Drawing Graphs by Eigenvectors: Theory and Practice*

Yehuda Koren

AT&T Labs – Research
Florham Park, NJ 07932
yehuda@research.att.com

Abstract. The spectral approach for graph visualization computes the layout of a graph using certain eigenvectors of related matrices. Some important advantages of this approach are an ability to compute optimal layouts (according to specific requirements) and a very rapid computation time. In this paper we explore spectral visualization techniques and study their properties from different points of view. We also suggest a novel algorithm for calculating spectral layouts resulting in an extremely fast computation by optimizing the layout within a small vector space.

Keywords: graph drawing, Laplacian, eigenvectors, Fiedler vector, force-directed layout, spectral graph theory

1 Introduction

A graph $G(V, E)$ is an abstract structure that is used to model a relation E over a set V of entities. Graph drawing is a standard means for visualizing relational information, and its ultimate usefulness depends on the readability of the resulting layout, that is, the drawing algorithm's ability to convey the meaning of the diagram quickly and clearly. To date, many approaches to graph drawing have been developed [2, 3]. There are many kinds of graph-drawing problems, such as drawing di-graphs, drawing planar graphs and others. Here we investigate the problem of drawing undirected graphs with straight-line edges. In fact, the methods that we utilize are not limited to traditional graph drawing and are also intended for general low dimensional visualization of a set of objects according to their pairwise similarities (see, e.g., Fig. 2).

We have focused on spectral graph drawing methods, which construct the layout using eigenvectors of certain matrices associated with the graph. To get some feeling, we provide results for three graphs in Fig. 1. This spectral approach is quite old, originating with the work of Hall [4] in 1970. However, since then it has not been used much. In

* An early and short version of this work appeared in [1].

fact, spectral graph drawing algorithms are almost absent in the graph-drawing literature (e.g., they are not mentioned in the two books [2, 3] that deal with graph drawing). It seems that in most visualization research the spectral approach is difficult to grasp in terms of aesthetics. Moreover, the numerical algorithms for computing the eigenvectors do not possess an intuitive aesthetic interpretation.

We believe that the spectral approach has two distinct advantages that make it very attractive. First, it provides us with a mathematically-sound formulation leading into an exact solution to the layout problem, whereas almost all other formulations result in an NP-hard problem, which can only be approximated. The second advantage is computation speed. Spectral drawings can be computed extremely fast as we show in Sec. 7. This is very important because the amount of information to be visualized is constantly growing rapidly.

Spectral methods have become standard techniques in algebraic graph theory; see, e.g., [5]. The most widely used techniques utilize eigenvalues and eigenvectors of the adjacency matrix of the graph. More recently, the interest has shifted somewhat to the spectrum of the closely related Laplacian. In fact, Mohar [6] claims that the Laplacian spectrum is more fundamental than this of the adjacency matrix.

Related areas where the spectral approach has been popularized include clustering [7], partitioning [8], and ordering [9]. However, these areas use discrete quantizations of the eigenvectors, unlike graph drawing, which employs the eigenvectors without any modification. Regarding this aspect, it is more fundamental to explore properties of graph-related eigenvectors in the framework of graph drawing.

In this paper we explore the properties of spectral visualization techniques, and provide different explanations for their ability to draw graphs nicely. Moreover, we consider a modified approach that uses what we will call *degree-normalized eigenvectors*, which have aesthetic advantages in certain cases. We provide an aesthetically-motivated algorithm for computing the degree-normalized eigenvectors. We also introduce a novel algorithm for computing spectral layouts that facilitates a significant reduction of running time by optimizing the layout within a small vector space. In addition to accelerating running time, we might also gain some aesthetical advantages compared to the more traditional spectral algorithms.

2 Basic Notions

A graph is usually written $G(V, E)$, where $V = \{1 \dots n\}$ is the set of n nodes, and E is the set of edges (we assume no self loops or parallel edges). Each edge $\langle i, j \rangle$ is associated with a non-negative weight w_{ij} that reflects the similarity of nodes i and j.

