugate gradient algorithm still converges in a small number of iterations (stress majorization took 5 iterations, the average number of iterations for conjugate gradient is 7), the relatively dense matrix makes the computational cost higher compared with PGM, whose matrix is almost as sparse as the adjacency matrix of the original graph. For PGM, stress majorization takes 19 iterations, and on average conjugate gradient takes 132 iterations to converge. As a benchmark, in the table we also included the CPU time for the FM^3 algorithm [16].

Table 1: Comparing the CPU time (in seconds) for the baseline algorithm SFDP, post processing algorithms LSM and the Delaunay triangulation based PGM, as well as the FM^3 algorithm.

Graph	|V|	|E|	SFDP	LSM	PGM	FM^3
`USA.ncol`	44954	44953	174	57	31	130
`dwt_1005`	1005	7616	1.	0.05	0.6	1.68
`qh882`	882	3066	0.93	0.03	0.44	1.79

There are algorithms that are specifically designed to work with tree like graphs and try to layout a graph by filling in existing white spaces, notably LGL [1], a popular program for visualizing biological networks. It is based on the Internet map visualization algorithm of Cheswick et al. [6]. LGL introduces nodes in the breadth-first spanning tree order into the layout, and uses iterations of a force directed algorithm to find good position for the nodes. Figure 8 shows the result of applying LGL to the `USA.ncol` graph. While LGL fills up the space very well, it obfuscates the fact that the graph is a tree, and there are a lot of edge crossings. It took 3288 seconds of CPU time, an order of magnitude longer than SFDP combined with LSM or PGM. It is not clear to us why LGL is so much slower.

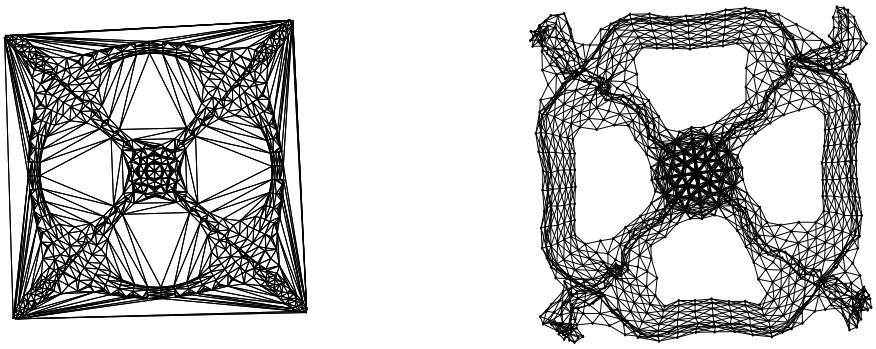


Figure 4: Applying the Delaunay triangulation based PGM algorithms to graph dwt_1005 with $|V| = 1005$ and $|E| = 3808$. Left: the Delaunay triangulation which contains many long edges. Right: the Delaunay triangulation based PGM fail to preserve the symmetry.

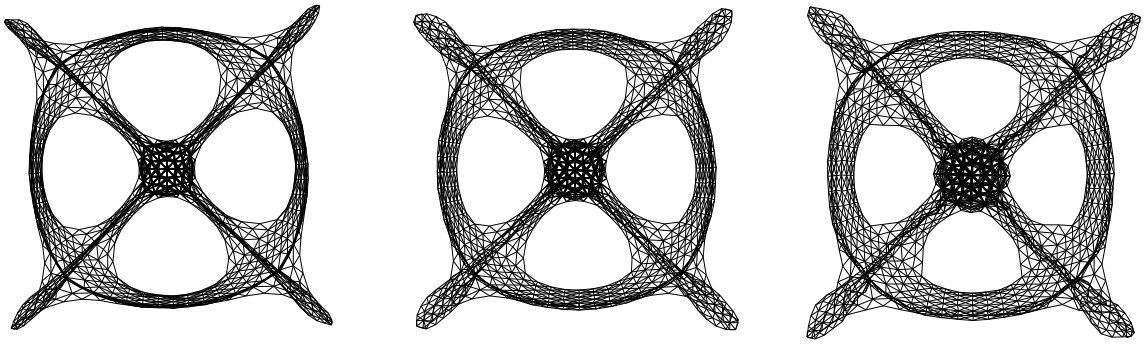


Figure 5: Applying post processing algorithms to graph dwt_1005 with $|V| = 1005$ and $|E| = 3808$. Left: layout by SFDP. Middle: after applying LSM. Right: after applying relative neighborhood based PGM.

