OpenOrd: An Open-Source Toolbox for Large Graph Layout

Shawn Martin[a], W. Michael Brown[a], Richard Klavans[b], and Kevin W. Boyack[b]

[a]Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185
[b]SciTech Strategies, Inc., 2405 White Horse Rd, Berwyn, PA, 19132

ABSTRACT

We document an open-source toolbox for drawing large-scale undirected graphs. This toolbox is based on a previously implemented closed-source algorithm known as VxOrd. Our toolbox, which we call OpenOrd, extends the capabilities of VxOrd to large graph layout by incorporating edge-cutting, a multi-level approach, average-link clustering, and a parallel implementation. At each level, vertices are grouped using force-directed layout and average-link clustering. The clustered vertices are then re-drawn and the process is repeated. When a suitable drawing of the coarsened graph is obtained, the algorithm is reversed to obtain a drawing of the original graph. This approach results in layouts of large graphs which incorporate both local and global structure. A detailed description of the algorithm is provided in this paper. Examples using datasets with over 600K nodes are given. Code is available at www.cs.sandia.gov/∼smartin.

Keywords: Multilevel, Force-Directed, Parallel, Large-Scale Graph Layout

1. INTRODUCTION

Graph drawing is used to visualize relational data, typically in two dimensions.[1,2] Some applications of graph drawing include social network analysis,[3] scientific literature analysis,[4,5] cartography,[6] and bioinformatics.[7,8] There are a variety of algorithms available for graph drawing, each of which optimizes a different set of aesthetic criteria. Some examples of aesthetic criteria include minimizing the number of edge crossings, minimizing total edge length, and maximizing separation between vertices. For undirected graphs drawn with straight line edges, one of the most commonly used algorithms is force-directed layout.[9–13]

In this paper, we document a graph drawing algorithm specialized for drawing large-scale real-world graphs. Our algorithm uses edge-cutting, average-link clustering, multilevel graph coarsening, and a parallel implementation of a force-directed method based on simulated annealing. Related algorithms for force-directed layout exist, including algorithms taking a multilevel approach;[13–16] algorithms which include node clustering;[16–18] and algorithms implemented using a parallel GPU architecture.[19,20] However, our algorithm is the only one available which incorporates all three of these ideas: a multilevel approach, node clustering, and a parallel implementation (note that our parallelism is cluster based, instead of GPU based). In addition, we introduce a heuristic for edge-cutting, designed to allow visualization of graphs which may not have a desirable degree distribution (often found in real-world graphs).

Our algorithm is based on a previous force-directed algorithm called VxOrd.[21,22] This new version, OpenOrd, is described in the following pages. In Section 2, we give the motivation for our modifications of VxOrd. In Section 3, we describe the various parts of our algorithm, including force-directed layout; layout in parallel (3.2); recursive graph coarsening (3.3); and average-link clustering (3.4). In Section 4, we demonstrate some of the properties of our algorithm using applications to several real-world datasets, including a 659K vertex Wikipedia article dataset. Finally, in Section 5, we provide our conclusions. Code for OpenOrd is available at http://www.cs.sandia.gov/∼smartin.

Further author information: (Send correspondence to S.M.)
S.M.: E-mail: smartin@sandia.gov, Telephone: 1 505 284 3601
W.M.B.: E-mail: wmbrown@sandia.gov, Telephone: 1 505 284 8938
R.K.: E-mail: rklavans@mapofscience.com, Telephone: 1 610 251 2135
K.W.B.: E-mail: kboyack@mapofscience.com, Telephone: 1 505 856-1267

Visualization and Data Analysis 2011, edited by Pak Chung Wong, Jinah Park, Ming C. Hao, Chaomei Chen, Katy Börner, David L. Kao, Jonathan C. Roberts, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7868, 786806 · © 2011 SPIE-IS&T · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.871402

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

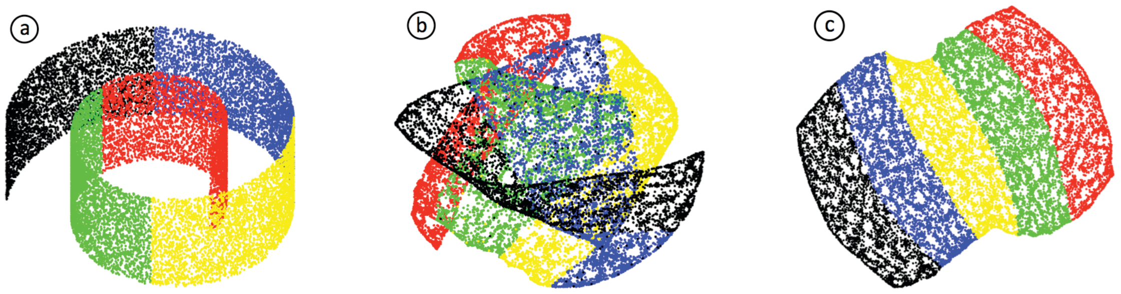

Figure 1. Tangled global structure. Shown here on the left (a) is the Swiss roll dataset, consisting of 20,000 points sampled at random from the Swiss roll manifold.[29,30] In the middle (b) is a force-directed layout of the graph obtained from the Swiss roll using 20 nearest neighbors using VxOrd. Only nodes are plotted and the graph is colored to illustrate the tangling of the global structure. On the right (c) the correctly drawn graph is shown, produced using the multilevel version of OpenOrd.

