Graph drawing in spectral layout*

Maureen Gallagher    Colleen Tygh    John Urschel    Ludmil Zikatanov

Beginning: July 18, 2013; Today is: October 12, 2013

1 Introduction

Our research focuses on the use of spectral graph drawing techniques (see [5] and [3]) and other
mathematical tools to create a two-dimensional (2D) representation of the graph of a sparse matrix.
The actual graph drawing then amounts to displaying a particular arrangement of the vertices and
edges of a given graph in two dimensions. We use a spectral layout that provides coordinates using
the second and third eigenvectors of the associated graph Laplacian. To compute the eigenvectors
and eigenvalues, we implement a subspace correction method proposed in [4] and further analyzed
in [1].

1.1 Some graph drawing applications

Several important applications of graph drawing emerge from the analysis of networks, including
those evident in social media sites. In particular, drawing graphs of networks can aid the following
tasks:

• Exploration of data: Displaying nodes and ties in various layouts and attributing color, size
and other properties to nodes allows for the better visualization and analysis of data.

• Analyzing collaboration graphs:
  – Vertices represent participants or people.
  – Edges join two participants if a collaborative relationship exists between them.
  – Edge weights denote positive and negative relationships.

• Analysis of sociograms: Digraphs that diagram the structures and patterns of group interac-
tions facilitate the study of choices or preferences within a group.

With the quick and recent expansion of social networking, graph drawing is gaining a great deal of
attention. Some of the currently available graph-drawing software includes the following:

• Wolfram Alpha: This online service can connect to Facebook and make charts of friend
connections and relationships.

• Weighted Spectral Distribution: This MATLAB tool applies to undirected graphs.

*These notes are based on the 2013 summer research as a part of the Computational Mathematics for Under-
dergraduate Students in the Department of Mathematics, Penn State, University Park, PA, 16802, USA; see
http://sites.psu.edu/cmus2013/ for details
LTAS Technologies, Inc.: Among many other applications, this web identity search tool for Facebook finds the degree of separation between any two Facebook users and displays it in a graph.

2 Graph Theory: Preliminaries

We consider a general combinatorial graph \( G = (V, E) \), where

- \( V \) is the set of vertices: a set of discrete objects (but usually isomorphic to \( \{1, \ldots, n\} \)).
- \( E \) is the set of edges: ordered or unordered pairs of vertices.

We will consider two different types of graphs: undirected graphs and directed graphs. In an undirected graph for an edge \( (c, m) \), we always have \( (c, m) = (m, c) \), while in a directed graph \( (c, m) \neq (m, c) \). In addition, multigraphs allow multiple edges between vertices and loop graphs contain an edge that connects a vertex to itself. If the graph is weighted, one may associate a weight with every edge of a graph. Finally, a simple graph is an undirected graph that contains no loops or parallel edges. We will focus on weighted and unweighted simple graphs.

2.1 Characteristics of Graphs

All graphs have the following characteristics:

- order: \( V(G) = |V| \) (the number of vertices)
- size: \( E(G) = |E| \) (the number of edges)
- degree of a vertex: the number of edges incident to the vertex (gives an idea of how directly connected it is)
- connectedness: A graph is connected if for any two vertices \( v_1, v_2 \in V \) there exists a path from \( v_1 \) to \( v_2 \); a path being a sequence of vertices \( v_1 = w_1, w_2, \ldots, w_k = v_2 \) such that \( (w_{j-1}, w_j) \in E \), for all \( j = 2, \ldots, k \).
- partition of a graph: splitting \( V \) in two disjoint subsets, namely, \( V = V_1 \cup V_2 \) with \( V_1 \cap V_2 = \emptyset \). A graph is connected if for every partition there exists an edge connecting the two sets of vertices.

2.2 Matrices and graphs

We will now introduce several matrix representations of graphs, such as the adjacency and graph Laplacian matrices. In this and the following sections, we shall consider simple, undirected graphs (possibly weighted).