6 Conclusions and Future Work

The main contribution of this paper is in proposing two related algorithms to overcome the warping effects of the spring-electrical model. The algorithms can be implemented efficiently, and are shown to produce aesthetic drawing for very large graphs not possible using any other existing algorithms known to us.

We have also explored other strategies. For example, one approach investigated is to start from a layout, and execute a few steps of refinement by adding to the spring-electrical model a force that attracts (repels) if the distance between two vertices are larger (smaller) than the ideal distance. However none seems to work as well as the two algorithms presented here.

For future work, we would like to investigate techniques that reduce the complexity of LSM for graphs that possess large 2-neighborhoods. We are also interested in devising an energy model which includes both the spring-electrical model and the LSM/PGM models, this would unify the current two-step process and enable the model to be incorporated into a multilevel procedure. It may also be fruitful to explore alternative force models. The LinLog model [24] aims to force clustering of nodes in graphs of small diameters through weakened attractive force, thus achieves the opposite of what we set out to do. However, warping effect may be alleviated by the opposite of the LinLog model: using a stronger attractive force. It would be interesting to compare and contract this with the alternative repulsive force model, as described by Equation (3).

We wish to note that laying out scale-free, real-life networks, particularly those with "small-world" property [30], remains a challenge. The algorithms presented here are one step forward in meeting that challenge.

Acknowledgements

We would like to thank Stefan Hachul and Michael Jünger for providing the FM^3 code, and Alex Adai and Edward Marcotte for the LGL code. We would like to thank Stephen North for helpful discussions, and the referees for detailed comments.

References

[1] A. T. Ada, S. V. Date, S. Wieland, and E. M. Marcotte. LGL: creating a map of protein function with an algorithm for visualizing very

(a) (b)

Figure 6: Spring-electrical algorithms on the `USA.ncol` graph with $|V| = 44954$ and $|E| = 44953$. (a) Layout by SFDP. (b) Layout by FM^3.

large biological networks. *Journal of Molecular Biology*, 340:440–442, 1998.

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

[3] U. Brandes, V. Kaab, A. Loh, D. Wagner, and T. Willhalm. Dynamic www structures in 3d. *Journal of Graph Algorithms and Applications*, 4:2000, 2000.

[4] U. Brandes and C. Pich. An experimental study on distance based graph drawing. In *GD'08: Proceedings of the Symposium on Graph Drawing*, 2008 (to appear).

[5] I. Bruss and A. Frick. Fast interactive 3-D graph visualization. *LNCS*, 1027:99–11, 1995.

[6] B. Cheswick, B. Burch, and S. Branigan. Mapping and visualizing the internet. In *Proceeding of USENIX Annual Technical Conference*, 2000.

[7] T. Davis. The university of florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices.

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

[9] Y. Frishman and A. Tal. Online dynamic graph drawing. In *proceeding of Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*, pages 75–82, 2007.

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

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

[12] E. R. Gansner and Y. F. Hu. Efficient node overlap removal using a proximity stress model. In *GD'08: Proceedings of the Symposium on Graph Drawing*, 2008 (to appear).

[13] E. R. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11:457–468, 2005.

[14] 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.

[15] G. H. Golub and C. F. V. Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, 1996.

[16] 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.

[17] D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. *J. Graph Algorithms and Applications*, 6:179–202, 2002.

[18] Y. F. Hu. A gallery of large graphs. http://research.att.com/∼yifanhu/GALLERY/GRAPHS.

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

[20] Y. F. Hu. Visualizing graphs with node and edge labels. manuscript, 2008.

[21] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80:1502–1517, 1992.

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

[23] K. A. Lyons, H. Meijer, and D. Rappaport. Algorithms for cluster busting in anchored graph drawing. *J. Graph Algorithms and Applications*, 2(1), 1998.

