Eric Reckwerdt

A Linear Algebraic Algorithm for Graph Drawing

This paper will give a brief overview of the realm of graph drawing, followed by a linear algebraic approach, ending with an example of our algorithm.

1 Graph Visualization

A graph, at the most basic level, is a collection of vertices or points connected together by a collection of edges or lines. They can be used for countless purposes to represent various kinds of relations between data, allowing a viewer to compactly view the relationships between elements. The data could also be shown in a table or encoded in a matrix, but the information is much harder to understand and some simple properties of a graph would be nigh-impossible to see. Concepts of symmetry and circuits are obvious when seen in a graph but hard to grasp when the data is solely in table form.

The visual representation, then, should be a high priority when dealing with graphs. The difference between a readable graph and one that hides important information is surprisingly easy to categorize. Di Battista et al.[1] lay out eleven aesthetics of a readable graph. Many of these are related to each other. For a typical graph, most of these cannot happen concurrently, but many can be minimized. The properties that we want to minimize are:

1. the number of crossing edges,
2. the total area of the drawing,
3. the sum of the lengths of the edges,
4. the maximum length of an edge,
5. the variance of edge lengths, i.e. trying to have uniform edge length,
6. the total number of bends in a graph,
7. the maximum number of bends on one edge,
8. the variance of edge bends,
9. and the aspect ratio, for better display on a screen.

Besides these we also want to display symmetries and maximize the smallest angle between two incident edges.

Arranging a small graph on paper to meet these criteria can be fairly simple, but when we start working with 100's and 1000's of vertices and associated edges, our task becomes very difficult. We must then rely on automated methods to ease our tasks, letting a computer layout our graph, tweaking the final image a little by hand if necessary.

Each algorithm has its own pros and cons, and is good at achieving a set of aesthetics while poor at fulfilling others. For each graph we must choose the algorithm that emphasizes the important information in the graph. For instance, if our graph is the intended layout of an electrical circuit board, we would want to minimize edge bends and edge length, and it would not make sense to try to maximize the smallest angle between incident edges and displaying symmetry is neither good nor bad.

Many of the algorithms used to visualize graphs are combinatorial in nature. They take fundamental properties of the graph and systematically lay out vertices in a certain order which produces a visually nice graph. These are typically type-dependent, since an algorithm might exploit the tree structure of a binary tree, and would not be applicable on a graph with cycles in it. Some are applicable to a wide range of graphs, working better for some, worse for others. Many divide-and-conquer techniques are this way.

Another type-independent technique is the force-based approach, which uses basic physics, viewing edges as springs with an ideal tension, and letting the system settle to a stable configuration. These algorithms create symmetric and balanced graphs, but will not minimize edge crossings. Some of these techniques use linear algebra in the implementation of the algorithm, but mainly for bookkeeping.

2 Spectral Distance Embedding

The approach that we will dissect is one that tries to approximate the graph-theoretical distances of the vertices. It was designed by Ali Civril, Malik Magdon-Ismail and Eli Bocek-Rivele at the Rensselaer Polytechnic Institute in Troy, New York. They described their technique in a paper entitled **SDE: Graph Drawing Using Spectral Distance Embedding**.[2] Similar to a forced-base approach (which would try to create uniform edge length, so each spring edge was near its ideal length), their method creates symmetric, balanced graphs by exploiting linear algebra. It uses the spectral decomposition of a distance matrix to return coordinates that estimate (or, in the case that the graph is embeddable, exact) the said distance matrix.

First we will define a few things that will be necessary for the algorithm to work. Then we will prove that the algorithm gives us coordinates that approximate the graph-theoretical distances.

Theorem 1 *Spectral Value Decomposition*

Suppose A is a Hermitian matrix of size n, rank r with eigenvalues $\lambda_1, \ldots, \lambda_n$ and associated orthonormal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$. Then $A = \sum_{k=1}^{n} \lambda_k \mathbf{u}_k \mathbf{u}_k^t$.