- Adjacency matrix \( A \in \mathbb{R}^{n \times n}, n = |V| \), where \( A_{ij} \) equals the number of edges between vertex \( i \) and vertex \( j \).
- graph Laplacian matrix \( L = D - A \), where \( D \) is the degree matrix of the graph \( (D = \text{diag}(d(v_1), d(v_2), \ldots, d(v_n))) \) and \( A \) is the adjacency matrix of the graph. Here, \( d(v_k) \) for a vertex \( v_k \in V \) is the number of edges incident with \( v_k \).
Show that for a simple undirected graph, \( L_{ij} = -1 \) for \( i \neq j \), and \( L_{ii} = d(i) \) (the degree of vertex \( i \)).

### 3 Graph drawing and eigenvalues

The graph Laplacian \( L \) of a simple graph is a symmetric matrix. If \( \sigma(L) \) is the set of eigenvalues of the symmetric matrix \( L \), we can conclude that all eigenvalues of \( L \) are real, that is, \( \sigma(L) \subset \mathbb{R} \). Moreover, since the row and column sums of a graph Laplacian are zero, we immediately obtain that \( L \mathbf{1} = 0 \), where \( \mathbf{1} = (1, \ldots, 1)^T \) (the constant vector). In other words, \( 0 \in \sigma(A) \).

On \( \mathbb{R}^n \) we have an inner (or scalar) product

\[
(u, v) = u_1v_1 + u_2v_2 + \ldots + u_nv_n = \sum_{j=1}^{n} u_jv_j. \tag{1}
\]

Using the inner product, with every matrix, such as \( L \), we associate a bilinear form:

\[
(Lu, v) = \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij}u_jv_i, \quad \text{for all } u \in \mathbb{R}^n, \ v \in \mathbb{R}^n.
\]

The fact that \( L \mathbf{1} = 0 \) implies that

\[
\sum_{j \neq i} L_{ij} = -L_{ii}, \quad \text{for all } i = 1, \ldots, n.
\]

The sum on the left side of the above identity is over all \( j \in \{1, \ldots, n\} \) such that \( j \neq i \).

**Homework** Given that \( L \mathbf{1} = 0 \), show that the above bilinear form can be written as:

\[
(Lu, v) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (-L_{ij})(u_i - u_j)(v_i - v_j).
\]

Further show that if all \( L_{ij} \leq 0 \), then \( (Lu, u) \geq 0 \) for all \( u \in \mathbb{R}^n \). Such an \( L \) is called symmetric positive semidefinite.

The extreme eigenvalues of \( L \) will be extreme values of the Rayleigh quotient if

\[
\sigma(L) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}, \quad \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n.
\]

\[
\lambda_1 = \min_{v \in \mathbb{R}^n} \frac{(Lv, v)}{(v, v)}, \quad \lambda_n = \max_{v \in \mathbb{R}^n} \frac{(Lv, v)}{(v, v)}.
\]

As we checked in the Homework, \( (Lv, v) \geq 0 \) and \( (L \mathbf{1}, \mathbf{1}) = 0 \). We conclude that \( \lambda_1 = 0 \) and the corresponding eigenvector is \( \mathbf{1} \). For a connected, undirected graph with positive edge weights it can be shown that 0 is a simple eigenvalue, namely, \( 0 = \lambda_1 < \lambda_2 \leq \lambda_2 \ldots \leq \lambda_n \).

Furthermore, it can be shown that if \( (\lambda_2, v_2) \) and \( (\lambda_2, v_3) \) are the eigenvalue and eigenvector pairs of \( L \), then

\[
\lambda_2 = \min_{v \in V_2} \frac{(Lv, v)}{(v, v)}, \quad \lambda_3 = \min_{v \in V_3} \frac{(Lv, v)}{(v, v)}, \tag{2}
\]

3
where $V_1$ is the subspace of vectors orthogonal to $1$, and $V_2$ is the subspace of vectors orthogonal to both $1$ and $v_2$. More precisely,

$$V_1 = \{ v \in \mathbb{R}^n, (v, 1) = 0 \},$$
$$V_2 = \{ v \in \mathbb{R}^n, (v, 1) = 0, (v, v_2) = 0 \}.$$

**Homework** Show this by proving the min-max principle (you will find this in [2]).

The spectral layout graph drawing uses $v_2$ and $v_3$ as coordinates of the vertices to display the graph in two dimensions. In other words, the vertex $i$ will have coordinates $(v_1,i,v_2,i) \subset \mathbb{R}^2$, and an edge is drawn between $i$ and $j$ if $(i,j) \in E$.