2. MOTIVATION

OpenOrd is a force-directed layout algorithm specialized to handle very large graphs, such as those encountered in a scientific literature citation analysis.[4,5] It is based on an implementation of the Frutcherman-Reingold algorithm[11] known as VxOrd,[22] which has been used previously in scientific citation analysis[23–25] and in the analysis of bioinformatics data.[26–28] OpenOrd represents our effort to scale VxOrd to very large graphs (greater than 100K vertices).

We have identified three problems associated with scaling force-directed layout to large graphs, which we describe in this section as our motivation for the subsequent algorithm in Section 3. First, the ability of the algorithm to correctly uncover the global structure of the layout decreases with the size of the graph. This effect undoubtedly varies depending on the degree distribution of the nodes, but it apparent even for relatively simple mesh-like graphs. To illustrate this phenomenon, we used the Swiss roll data set originally introduced in the field of nonlinear dimension reduction.[29,30] (Note that we use this dataset only as an example – our algorithm is for graph drawing, not dimension reduction.) We imposed a graph on 20,000 points sampled from the Swiss roll manifold using 20 nearest neighbors and drew the graph using VxOrd. The results are shown in Figure 1(a,b). The inability of force-directed layout to correctly uncover global structure was also noted in previous work,[13,31] where it was shown that a multilevel approach could be used to alleviate the effect. Thus we include a multilevel strategy in OpenOrd.

Second, we have found that the use of force-directed layout with large real-world graphs often results in visually unappealing layouts. Often these graphs are sparse but still well-connected (i.e. not scale-free) so that the resulting drawing looks fully connected. An example using a 6,147 node graph with 61,646 edge imposed on Yeast microarray data (fully described in Section 4) is shown in Figure 2(a). Although the graph has only 61,646 edges (0.3% of the number possible), it looks fully connected. To encourage more visually appealing layouts, we use an edge-cutting strategy to encourage clustering of the nodes. This can be seen as a trade-off between the two competing forces in the force-directed layout and is described in Section 3.1.

Finally, the Frutcherman-Reingold approach to force-directed layout has a high running time (on the order of $O(n^2)$ in the number of nodes n). This is of course a major constraint in the application of force-directed layout to large graphs. The running time can be improved using a grid based density calculation, and by employing a multilevel approach. We implement both of these options, and also include an option to use parallel computation in OpenOrd.

3. ALGORITHM

As mentioned in Section 2, OpenOrd is based on the Frutcherman-Reingold algorithm for force-directed layout,[11] previously implemented as VxOrd.[22] To provide background, we describe that algorithm here, with modifications described in the following subsections. Suppose we have an undirected weighted graph $G = (V,E)$, where the vertices are given by $V = \{v_1,\ldots,v_n\}$ and the edges are given by $E = \{e_{ij}\}$. Let $W = (w_{ij})$ be the adjacency matrix corresponding to the graph G so that edge e_{ij} has weight w_{ij}. Since the graph is undirected, we know that $w_{ij} = w_{ji}$ so that W is symmetric.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

The goal of OpenOrd is to draw G in two dimensions. Let $\mathbf{x}_i = (x_{i,1}, x_{i,2})$ denote the position of v_i in the plane. OpenOrd draws G by attempting to solve

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_i \left(\sum_j \left(w_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2 \right) + D_{\mathbf{x}_i} \right), \tag{1}$$

where $D_{\mathbf{x}_i}$ denotes the density of the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ near $\mathbf{x}_i$. The sum in (1) contains both an attractive and a repulsive term. The attractive term $\sum_j (w_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2))$ attempts to draw together vertices which have strong relations via w_{ij}. The repulsive term $D_{\mathbf{x}_i}$ attempts to push vertices into areas of the plane that are sparsely populated.

The minimization in (1) is a difficult nonlinear problem. For that reason, we use a greedy optimization procedure based on simulated annealing. Our procedure is greedy in that we update the position of each vertex by optimizing the inner sum $\sum_j (w_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2) + D_{\mathbf{x}_i}$ while fixing the positions of the other vertices. All vertices are initially placed at the origin, and the update is repeated for each vertex in the graph to complete one iteration of the optimization. The iterations are controlled via a simulated annealing type schedule which consists of five different phases: liquid, expansion, cool-down, crunch, and simmer.

During each stage of the annealing schedule, we vary several parameters of the optimization: temperature, attraction, and damping. These parameters control how far vertices are allowed to move. At each step of the algorithm, we compute two possible vertex moves. The first possible move is always a random jump, whose distance is determined by the temperature. The second possible move is analytically calculated (known as a barrier jump[22]). This move is computed as the weighted centroid of the neighbors of the vertex. The damping multiplier determines how far towards this centroid the vertex is allowed to move and the attraction factor weights the resulting energy to determine the desirability of such a move. Of these two possible moves, we choose the move which results in the lowest inner sum energy $\sum_j (w_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2) + D_{\mathbf{x}_i}$.

The annealing schedule is determined by how much time is spent during each stage and the behavior of the optimization is determined by adjusting the various parameters. The default schedule spends approximately 25% of its time in the liquid stage, 25% in the expansion stage, 25% in the cool-down stage, 10% in the crunch stage, and 15% in the simmer stage. The liquid, expansion and cool-down stages all use the same temperature but vary the attraction factor and damping multiplier. The crunch and simmer stages use a lower temperature (approximately $1/4$ the temperature used during liquid, expansion, and cool-down), as well as lower attraction and damping. The annealing schedule was determined through extensive experimentation on various datasets (including the yeast dataset described in Section 4). The experiments were carried out via an interactive visualization environment so that each the effects of the parameters could be observed at each stage of the annealing process. Qualitatively, the expansion stage allows the largest movements in terms of attraction and damping, the liquid stage allows the next largest moevements, and the cool-down, crunch, and simmer stages all restrict movement.