Proof

Since A is Hermitian, $A = UDU^*$, where $U = [\mathbf{u}_1 | \ldots | \mathbf{u}_n]$, $\mathbf{u}_i$ forming an orthonormal basis for $\mathbb{R}^n$, thus U is unitary, and D is the diagonal matrix with λ_i on the diagonal. Then we can apply Rank One Decomposition, so $A = \sum_{k=1}^{n} A_k$, $A_k = \lambda_k \mathbf{x}_k \mathbf{y}_k^t$, where $\mathbf{x}_k$ is the k^{th} column of U and y_k is the

k^{th} row of U^{-1}. But the k^{th} column of U is $\mathbf{u}_k$, so $\mathbf{x}_k = \mathbf{u}_k$ and $U^{-1} = U^t$, so $\mathbf{y}_k = \mathbf{u}_k$. Thus $A = \sum_{k=1}^{n} \lambda_k \mathbf{u}_k \mathbf{u}_k^t$. (see [3] for rank-one decomposition, etc).

Definition 1 *The rank-d approximation to A with respect to the spectral norm is* $A_d = \sum_{k=1}^{d} \lambda_k \mathbf{u}_k \mathbf{u}_k^t$, *where* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ *and* $\mathbf{u}_i$ *is the eigenvector associated with* λ_i.

Definition 2 *The graph-theoretical distance between two vertices* v_i *and* v_j *is the shortest number of edges that must be transversed to move from one vertex to the other.*

Definition 3 *A projection matrix A is an* $n \times n$ *matrix that fulfills two properties: It is symmetric and idempotent, ie* $A^2 = A$

In our proof we will use the projection matrix $\gamma = I_n - \frac{1}{n} 1_n 1_n^t$ where

$$1_n = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \text{ so } \gamma = \begin{bmatrix} 1 - \frac{1}{n} & -\frac{1}{n} & \dots & -\frac{1}{n} \\ -\frac{1}{n} & 1 - \frac{1}{n} & \dots & -\frac{1}{n} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{n} & -\frac{1}{n} & \dots & 1 - \frac{1}{n} \end{bmatrix}.$$

γ is clearly symmetric, and $\gamma^2 = \gamma$, thus γ is a projection matrix. Another nice property of γ is that $1_n^t \gamma = \gamma 1_n = \mathbf{0}$.

Theorem 2 *Spectral Distance Embedding*

Given a graph $G = (V, E)$ *with n nodes, let* $V = \{v_1, v_2, \dots, v_n\}$ *be the set of vertices and E be the set of edges between V. Let* $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^t$ *where* $\mathbf{x}_i$ *is the set of planar coordinates of* v_i. *Then* $XX^t \approx M_2$ *where* M_2 *is the rank-2 approximation of a Hermitian matrix of size n.*

Proof Following Civril et al.'s construction of the algorithm,

Let D be a size n square matrix such that $[D]_{ij}$ is the square of the graph-theoretical distance between v_i and v_j. Then what we want is $\|\mathbf{x}_i - \mathbf{x}_j\|^2 \approx [D]_{ij}$. Expanding our equation, we see that

$$\mathbf{x}_i^2 + \mathbf{x}_j^2 + 2\mathbf{x}_i \mathbf{x}_j \approx [D]_{ij}. \tag{1}$$

If we want to interpret this equation in terms of matrices, let

$$Y = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} Q = \begin{bmatrix} \|\mathbf{x}_1\|^2 \\ \|\mathbf{x}_2\|^2 \\ \vdots \\ \|\mathbf{x}_n\|^2 \end{bmatrix} \text{ and } 1_n = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Then the $n \times 2$ matrix Y is the matrix of vector positions (We're working in two dimensions, but the algorithm is applicable for any desired dimension), Q

is the vector of squared norms of the vector positions, and 1_n is the vector of n ones.

So our equation can be rewritten as $[Q1_n^t]_{ij} + [1_n Q^t]_{ij} - 2[YY^t]_{ij} \approx [D]_{ij}$, then

$$Q1_n^t + 1_n Q^t - 2YY^t \approx D.$$

If we multiply our equation on both sides by the projection matrix γ that we discussed earlier, we get

$$\begin{aligned} \gamma Q 1_n^t \gamma + \gamma 1_n Q^t \gamma - \gamma 2 Y Y^t \gamma &\approx \gamma D \gamma \\ \Rightarrow \gamma 2 Y Y^t \gamma &\approx \gamma D \gamma \\ \Rightarrow \gamma Y (\gamma Y)^t &\approx -\frac{1}{2} \gamma D \gamma \end{aligned}$$

since $1_n^t \gamma = \gamma 1_n = 0$ and $\gamma = \gamma^t$.