Thus, to utilize a spectral layout drawing, the problem is to find $(\lambda_2, v_2)$ and $(\lambda_3, v_3)$ which solve

$$Lv_2 = \lambda_2 v_2, \quad Lv_3 = \lambda_3 v_3, \quad (v_2, 1) = (v_3, 1) = 0, \quad (v_2, v_3) = 0.$$

This is equivalent to minimizing the Rayleigh quotient, written as $\lambda = \min_{(x,1)=0} \frac{(Lv, v)}{(v, v)}$.

We would like to emphasize that instead of the scalar product given in (1), in all of the Rayleigh quotients we use $(Dv, v)$ instead of $(v, v)$. In such a case we will be solving the so-called generalized eigenvalue problem

$$Lv_2 = \lambda_2 Dv_2, \quad Lv_3 = \lambda_3 Dv_3, \quad (Dv_2, 1) = (Dv_3, 1) = 0, \quad (Dv_2, v_3) = 0. \quad (3)$$

Indeed, it is easy to verify that all considerations below are valid with this scalar product.

**Homework** Using the definition of scalar product in [2], show that

$$(Dv, w) = \sum_{i=1}^{n} d(i)v_i w_i$$

is indeed a dot product between $v$ and $w$. For $n = 2$, (two dimensions) draw the unit sphere, or the tips of all vectors $v \in \mathbb{R}^2$ satisfying $(Dv, v) = 1$ if: (a) $d(1) = d(2) = 1$; (b) $d(1) = 1$ and $d(2) = 10$.

### 3.1 Subspace correction method for eigenvalue problem

We consider the subspace correction method for eigenvalue problems (see [4], [1]). We begin by splitting $\mathbb{R}^n$ into the sum of subspaces,

$$\mathbb{R}^n = W_1 + W_2 + \cdots + W_n.$$

For each of the subspaces $W_k$ we assume

$$W_k = \text{Range}(P_k), \quad P_k \in \mathbb{R}^{n \times n_k}, \quad n_k = \dim W_k.$$

The algorithm visits each subspace and improves the current eigenvalue approximation from each of these subspaces. To explain this in more detail, let $y$ be a given approximation to the eigenvector $v_2$, with $(y, 1) = 0$. We want to improve the approximation $y$ by finding $y_1 \in W_1$ such that

$$y_1 = \arg \min_{z \in W_1} \frac{(L(y + z), (y + z))}{(y + z, y + z)}. \quad (4)$$
The notation arg min means that $y_1$ is the argument $z$ which minimizes the right side of the equation above. We now show how we can find $y_1$ by solving a $(n_i + 1) \times (n_i + 1)$ problem. We first compute the numerator on the right hand side of (4):

\[(L(y + z), (y + z)) = (Ly + Lz, y + z) = (Ly, y) + (Ly, z) + (Lz, y) + (Lz, z)\]

We compute the denominator in a similar fashion. Denoting now

\[
\mu = \min_{z \in \mathbb{W}_1} \frac{(Ly, y) + 2(Ly, z) + (Lz, z)}{(y, y) + 2(y, z) + (z, z)},
\]

we note (as we also noted before) that $y_1 \in \mathbb{W}_1$ is the minimizer (i.e. the $z$ for which the minimum is achieved), namely

\[
\frac{(Ly, y) + 2(Ly, y_1) + (Ly_1, y_1)}{(y, y) + 2(y, y_1) + (y_1, y_1)} = \min_{z \in \mathbb{W}_1} \frac{(Ly, y) + 2(Ly, z) + (Lz, z)}{(y, y) + 2(y, z) + (z, z)}.
\]