Finally, we use a grid based method for computing the density term $D_{\mathbf{x}_i}$. Ordinarily, the density calculation would be $O(n^2)$, but by using a grid for the density calculation, we can reduce the cost to $O(n)$, with memory use increasing according to however many grid boxes we use in our calculation. This coarsening, however, may cause inaccuracies due to the fact that density varies discontinuously along grid lines. For this reason, the density grid is not used during the final simmer stage.

3.1 Edge-Cutting

In order to produce visually appealing layouts, we have developed a heuristic which allows user control of the amount of node clustering and white space present in the layout. To control node clustering, our heuristic affects the relative importance of the two competing terms in the objective function in Equation (1). To control white space we allow the ability to ignore certain long edges during the optimization of the objective function. In fact, both node clustering and white space can be simultaneously controlled by ignoring, or cutting, the long edges.

As mentioned previously, the term $\sum_j (w_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2)$ in the objective function from Equation (1) serves to attract nodes with large weight connections, while the term $D_{\mathbf{x}_i}$ is repulsive and discourages high average node density, or clusters. It is the relative importance of the two terms which determines the degree of clustering in a layout. If the attractive term dominates, less clustering is expected; if the repulsive term dominates more clustering will occur. Cutting long edges during the optimization decreases the value of the attractive term and therefore allows an increase in the value of the repulsive term, thus providing control over node clustering in the layout.

White space can be controlled by the number of long edges that we use in our calculation. Edges that are long but have large weight can exert undue influence on distant clusters, causing two groups to be placed on top of each other when a

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

more appealing layout might be obtained if they were allowed to separate. Cutting long edges during optimization allows clusters to separate. It also improves the final drawing since these edges extend across the entire layout and therefore obscure the finer structure present in the drawing. (This effect can be enhanced by using edge transparency in the final rendering, but must be present in the calculation to avoid placing clusters on top of each other.)

Edge-cutting in OpenOrd is specified using a fraction from 0 to 1. An edge-cutting value of 0 corresponds to the standard Frutcherman-Reingold layout algorithm (no cutting), while an edge-cutting value of 1 corresponds to aggressive cutting. Edge-cutting is allowed during the expansion and cool-down stages of the annealing schedule. At the beginning of these two stages, a threshold is specified as a percentage of the greatest distance between two nodes in the drawing. This percentage is controlled by the edge-cutting parameter, with a value of 0 being 100% of the greatest distance and a 1 being 0% of the greatest distance. Edge-cutting is subject to the constraint that each node must have at least one edge. During the expansion and cool-down stages, an edge will be cut if its distance is greater than the threshold.

The edge-cutting parameter does not correspond to the fraction of edges that will be cut. Depending on the input graph, the edge-cutting parameter may have very little effect, since a graph may be unlikley to have long edges during optimization (often the case for meshes). However, for real-world graphs the edge-cutting parameter can have a major influence on the layout of the final graph, and will be explored in Section 4.

3.2 Parallel Force-Directed Layout

OpenOrd can be run on both serial and parallel computers. The parallel version is similar to the serial version. Both use the same greedy update and follow the same annealing schedule. In the parallel version, however, the updates are performed in parallel instead of sequentially.

The parallel force-directed layout algorithm in OpenOrd starts by assigning each processor a random non-overlapping subset of the nodes of the graph. The processor keeps track of its assigned vertices as well any neighboring vertices. All processors keep track of the positions of each vertex so that each processor can maintain an identical copy of the density grid.

Each processor is responsible for moving its assigned vertices so as to optimize the objective function in Equation (1). Since each processor knows the positions of the assigned vertices, as well as the positions of neighboring vertices, the vertex positions can be updated using the same greedy procedure that was used in the serial version of the algorithm. After each vertex update, position information is exchanged among the processors and the process continues to completion.

In addition to increased computational speed, the parallel version of OpenOrd has the advantage that it can distribute a very large graph across many processors, thus using a computer with a huge amount of effective memory. This is feasible because any given graph will have many more edges than vertices. Further, by maintaining the same greedy update procedure and simulated annealing schedule, the results of both the serial and parallel version of OpenOrd are similar. The performance of the two algorithms as well as differences in output are discussed in Section 4.2.

3.3 Multilevel Graph Layout

Using OpenOrd in parallel allows the layout of very large graphs. Edge-cutting enhances the visual appeal of the layouts. However, the layouts still suffer from the potential for incorrect global structure as described in Section 2 using the Swiss roll. To address this concern, we adapted the multilevel approach taken by Walshaw[13] to OpenOrd. Walshaw's procedure proceeds as follows. First, a sequence of graphs $G_0 = G, G_1, \ldots, G_L$ is produced using a random coarsening procedure.[32] In the coarsening procedure, neighboring vertices are merged at random, and their edge weights are added if two neighbors have another neighbor in common. This process is repeated until a sufficiently small graph G_L is obtained. The graph G_L is drawn using a force-directed algorithm. The vertex placement in the drawing of G_L is used as a starting point for drawing the graph G_{L-1}. If, for example, vertices u and v in G_{L-1} were merged into w in G_L then u and v are placed in the position formerly occupied by w in the drawing of G_L. The force-directed algorithm is again applied to obtain a drawing of G_{L-1} and the process is repeated.