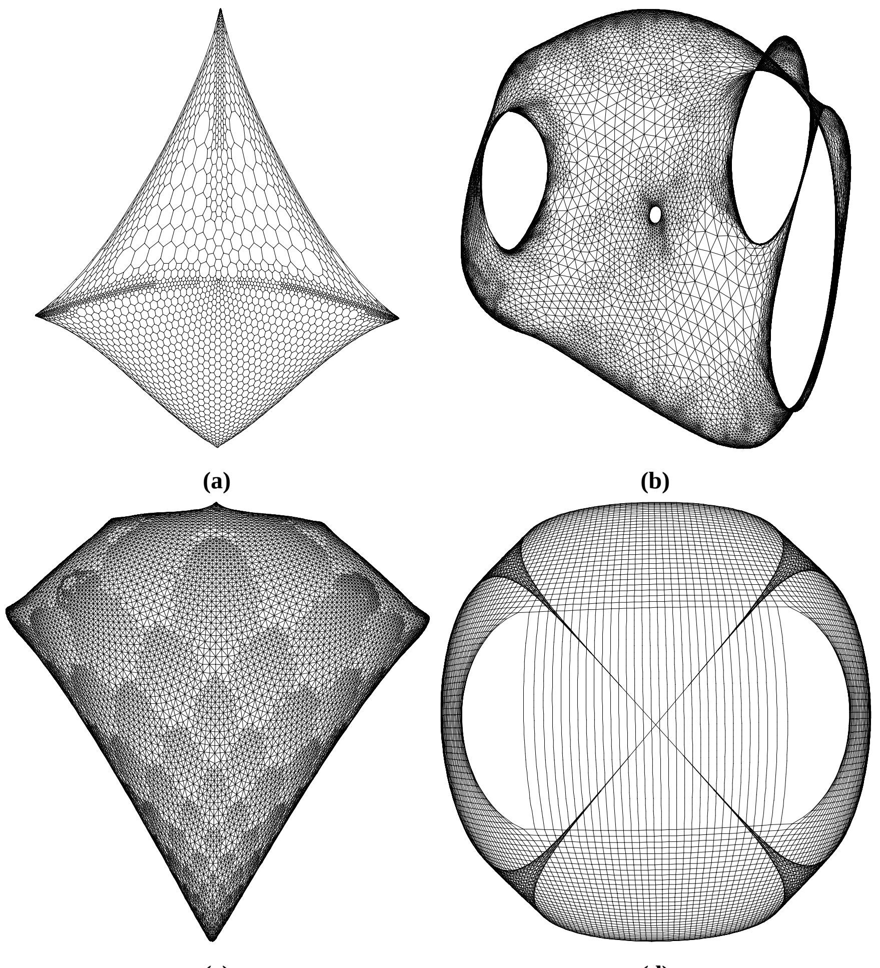

Fig. 1. Drawings obtained from the Laplacian eigenvectors. **(a)** The 4970 graph. $|V| = 4,970$, $|E| = 7,400$. **(b)** The 4elt graph [10]. $|V| = 15,606$, $|E| = 45,878$. **(c)** The Crack graph [10]. $|V| = 10,240$, $|E| = 30,380$. **(d)** A 100×100 folded grid with central horizontal edge removed. $|V| = 10,000$, $|E| = 18,713$.

Thus, more similar nodes are connected with "heavier" edges. For unweighted graphs, one usually takes uniform weights. Let us denote the neighborhood of i by $N(i) = \{j \mid \langle i, j\rangle \in E\}$. The degree of node i is $\mathrm{deg(i)} \overset{\mathrm{def}}{=} \sum_{\mathrm{j \in N(i)}} \mathrm{w_{ij}}$. Throughout the paper we have assumed, without loss of generality, that G is connected, otherwise the problem we deal with can be solved independently for each connected component.