We now try to write the right-hand side of the equation above as a Rayleigh quotient for some matrix. Recall that $W_1 = \text{Range}(P_1)$, and let us denote

\[\tilde{P}_1 = \begin{bmatrix}
\begin{array}{l}
\text{(n}_1\text{ columns)}
\end{array}
\end{bmatrix}
\begin{array}{c}
P_1 \\
y
\end{array}
\begin{array}{c}
\text{(n}_1\text{ + 1 columns)}
\end{array}
\end{bmatrix}
\text{n rows, } n_1 = \text{dim } W_1.
\]

A direct computation then shows that

\[
\tilde{L}_1 = \tilde{P}_1^T L \tilde{P}_1 = \begin{bmatrix}
P_1^T & 0 \\
y^T & 0
\end{bmatrix} L \begin{bmatrix}
P_1 \\
y
\end{bmatrix} = \begin{bmatrix}
P_1^T & 0 \\
y^T & 0
\end{bmatrix} \begin{bmatrix}
LP_1 & Ly \\
y^T & 0
\end{bmatrix} = \begin{bmatrix}
P_1^T LP_1 & P_1^T Ly \\
y^T & 0
\end{bmatrix}
\]

Since $z \in \text{Range}(P_1)$, it follows that a $d_z \in \mathbb{R}^{n_1}$ exists such that

\[
\begin{bmatrix}
z \\
\end{bmatrix} = \begin{bmatrix}
P_1 \\
d_z
\end{bmatrix} \in \mathbb{R}^{n_1} \text{ exists such that }
\]

Thus, minimizing with respect to $z \in \mathbb{W}_1$ is the same as finding $d_z \in \mathbb{R}^{n_1}$. We then have

\[
\mu = \min_{z \in \mathbb{W}_1} \frac{(Ly, y) + 2(Ly, z) + (Lz, z)}{(y, y) + 2(y, z) + (z, z)}
\]

\[
e e \min_{d_z \in \mathbb{R}^{n_1}} \frac{(Ly, y) + 2(Ly, P_1 d_z) + (LP_1 d_z, P_1 d_z)}{(y, y) + 2(y, P_1 d_z) + (P_1 d_z, P_1 d_z)}
\]

\[
= \min_{d_z \in \mathbb{R}^{n_1}} \frac{(Ly, y) + 2(P_1^T Ly, d_z) + (P_1^T LP_1 d_z, d_z)}{(y, y) + 2(P_1^T y, d_z) + (P_1^T P_1 d_z, d_z)}.
\]

Introducing the matrix

\[
\tilde{I}_1 = \begin{bmatrix}
P_1^T \\
y^T
\end{bmatrix} \begin{bmatrix}
P_1 \\
y
\end{bmatrix} = \begin{bmatrix}
P_1^T P_1 & P_1^T y \\
y^T P_1 & y^T y
\end{bmatrix},
\]

and the vector

\[
\tilde{d}_z = \begin{bmatrix}
d_1 \\
1
\end{bmatrix} \text{ (n}_1\text{ + 1)},
\]

it is straightforward to check the following identities:

\[
(L_1 \tilde{d}_z, \tilde{d}_z) = (Ly, y) + 2(P_1^T Ly, d_z) + (P_1^T LP_1 d_z, d_z)
\]

\[
(I_1 \tilde{d}_z, \tilde{d}_z) = (y, y) + 2(P_1^T y, d_z) + (P_1^T P_1 d_z, d_z).
\]
Consequently, we have that
\[ \mu = \min_{\tilde{d}_z} \frac{(\tilde{L}_1 \tilde{d}_z, \tilde{d}_z)}{(I_1 \tilde{d}_z, \tilde{d}_z)}. \]

Hence, the minimizer of the above Rayleigh quotient, \( \tilde{d}_1 \), is a generalized eigenvector of the generalized eigenvalue problem
\[ \tilde{L}_1 \tilde{d}_1 = \mu \tilde{I}_1 \tilde{d}_1. \] (5)

So to find \( y_1 \), we need to solve the \((n_1 + 1) \times (n_1 + 1)\) eigenvalue problem \([5]\).