Walshaw's method is very fast and is demonstrated to work well on large graphs.[13] We use Walshaw's approach with an additional refinement. Instead of a random coarsening procedure, we use a clustering based coarsening procedure (described next). Of course, we also use OpenOrd as the force-directed layout. Otherwise, we use the procedure outlined in the previous paragraph and described in more detail by Walshaw.[13]

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

Using our force-directed layout algorithm requires additional adjustments to accommodate the simulated annealing schedule and allow the use of edge-cutting in the multilevel approach. After obtaining a sequence of coarsened graphs $G_0, \ldots, G_L$, we follow Walshaw by producing a layout of G_L with default edge-cutting, using the standard annealing schedule. During refinement, we place the vertices as done in Walshaw,[13] again using default edge-cutting, but modifying our annealing schedule to avoid the liquid stage and minimizing the expansion stage. We also eliminate the simmer stage during refinement. The final layout is produced using more aggressive edge-cutting and includes the simmer stage. Surprisingly, our experiments indicate that our multilevel layout method works well for datasets ranging in size from 6K to 850K vertices, even using the same annealing schedule/edge-cutting parameters. Experiments performed and datasets used are described in Section 4.

3.4 Average-Link Clustering

In our multilevel layout algorithm, we use average-link clustering to provide the coarsened graphs $G_0 = G, G_1, \ldots, G_L$. Our clustering algorithm is based on a average-link agglomerative model, where we use both edge weights and distance to provide clusters. Distances are taken from a drawing by our force-directed layout algorithm. Once clusters are determined, we merge all the vertices in a given cluster to obtain a vertex in the new coarser graph. Edges are merged according to the method described previously.

In describing our method, let us assume that we are coarsening G_0 to obtain G_1. Obtaining the remaining coarsened graphs is done using the same procedure. The first step in our clustering algorithm is to draw G_0 using maximum edge-cutting with our force-directed layout algorithm. This step provides us with a proxy representation that can be used to provide distances between vertices, in addition to the edge weights defining G_0. Maximum edge-cutting encourages the layout algorithm to produce a naturally clustered representation (see Section 4.3 for an example).

Next we produce a new undirected weighted graph $\widetilde{G}_0$ whose edges are computed according to the distances in our drawing of G_0. Edges in this graph include any edges not cut by the layout algorithm along with the largest weight edges for each node in G_0. We include the largest weight edges to ensure a connected graph. The edge weights in $\widetilde{G}_0$ are no longer the original edge weights provided in G_0; they are now the distances between connected vertices in $\widetilde{G}_0$.

Our derived graph $\widetilde{G}_0$ can be used in an agglomerative clustering algorithm. The algorithm that we use is a form of average-link hierarchical clustering.[33,34] In this algorithm, each vertex is initially assigned to a unique cluster. Clusters are then merged with neighboring clusters, as measured by distance between cluster centroids. Traditionally, this process is repeated until everything has been merged into a single cluster. In our algorithm, however, we provide a distance threshold after which we discontinue forming new clusters. The distance threshold in the average-link clustering can be provided by the user.

Alternatively, the threshold can be automatically selected by OpenOrd. The automatic selection is done by locating the point on the plot of normalized rank vs. normalized distance in $\widetilde{G}_0$ at which the slope is $45°$. Rank is computed by sorting the edge values in $\widetilde{G}_0$; normalized rank is rank normalized to range from 0 to 1. Distance is given by the actual edge value in $\widetilde{G}_0$; normalized distance is distance normalized to range from 0 to 1. For real-world datasets, we have found the normalized rank vs. distance curve to often provide a good cut-off for the average-link clustering when the slope is $45°$ (this threshold often falls between .9 and .95 in terms of normalized rank).

4. EXAMPLES

We have benchmarked OpenOrd on several datasets. As mentioned in Section 2, we have used a microarray gene expression expression experiment generated in the study of the cell cycle in yeast.[35] This dataset has been previously tested with the VxOrd precursor to OpenOrd.[22,27] The data consists of simultaneous measurements of 6,147 genes over 18 time points. A graph structure was imposed on the data by taking for each gene the top 10 genes most highly correlated over time. This produced a graph with 6,147 nodes and 61,646 edges. We use the yeast dataset to demonstrate the effect of edge-cutting and the use of parallel computation with OpenOrd.

We have also used an incarnation of the Swiss roll dataset,[29,30] with a sample of 20,000 points and a graph imposed using the 20 nearest neighbors of each point. We used the Swiss roll dataset to examine the multilevel capabilities of OpenOrd. In addition, we have used a large dataset from the Wikimedia foundation (http://www.wikimedia.org) to further test the multilevel capabilities. This dataset was collected and processed by B. Herr *et al.*[36] using Wikipedia articles from