To simplify this, let us define $X = \gamma Y = (Y - \frac{1}{n} 1_n 1_n^t Y)$. So X is a vector of coordinates of V, each translated by $\frac{1}{n} 1_n 1_n^t Y$. But we don't care if the coordinates have been translated, since we only care about their relationship to each other, which has been preserved. So finding X is just as good as finding Y to us, which, as you remember, is the point of all this. Anyway, if we define $M = -\frac{1}{2} \gamma D \gamma$, then we have a translated distance squared matrix.

Notice that X has rank 2, since it's an $n \times 2$ matrix. We hope that M would have rank 2, which would be the case if D were a true Euclidean distance matrix. Since it's the graph-theoretical distance matrix, if D is not embeddable in $\mathbb{R}^2$, M might have a rank greater than 2. So we have to use the rank-2 approximation of M, M_2. Thus

$$XX^t = M_2.$$

q.e.d.

We now see that to retrieve X, all we need to create is D and γ, and find the rank-2 approximation of $-\frac{1}{2} \gamma D \gamma$. Then $X = [\sqrt{\lambda_1} \mathbf{u}_1 | \sqrt{\lambda_2} \mathbf{u}_2]$.

3 An Example

In order to see this clearly, let's do an example. Take the eight vertex graph G in figure one. This is not a bad graph, by any means, but I feel like we might know this graph by a different name if it was arranged slightly differently. Let's see what our algorithm gives us.

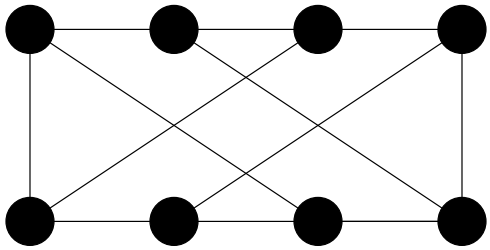

Figure 1: A graph G of 8 vertices.

To start, we need to compute D. Let D' be the graph-theoretical distance matrix, then $[D]_{ij} = [D']_{ij}^2$. So

$$D' = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 & 1 & 2 & 3 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 & 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 & 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 2 & 1 & 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 2 & 3 & 2 & 1 & 0 \end{bmatrix} \text{ and } D = \begin{bmatrix} 0 & 1 & 4 & 9 & 4 & 1 & 4 & 1 \\ 1 & 0 & 1 & 4 & 1 & 4 & 9 & 4 \\ 4 & 1 & 0 & 1 & 4 & 9 & 4 & 1 \\ 9 & 4 & 1 & 0 & 1 & 4 & 1 & 4 \\ 4 & 1 & 4 & 1 & 0 & 1 & 4 & 9 \\ 1 & 4 & 9 & 4 & 1 & 0 & 1 & 4 \\ 4 & 9 & 4 & 1 & 4 & 1 & 0 & 1 \\ 1 & 4 & 1 & 4 & 9 & 4 & 1 & 0 \end{bmatrix}$$

Since $n = 8$,

$$\gamma = \begin{bmatrix} \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & \frac{7}{8} \end{bmatrix}$$

and

$$M = -\frac{1}{2}\gamma D\gamma = \begin{bmatrix} 1.5 & 1 & -.5 & -3 & -.5 & 1 & -.5 & 1 \\ 1 & 1.5 & 1 & -.5 & 1 & -.5 & -3 & -.5 \\ -.5 & 1 & 1.5 & 1 & -.5 & -3 & -.5 & 1 \\ -3 & -.5 & 1 & 1.5 & 1 & -.5 & 1 & -.5 \\ -.5 & 1 & -.5 & 1 & 1.5 & 1 & -.5 & -3 \\ 1 & -.5 & -3 & -.5 & 1 & 1.5 & 1 & -.5 \\ -.5 & -3 & -.5 & 1 & -.5 & 1 & 1.5 & 1 \\ 1 & -.5 & 1 & -.5 & -3 & -.5 & 1 & 1.5 \end{bmatrix}$$

We can see that M is still Hermitian, which is good. Now to find the eigenvalues, we can put it through Octave, Mathematica, or another similar

calculator to get such values. We see that M has rank 6, so we will have to find the rank-2 approximation of M. The eigenvalues of M are 6, 2, and 0. So $\lambda_1 = \lambda_2 = 6$.