4 Implementation of the subspace correction method

In this section, we outline the algorithm for computing the eigenvectors. We first split the set of vertices of the graph into \( n_c \) non-overlapping subsets \( \{\Omega_k\}_{k=1}^{n_c} \), namely,
\[ \{1, \ldots, n\} = \bigcup_{k=1}^{n_c} \Omega_k, \quad \Omega_k \cap \Omega_j = \emptyset, \quad \text{whenever} \quad k \neq j. \]

We use a simple “greedy” algorithm to construct such a splitting.

<table>
<thead>
<tr>
<th>Algorithm 1: Greedy vertex splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set ( n_c = 0 ); \n</td>
</tr>
</tbody>
</table>

Next, we formalize the subspace correction algorithm as follows. In our computations, we use a modification that can simultaneously compute \( n_\lambda \) eigenvalues. In the tests we have done, we always set \( n_\lambda = 3 \). The number of eigenvalues we find on each subspace is \( n_m = (n_\lambda + 2) \), because the first \( n_\lambda \) eigenvalues of the subspace problem are more accurately computed.

We find the three smallest eigenvalues of \( Lv = \lambda Dv \) (0 = \( \lambda_1 \), \( \lambda_2 \), and \( \lambda_3 \)). Here, \( D \) is the degree matrix of the graph, \( L = D - A \) is the Laplacian matrix of the graph, and \( Y \) is the approximation of the eigenvectors, which we orthogonalize and normalize. We obtain the eigenvalues of the graph, and we use the corresponding eigenvectors to draw a representation of the graph. We also consider subspaces \( W_i = \text{Range}(P_i) \) subordinate to the partitioning provided by the clusters \( \Omega_i \). More precisely, let
\[ \Omega_i = \{j_1, \ldots, j_{n_i}\} \subset \{1, \ldots, n\}. \]

Then \( P_i \in \mathbb{R}^{n \times n_i} \) is defined as
\[ (P_i)_{j,\ell} \begin{cases} 1, & k = \ell \\ 0, & \text{otherwise}. \end{cases} \]

The algorithm below takes an initial guess \( Y \in \mathbb{R}^{n \times n_\lambda} \) and, as output, provides a new and improved approximation.
Algorithm 2: Schwartz-based method

\( Y \in \mathbb{R}^{n \times n_{\lambda}} \) is the current approximation to the first \( n_{\lambda} \) eigenvectors

\[
\text{for } i = 1 : n_c \text{ do }
\]

\[
\text{set } \tilde{P}_i = [P_i, Y] \\
\text{compute } \tilde{L}_i = \tilde{P}_i^T L \tilde{P}_i \\
\text{compute } \tilde{I}_i = \tilde{P}_i^T D \tilde{P}_i
\]

let \((\lambda_k, u_k), k = 1, \ldots, n_{\lambda}, 0 = \lambda_1 \leq \ldots \leq \lambda_{n_{\lambda}}\) be the eigenvector, eigenvalue pair of the eigenvalue problem \( \tilde{L}_k u_k = \lambda_k \tilde{I}_i u_k \)

Set \( Y = \tilde{P}_i [u_1, \ldots, u_{n_{\lambda}}] \) (\( Y \) is now the new approximation to the eigenvectors)

---

4.1 Adding a coarse “grid” correction

A coarser grid (coarser graph) is a simpler representation of a graph in which the vertices are the clusters \( \{\omega_k\}_{k=1}^{n_{c}} \). We consider two clusters connected if and only if a member of one cluster is connected to a member of the other.