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

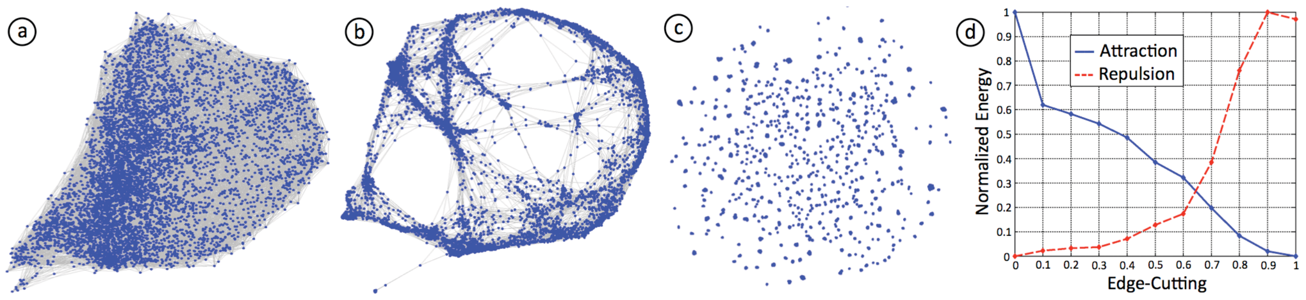

Figure 2. Edge-cutting on yeast. Shown here from left to right we have layouts of Spellman's yeast microarray dataset[35] with (a) no cutting (parameter value 0), (b) default cutting (parameter value 0.8), and (c) maximum cutting (parameter value 1). The effect of the edge-cutting parameter on the attractive and repulsive terms in the objective function from Equation (1) is shown in (d). The edge-cutting parameter is varied from 0 to 1 on the x-axis and a normalized energy value is shown on the y-axis. As described in Section 3.1, the attractive term decreases as edges are cut, allowing an increase in the repulsive term, and therefore encouraging clustering.

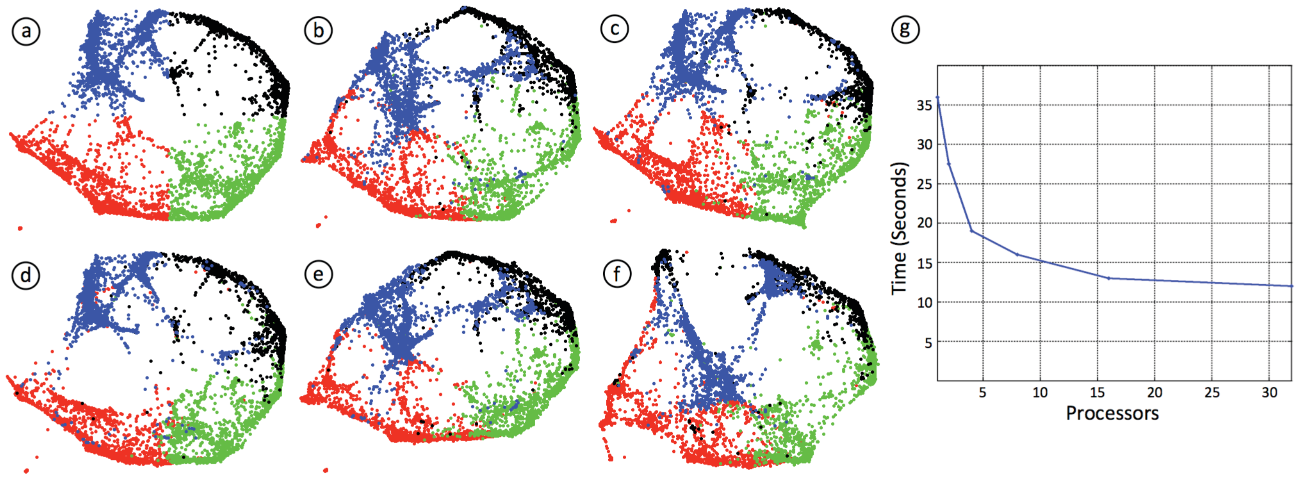

Figure 3. OpenOrd in parallel. Here we provide a qualitative comparison of the results of OpenOrd using 1, 2, 4, 8, 16, and 32 processors. In (a) we show the serial (1 processor) layout. We draw only nodes for clarity, and we color the layout by quadrant using blue, black, green, and red. In (b-f) we show the 2, 4, 8, 16, and 32 processor layouts, respectively. These layouts are colored using the same colors obtained from the single processor layout in (a). In (g) we show the computational speed up using more processors.

the year 2007 (see http://scimaps.org/maps/wikipedia). The dataset consists of 659,388 Wikipedia articles connected by 16,582,426 hyperlinks.

Finally, we have performed parameter testing in OpenOrd on many additional datasets. These datasets include 8,712 journals from the year 2003 taken from Thomson Scientific (http://scientific.thomson.com/isi) Institute of Scientific Information (ISI) Journal Citation Reports and linked with 98,705 edges;[37] a 32,776 document dataset focused on solid state lighting taken from ISI over the past 25 years and linked using 222,626 co-citation similarities;[38] a 218,716 article dataset taken from the first quarter of the year 2003 in the ISI database and linked using 1,821,976 co-citation similarities; and finally a 849,888 article dataset taken from the year 2004 in the ISI database and linked using 5,843,729 co-citation similarities. We used these datasets to examine the effects of the various parameters on different sizes and types of data.

4.1 Edge-Cutting

Edge-cutting in OpenOrd is specified using a fraction from 0 to 1. An edge-cutting value of 0 corresponds to the standard Frutcherman-Reingold layout algorithm (no cutting), while an edge-cutting value of 1 corresponds to aggressive cutting. Aggressive cutting promotes clustering but will not cut every edge. The default value for edge-cutting in OpenOrd is 0.8. We demonstrate the effects of edge-cutting on the layout of Spellman's yeast data[35] in Figure 2.

To produce the plot in Figure 2(d), we computed the total attractive term $\sum_{i,j}(w_{ij}d(\mathbf{x}_i,\mathbf{x}_j)^2))$ and the total repulsive term $\sum_i D_{\mathbf{x}_i}$ for 11 layouts of the yeast data, using edge-cutting parameter values $0, 0.1, \ldots, 0.9, 1$. The plots compares

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

the attractive and repulsive terms normalized to lie between 0 and 1. As discussed in Section 3.1, the attractive term decreases as edges are cut, thus allowing an increase in the repulsive term, and therefore encouraging clustering. As was also discussed in Section 3.1, cutting the long edges provides additional white space in the drawings. These effects are apparent in Figure 2(a-c).