[24] A. Noack. An energy model for visual graph clustering. In *Proceedings of the 11th International Symposium on Graph Drawing (GD 2003)*, volume 2912 of *LNCS*, pages 425–436. Springer, 2004.

[25] A. Quigley and P. Eades. Fade: Graph drawing, clustering, and visual abstraction. *LNCS*, 1984:183–196, 2000.

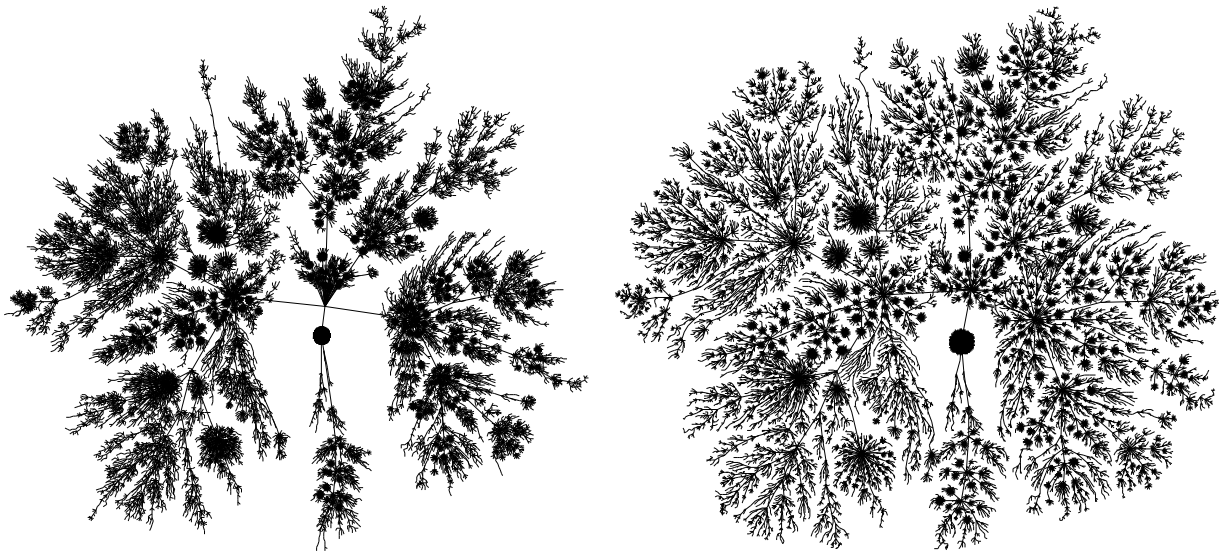
[26] J. R. Shewchuk. Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *LNCS*, pages 203–222. Springer, 1996.

[27] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22:21–74, 2002.

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

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

[30] D. Watts and S. Strogate. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.



(a) (b)

Figure 7: Applying post processing algorithms to the layout in Figure 6(a). (a) LSM. (b) the Delaunay triangulation based PGM.

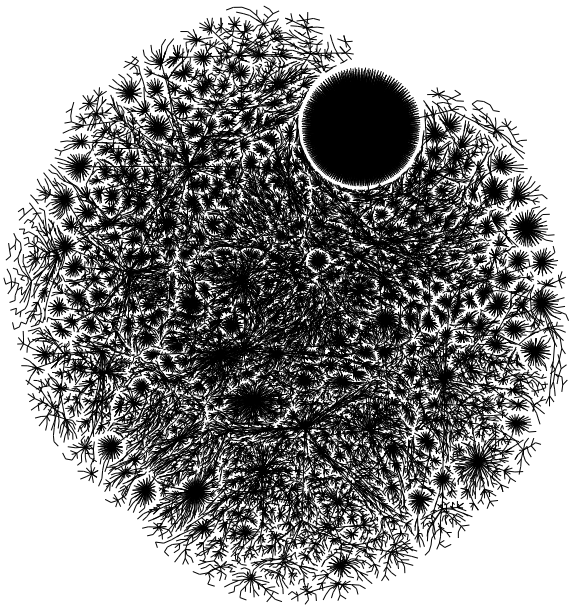


Figure 8: Result of LGL on `USA.ncol` graph.