Two orthogonal eigenvectors for $\lambda = 6$ are $\mathbf{u}_1 = \begin{bmatrix} 0 \\ -.57630 \\ -.25807 \\ 0 \\ -.28523 \\ .57734 \\ -.29210 \\ -.28523 \end{bmatrix}, \mathbf{u}_2 = \begin{bmatrix} -.61237 \\ -.20412 \\ .20412 \\ .61237 \\ .20412 \\ -.20412 \\ .20412 \\ -.20142 \end{bmatrix}$

So we can now retrieve X.

$$X = [\sqrt{(}\lambda_1)\mathbf{u}_1 | \sqrt{(}\lambda_2)\mathbf{u}_2] = \begin{bmatrix} 0 & -1.5 \\ -1.41165 & -.5 \\ -.63214 & .5 \\ 0 & 1.5 \\ -.77951 & .5 \\ .63214 & -.5 \\ 1.41165 & .5 \\ .77951 & -.5 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \\ \mathbf{x}_7 \\ \mathbf{x}_8 \end{bmatrix}$$

Just to check, $XX^t = M_2$, and $\|\mathbf{x}_2 - \mathbf{x}_7\| = 2.99517 \approx 3 = [D']_{27}$, which was the criteria we were hoping to achieve. We are off by .005 there, but that's not too bad. The visualization then ends up looking like figure 2, a slightly squished cube.

Is this visualization better than the one in figure 1? It lets us see a greater number of symmetries, and we can abstract the 2d figure into a cube, which we naturally understand. So yes, it probably is. The graph could use some tweaking to get it looking better and less squished, but the algorithm gives us a start to work from. So overall, the algorithm is effective and simple, if one has the computing power.

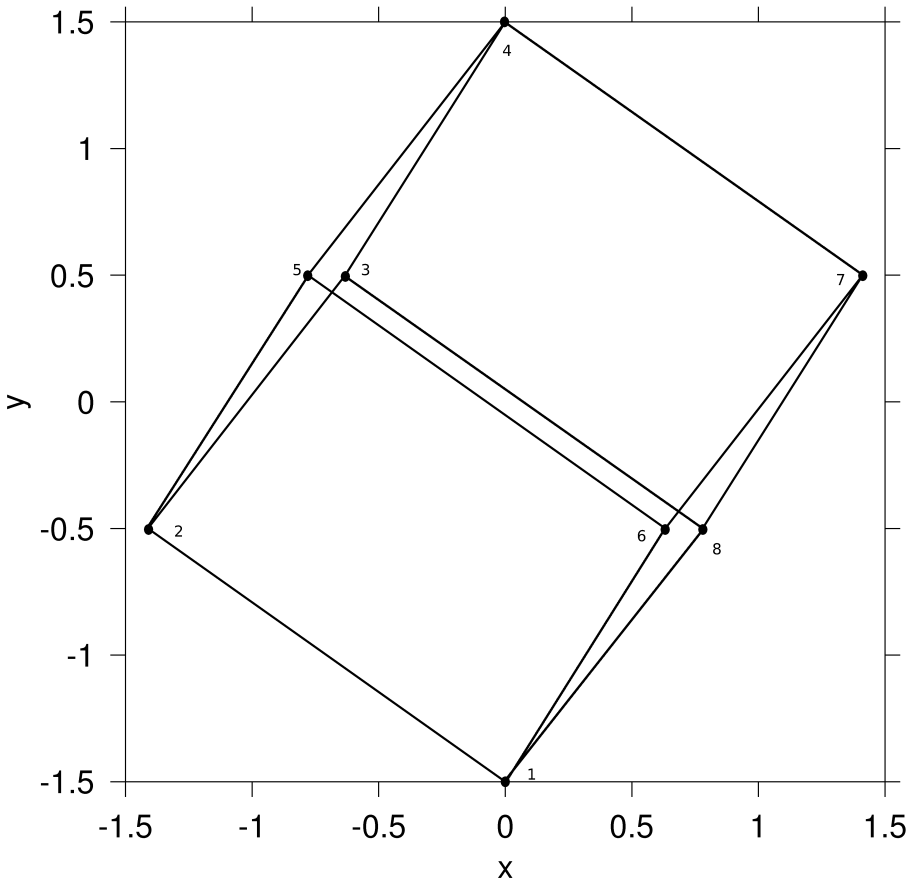

Figure 2: The new visualization of our graph.

References

[1] P. E. Ioannis G. Tollis, Giuseppe Di Battista and R. Tamassia, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, Inc., 1999.

[2] A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele, "Sde: Graph drawing using spectral distance embedding," http://citeseer.ist.psu.edu/751162.html.

[3] R. Beezer, *a first course in Linear Algebra*. self-published, ver. 1.08, 2007.