4.2 Serial vs. Parallel

OpenOrd can be run in either serial or parallel mode. The algorithm in either case is the same, but the results in parallel are not guaranteed to be identical to the results in serial. Hence our first test of OpenOrd in parallel was to assess the potential difference between the two modes. For this test we again used Spellman's yeast data.[35] We ran OpenOrd on the yeast data with 1, 2, 4, 8, 16, and 32 processors. The 1 processor case is the serial version.

We compared the outputs of each run by computing a similarity metric $s_\varepsilon(U,V) \in [0,1]$, where U,V are two layouts of the same m-node dataset $\{\mathbf{x}_1,\ldots,\mathbf{x}_m\}$. The metric is computed by first constructing neighborhood incidence matrix $N_{U,\varepsilon}$ and $N_{V,\varepsilon}$, where $N_{\bullet,\varepsilon}$ is an $m \times m$ matrix $N_{\bullet,\varepsilon} = (n_{ij})$, with

$$n_{ij} = \begin{cases} 1 & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < \varepsilon \\ 0 & \text{otherwise} \end{cases} .$$

Now

$$s_\varepsilon(U,V) = \frac{N_{U,\varepsilon} \cdot N_{V,\varepsilon}}{\|N_{U,\varepsilon}\| \|N_{V,\varepsilon}\|},$$

where $N_{U,\varepsilon} \cdot N_{V,\varepsilon}$ is the dot product of $N_{U,\varepsilon}$ and $N_{V,\varepsilon}$, when both matrices are considered to be vectors of length m^2. This metric is bounded between 0 and 1, with larger values indicating greater similarity. It is a modification of a previously proposed similarity metric.[39]

For our calculations, we scaled each layout U and V to lie in the area $[0,1] \times [0,1]$ and used $\varepsilon = 0.1$. We obtained an average similarity of 0.72 for the parallel layouts with the serial layout. The average neighborhood size was 24 nodes. In addition to this metric, we compared the outputs of each run qualitatively, as shown in Figure 3(a-f). Our metric computations show that the local structure of the layout is preserved during the parallel calculations, and the qualitative results show that the global structure is also preserved. In addition to providing similar results, the parallel version of OpenOrd also offers a computational speed up, as shown in Figure 3(g).

4.3 Multilevel Layout

As discussed in previously, the Frutcherman-Reingold algorithm does not scale well to very large graphs. In addition to high running time, the algorithm will often confuse the global structure of the input, as demonstrated using the Swiss roll data in Figure 1(a,b). In the Swiss roll data, the drawing is correct on a local scale but tangled on a global scale. To achieve better results, OpenOrd uses multilevel graph coarsening,[13] with various modifications as described in Section 3.3.

We first demonstrate our results using coarsening by revisiting the Swiss roll dataset. Using coarsening with 9 levels and no edge-cutting, we obtained the globally correct layout of the Swiss roll shown in Figure 1(c).

Next we demonstrate the results of the multilevel approach by producing a layout of 659,388 Wikipedia articles from the year 2007. This layout was computed using 6 levels of recursion and is shown in Figure 4. The approach uses aggresive edge-cutting and clustering during coarsening, followed by repeated applications of the standard OpenOrd layout algorithm during refinement. Without coarsening, OpenOrd will draw the same graph as a uniformly dense, highly connected, and visually unappealing ball.

4.4 Parameter Testing

The layout algorithm used in OpenOrd has a host of parameters governing the behavior of the resulting layout, including random starting seed, simulated annealing optimization schedule, and the edge-cutting parameter. When using the layout algorithm in the multilevel mode, we must adjust these parameters according to the current stage of coarsening or refining so that layout continuity is preserved as we progress. In this section we benchmark our selection of parameters using a multitude of datasets.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