A coarse grid correction provides faster and more accurate results because such a correction is "global," and vertices which could be "far" from each other are simultaneously corrected. In general, the algorithm can be run without changes by introducing a coarse space \( W_c = \text{Range}(P) \), \( \dim W_c = n_c \), and the \( k \)-th column of \( P \) equalling 1 at the vertices from \( \Omega_k \) and 0 otherwise. Clearly, \( P1_{n_c} = 1_n \), where \( 1_{n_c} = (1, \ldots, 1)^T \). We can now form

\[
\tilde{P} = [P, Y], \quad \tilde{L} = \tilde{P}^T L \tilde{P}, \quad \tilde{I} = \tilde{P}^T D \tilde{P}.
\]

However, if we attempt to solve in this subspace in the same way as we did before, it will not work; \( Y \) contains column \( 1 \), and we also have, as we mentioned earlier, \( P1_{n_c} = 1_n \). Note that

\[
Lx = \lambda Dx \tag{6}
\]

is well defined if \( D \) is invertible. If \( D \) is not invertible, \( D \) has a non-trivial null space and the eigenvalue problem \( Lx = \lambda Dx \) does not make sense in general. If \( D \) is invertible, then clearly any solution to \( Lx = \lambda x \) is a solution to the eigenvalue problem \( D^{-1}Lx = \lambda x \), and everything is well defined.
These observations are summarized as follows:

\[ \tilde{P} = [P, Y], \quad Y = \begin{bmatrix} 1 & * & \cdots & * \\ 1 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 1 & * & \cdots & * \end{bmatrix}, \quad P = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \]

We then have (with \( e_1 = [1, 0, 0]^T \)):

\[ \tilde{P}[1_{n_c}, -e_1] = [P, Y] [1_{n_c}, -e_1] = P \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - Y \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0. \]

We thus immediately conclude that \( \tilde{P} \) is not full rank, and, therefore, \( \tilde{P}^T D \tilde{P} \) is singular, and the eigenvalue problem on this subspace is not well posed (this is only due to the matrices we use). In order to eliminate this redundancy, we use the two columns of \( Y \) only when we do the coarse grid correction. In short, we only use the last two columns of \( Y \) to form \( \tilde{P} \). More precisely, we have,

if \( Y = [y_1, y_2, y_3] \), then set \( Z = [y_2, y_3] \) and \( \tilde{P} = [P, Z] \).

Another remark we would like to make here is that instead of the \( P \) which we described above, we also use in the algorithm the so-called “smoothed aggregation” \( P \), which is defined as

\[ P \leftarrow (I - D^{-1}L)P. \]

The results in the next section are obtained using this \( P \) for the coarse grid correction.

5 Numerical tests

In our tests, we used several adjacency matrices from The University of Florida Sparse Matrix Collection [http://www.cise.ufl.edu/research/sparse/matrices/].

![Figure 2: gridi_dual (left) and delaunay_n13 (right)]
Table 1: Comparison of a single level method (without coarse grid correction) and two level method (with coarse grid correction).

<table>
<thead>
<tr>
<th>Figure</th>
<th>Graph Name</th>
<th>Iterations (no coarse grid)</th>
<th>Iterations (coarse grid)</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>grid1_dual</td>
<td>127</td>
<td>7</td>
<td>224</td>
<td>420</td>
</tr>
<tr>
<td>3</td>
<td>delaunay_n13</td>
<td>&gt; 200</td>
<td>5</td>
<td>8192</td>
<td>24547</td>
</tr>
<tr>
<td>4</td>
<td>barth5</td>
<td>&gt; 200</td>
<td>5</td>
<td>15606</td>
<td>45878</td>
</tr>
<tr>
<td>5</td>
<td>big_dual</td>
<td>&gt; 200</td>
<td>5</td>
<td>30269</td>
<td>44929</td>
</tr>
<tr>
<td>6</td>
<td>Alemdar</td>
<td>194</td>
<td>7</td>
<td>6240</td>
<td>18168</td>
</tr>
<tr>
<td>7</td>
<td>fe_4elt2</td>
<td>&gt; 200</td>
<td>5</td>
<td>11143</td>
<td>32818</td>
</tr>
</tbody>
</table>

We ran our subspace correction algorithm both with and without the coarse grid correction. A comparison of the number of iterations can be seen in Table 1, and the corresponding graph drawings obtained using the two-level method (with coarse grid correction) are shown in Figures 2-4.

Figure 3: barth5 (left) and big_dual (right)

Figure 4: Alemdar (left) fe_4elt2 (right)
References