A p-dimensional layout of the graph is defined by p vectors $x^1, \ldots, x^p \in \mathbb{R}^n$, where $x^1(i), \ldots, x^p(i)$ are the coordinates of node i. In most applications $p \leqslant 3$, but here we will not specify p so as to keep the theory general (however, always $p < n$). Let us denote by d_{ij} the Euclidean distance between nodes i and j (in the p-D layout), so $d_{ij} = \sqrt{\sum_{k=1}^{p}(x^k(i) - x^k(j))^2}$.

The *adjacency-matrix* of the graph G is the symmetric $n \times n$ matrix A^G, where

$$A^G_{ij} = \begin{cases} 0 & \langle i,j \rangle \notin E \\ w_{ij} & \langle i,j \rangle \in E \end{cases} \qquad i,j = 1, \ldots, n.$$

We will often omit the G in A^G.

The *Laplacian* is another symmetric $n \times n$ matrix associated with the graph, denoted by L^G, where

$$L^G_{ij} = \begin{cases} \mathrm{deg(i)} & i = j \\ 0 & i \neq j, \langle i,j \rangle \notin E \\ -w_{ij} & \langle i,j \rangle \in E \end{cases} \qquad i,j = 1, \ldots, n.$$

Again, we will often omit the G in L^G.

The Laplacian has many interesting properties (see, e.g., [6]). Here we state some useful features:

– L is a real symmetric and hence its n eigenvalues are real and its eigenvectors are orthogonal.
– L is positive semi-definite and hence all eigenvalues of L are non-negative.
– $1_n \overset{\mathrm{def}}{=} (1, 1, \ldots, 1)^T \in \mathbb{R}^n$ is an eigenvector of L, with the associated eigenvalue 0.
– The multiplicity of the zero eigenvalue is equal to the number of connected components of G. In particular, if G is connected, then 1_n is the only eigenvector associated with eigenvalue 0.

The usefulness of the Laplacian stems from the fact that the quadratic form associated with it is just a weighted sum of all pairwise squared edge-lengths:

Lemma 1. *Let L be an $n \times n$ Laplacian, and let $x \in \mathbb{R}^n$. Then*

$$x^T L x = \sum_{\langle i,j \rangle \in E} w_{ij}(x(i) - x(j))^2.$$

More generally, for p vectors $x^1, \ldots, x^p \in \mathbb{R}^n$ we have:

$$\sum_{k=1}^{p} \left(x^k\right)^T L x^k = \sum_{\langle i,j \rangle \in E} w_{ij} d_{ij}^2.$$

The proof of this lemma is direct.

Throughout the paper we will use the convention $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$ for the eigenvalues of L, and denote the corresponding real orthonormal eigenvectors by $v^1 = (1/\sqrt{n}) \cdot 1_n, v^2, \ldots, v^n$.

Let us define the *degrees matrix* as the $n \times n$ diagonal matrix D that satisfies $D_{ii} = \mathrm{deg(i)}$. Given a degrees matrix, D, and a Laplacian, L, then a vector u and a scalar μ are termed *generalized eigen-pairs* of (L, D) if $Lu = \mu Du$. Our convention is to denote the generalized eigenvectors of (L, D) by $u^1 = \alpha \cdot 1_n, u^2, \ldots, u^n$, with corresponding generalized eigenvalues $0 = \mu_1 < \mu_2 \leqslant \cdots \leqslant \mu_n$. (Thus, $Lu^k = \mu_i Du^k, \ k = 1, \ldots, n$.) To uniquely define $u^1, u^2, \ldots, u^n$, we require them to be D-normalized: so $\left(u^k\right)^T Du^k = 1, \ k = 1, \ldots n$. We term these generalized eigenvectors *the degree normalized eigenvectors*.

In general, for a symmetric (positive-semidefinite) matrix A and a positive-definite matrix B, it can be shown that all the generalized eigenvalues of (A, B) are real (non-negative), and that all the generalized eigenvectors are B-orthogonal. Consequently, $\left(u^k\right)^T Du^l = 0, \forall k \neq l$.

2.1 Mathematical preliminaries

Now, we develop some essential mathematical background that is needed for subsequent derivations. Different parts of this material can be found in standard linear algebra textbooks. The casual reader can make do with understanding the Theorems and Corollaries, and does not have to delve into the proofs. In the following, δ_{ij} is the Kronecker delta (defined as 1 for $i = j$ and as 0 otherwise), and A^k is the kth column of matrix A.