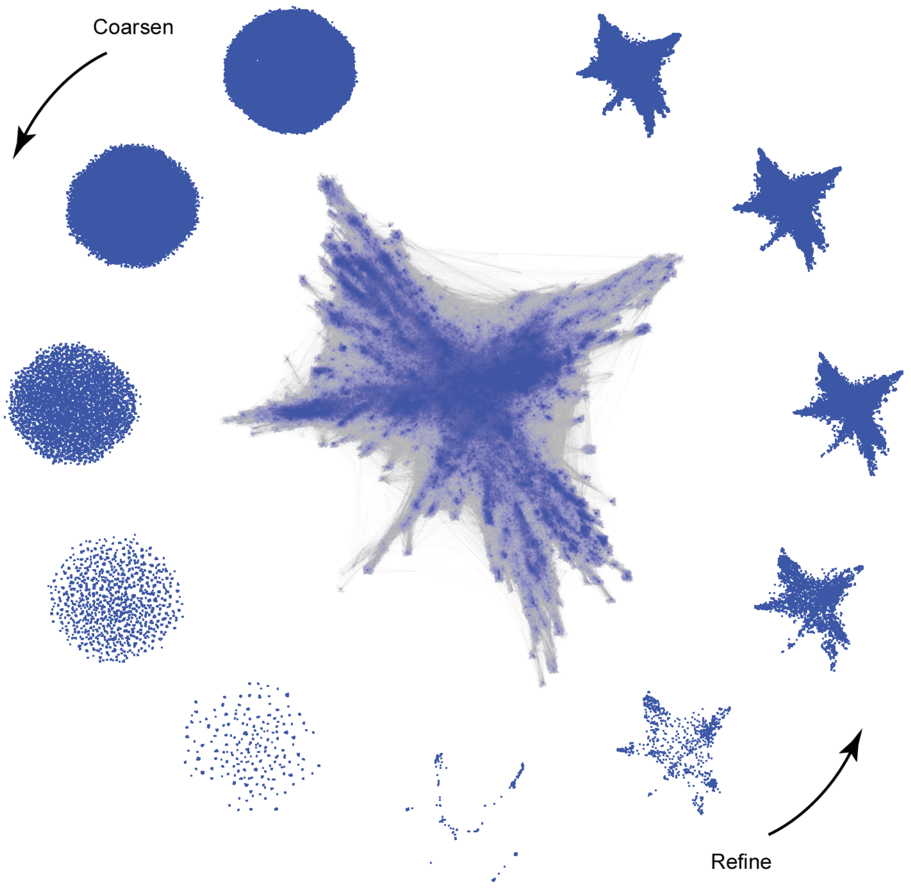

Figure 4. OpenOrd recursion. Here we show a layout of a 659,388 Wikipedia article dataset from the year 2007. The ultimate layout is shown in the center of the figure. The various layouts produced during the recursive procedure are shown around the outside edges of the final layout. Proceeding counterclockwise from 11 o'clock to 6 o'clock, we show layouts produced using the graph coarsening process. From 5 o'clock to 1 o'clock we show the layouts produced during refinement.

The results of our benchmarking efforts are shown in Table 1. Despite the diversity of datasets, we found that a common set of parameters provided good layouts using the multilevel mode. These parameters are provided as defaults in the code at www.cs.sandia.gov/∼smartin. The datasets, their sizes, and the level of recursion used are all shown in Table 1. Running times are also provided to give the user some indication of the effort required for a given dataset size. The times were obtained using a 3.4 GHz Intel Xeon workstation with 4 GB of RAM.

5. CONCLUSIONS

In this paper we have documented a collection of algorithms (OpenOrd) available for drawing large graphs. The focus of these algorithms is on real-world datasets such as those encountered in scientific literature and biological applications. Our approach is based on the Frutcherman-Reingold force-directed approach.[11] This approach has been refined using simulated annealing and grid based computation to obtain a practical algorithm which has been demonstrated to work well on real-world datasets.[23–28] However, the algorithm produces visually unappealing and globally inaccurate layouts for large datasets. We have addressed the problem of visual appeal using edge-cutting and the problem of global inaccuracy using a multilevel approach with average-link clustering to capture real-world global structure.

We have demonstrated the behavior of OpenOrd on various datasets, including yeast microarray data,[35] scientific journal data,[37] scientific literature data,[38] and Wikipedia articles.[36] We have determined appropriate default parameters

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

Parameter Testing				
Dataset	Nodes	Edges	Level	Time
Yeast	6,147	61,646	3	1:29
Journals	8,712	98,705	3	2:13
Swiss Roll	20,000	400,000	9	4:01
Solid State Lighting	32,776	222,626	4	7:16
Quarter Year ISI 2003	218,716	1,821,976	5	1:09:36
Wikipedia	659,388	16,582,426	6	3:39:23
Full Year ISI 2004	849,888	5,843,729	7	3:40:23

Table 1. Parameter testing. Different datasets used to benchmark default parameters for the multilevel version of OpenOrd are listed in the first column. The second and third columns contain the dataset size, the fourth column contains the level used, and the fifth column shows the time required to run on a workstation (hours:mins:seconds).

that can be used on differently sized datasets and have provided those defaults in our open source code, available at www.cs.sandia.gov/∼smartin. Finally, the algorithm has been adapted for use on parallel computers for extremely large datasets.

ACKNOWLEDGMENTS

The authors thank B. Herr *et al.* for providing the Wikipedia article data. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000. This work has been supported by Sandia's Computer Science Research Fund.

REFERENCES

[1] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G., [*Graph Drawing Algorithms for the Visualization of Graphs*], Prentice Hall (1999).

[2] Jünger, M. and Mutzel, P., [*Graph Drawing Software*], Springer-Verlag (2004).

[3] Freeman, L. C., "Visualizing social networks," *Journal of Social Structure* **1**(1) (2000).

[4] Borner, K., Chen, C., and Boyack, K., "Visualizing knowledge domains," in [*Annual Review of Information Science and Technology*], Cronin, B., ed., **37**, ch. 5, 179–255, American Society for Information Science and Technology, Medford, NJ (2003).

[5] Shiffrin, R. and Borner, K., "Mapping knowledge domains," *Proc. Natl. Acad. Sci.* **101 suppl. 1**, 5183–5185 (2004).

[6] Wolff, A., "Drawing subway maps: A survey," *Informatik - Forschung und Entwicklung* **22**, 23–44 (2007).

[7] Wiese, K. C. and Eicher, C., "Graph drawing tools for bioinformatics research: An overview," in [*CBMS '06: Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*], 653–658, IEEE Computer Society, Washington, DC, USA (2006).

[8] Adai, A. T., Date, S. V., Wieland, S., and Marcotte, E. M., "LGL: Creating a map of protein function with an algorithm for visualizing very large biological networks," *Journal of Molecular Biology* **340**, 179–190 (2004).

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

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

[11] Frutcherman, T. and Reingold, E., "Graph drawing by force-directed placement," *Software-Practice and Experience* **21**, 1129–1164 (1991).

[12] Davidson, R. and Harel, D., "Drawing graphs nicely using simulated annealing," *ACM Trans. on Graphics* **15**, 301–331 (1996).

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

[13] Walshaw, C., "A multilevel algorithm for force-directed graph-drawing," *Journal of Graph Algorithms and Applications* **7**(3), 253–285 (2003).

[14] Harel, D. and Koren, Y., "A fast multi-scale method for drawing large graphs," *Journal of Graph Algorithms and Applications* **6**, 179–202 (2002).

[15] Hachul, S. and Junger, M., "Drawing large graphs with a potential-field-based multilevel algorithm," in [*Graph Drawing*], Pach, J., ed., 285–295, Springer Berlin (2005).

[16] Hu, Y. F., "Efficient and high quality force-directed graph drawing," *The Mathematica Journal* **10**, 37–71 (2005).

[17] Quigley, A. and Eades, P., "Fade: Graph drawing, clustering, and visual abstraction," in [*Graph Drawing*], Marks, J., ed., 197–210, Springer Berlin (2001).

[18] Noack, A., "An energy model for visual graph clustering," in [*Graph Drawing*], Liotta, G., ed., 425–436, Springer Berlin (2004).

[19] Frishman, Y. and Tal, A., "Multi-level graph layout on the gpu," *IEEE Transactions on Visualization and Computer Graphics* **13**, 1310–1319 (2007).

[20] Godiyal, A., Hoberock, J., Garland, M., and Hart, J. C., "Rapid multipole graph drawing on the gpu," in [*Graph Drawing*], Tollis, I. G. and Patrignani, M., eds., 90–101, Springer Berlin (2009).

[21] Davidson, G., Hendrickson, B., Johnson, D., Meyers, C., and Wylie, B., "Knowledge mining with VxInsight: Discovery through interaction," *Journal of Intelligent Information Systems* **11**, 259–285 (1998).

[22] Davidson, G., Wylie, B., and Boyack, K., "Cluster stability and the use of noise in interpretation of clustering," in [*IEEE Symposium on Information Visualization (INFOVIS)*], 23–30 (2001).

[23] Boyack, K., Wylie, B., and Davidson, G., "Domain visualization using VxInsight for science and technology management," *Journal of the American Society for Information Science and Technology* **53**(9), 74–774 (2002).

[24] Boyack, K., "Mapping knowledge domains: Characterizing PNAS," *Proc. Natl. Acad. Sci.* **101 suppl. 1**, 5192–5199 (2004).

[25] Boyack, K., Klavans, R., and Borner, K., "Mapping the backbone of science," *Scientometrics* **64**(3), 351–374 (2005).

[26] Kim, S., Lund, J., Kiraly, M., Duke, K., Jiang, M., Stuart, J., Eizinger, A., Wylie, B., and Davidson, G., "A gene expression map for Caenorhabditis elegans," *Science* **293**(5537), 2087–2092 (2001).

[27] Werner-Washburne, M., Wylie, B., Boyack, K., Fuge, E., Galbraith, J., Weber, J., and Davidson, G., "Comparative analysis of multiple genome-scale data sets," *Genome Research* **12**(10), 1564–1573 (2002).

[28] Wilson, C., Davidson, G., Martin, S., Andries, E., Potter, J., Harvey, R., Ar, K., Xu, Y., Kopecky, K., Ankerst, D., Gundacker, H., Slovak, M., Mosquera-Caro, M., Chen, I.-M., Stirewalt, D., Murphy, M., Schultz, F., Kang, H., Wang, X., Radich, J., Appelbaum, F., Atlas, S., Godwin, J., and Willman, C., "Gene expression profiling of adult acute myeloid leukemia identifies novel biologic clusters for risk classification and outcome prediction," *Blood* **108**, 685–696 (2006).

[29] Tenenbuam, J. B., de Silva, V., and Langford, J. C., "A global geometric framework for nonlinear dimensionality reduction," *Science* **290**, 2319–2323 (2000).

[30] Roweis, S. and Saul, L., "Nonlinear dimensionality reduction by locally linear embedding," *Science* **290**, 2323–2326 (2000).

[31] Davidson, G., Martin, S., Boyack, K., Wylie, B., Martinez, J., Aragon, A., Werner-Washburne, M., Mosquera-Caro, M., and Willman, C., "Robust methods in microarray analysis," in [*Genomics and Proteomics Engineering in Medicine and Biology*], Akay, M., ed., 99–130, Wiley IEEE (2007).

[32] Hendrickson, B. and Leland, R., "A multilevel algorithm for partitioning graphs," in [*Proc. Supercomputing '95, San Diego*], ACM Press (1995).

[33] King, B., "Step-wise clustering procedures," *J. Am. Stat. Assoc.* **69**, 86–101 (1967).

[34] Jain, A. K., Murty, M. N., and Flynn, P. J., "Data clustering: A review," *ACM Computing Surveys* **31**(3), 264–323 (1999).

[35] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B., "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell* **9**, 3273–3297 (1998).

[36] Herr, B., Holloway, T., and Borner, K., "An emergent mosaic of Wikipedian activity." International Workshop and Conference on Network Science (2007).

[37] Boyack, K., "Using detailed maps of science to identify potential collaborations," *Scientometrics* **79**, 27–44 (2009).

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms

[38] Boyack, K., Tsao, J., Miksovic, A., and Huey, M., "A recursive process for mapping and clustering literatures: International trends in solid state lighting," *International Journal of Technology Transfer and Commercialization* **8**, 51–87 (2009).

[39] Ben-Hur, A., Elisseeff, A., and Guyon, I., "A stability based method for discovering structure in clustered data," in [*Pacific Symposium on Biocomputing*], 6–17 (2002).

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 08/20/2013 Terms of Use: http://spiedl.org/terms