Theorem 1. *Given a symmetric matrix $A_{n \times n}$, denote by $v^1, \ldots, v^n$ its eigenvectors, with corresponding eigenvalues $\lambda_1 \leqslant \cdots \leqslant \lambda_n$. Then, $v^1, \ldots, v^p$ are an optimal solution of the constrained minimization problem:*

$$\begin{aligned} &\min_{x^1, \ldots, x^p} \quad \sum_{k=1}^{p} (x^k)^T A x^k \\ &\text{subject to: } (x^k)^T x^l = \delta_{kl}, \quad k, l = 1, \ldots p. \end{aligned} \tag{1}$$

Throughout this section, we assume $p < n$. Before providing the proof of Theorem 1, we prove the following lemma.

Lemma 2. *Let $v^1, \ldots, v^{p-1} \in \mathbb{R}^n$ be some vectors, and let X be an $n \times p$ matrix with orthogonal columns (i.e., $X^T X = I$), then there is an $n \times p$ matrix with orthogonal columns Y that satisfies:*

1. *For every $2 \leqslant k \leqslant p$: Y^k is orthogonal to $v^1, \ldots, v^{k-1}$.*

2. *For every* $n \times n$ *matrix* A, $\mathrm{trace}(X^T AX) = \mathrm{trace}(Y^T AY)$.

Proof. Let us denote the projection of v^1 into $\mathcal{R}ange(X)$ (i.e., $\mathrm{span}(X^1, \ldots, X^p)$) by $\tilde{v}^1$. If $\tilde{v}^1 = 0$ then we set $Y^1 = X^1$ and $\hat{Y}^2, \ldots, \hat{Y}^p = X^2, \ldots, X^p$. Certainly $\hat{Y}^2, \ldots, \hat{Y}^p$ are orthogonal to v^1. Otherwise (when $\tilde{v}^1 \neq 0$), we rotate $X^1, X^2 \ldots, X^p$ within $\mathcal{R}ange(X)$ obtaining $Y^1, \hat{Y}^2, \ldots, \hat{Y}^p$, such that $Y^1 = \tilde{v}^1/\|\tilde{v}^1\|$. Since rotations do not alter orthogonality relations, we still have that $\hat{Y}^2 \ldots, \hat{Y}^p$ are orthogonal to v^1. We continue recursively with the vectors $v^2, \ldots, v^{p-1}$ and the matrix $(\hat{Y}^2, \ldots, \hat{Y}^p)$. Note that the recursion performs rotations only within $\mathrm{span}(\hat{Y}^2, \ldots, \hat{Y}^p)$ so all resulting vectors are orthogonal to v^1 and to Y^1. At the end of the process we obtain p orthogonal vectors $Y^1, \ldots, Y^p$ that satisfy the first requirement from Y in the Lemma.

Since all rotations are performed within $\mathcal{R}ange(X)$, there is some $p \times p$ matrix R such that $Y = XR$. Moreover, since X and Y have orthogonal columns we obtain $I = Y^T Y = R^T X^T XR = R^T R$. Hence, R is an orthogonal matrix implying that also $RR^T = I$. We use the fact that the trace is cyclically-commutative to obtain:

$$\mathrm{trace}(Y^T AY) = \mathrm{trace}(R^T X^T AXR) = \mathrm{trace}(RR^T X^T AX) = \mathrm{trace}(X^T AX)\,.$$

$\square$

Now we can prove Theorem 1.

Proof. Let $x^1, \ldots, x^p$ be arranged column-wise in the $n \times p$ matrix X. Now, we can rewrite (1) in a simpler notation:

$$\begin{aligned} &\min_X \quad \mathrm{trace}(X^T AX) \\ &\text{subject to: } X^T X = I\,. \end{aligned} \tag{2}$$

Let $V_0 = (v_0^1, \ldots, v_0^p)$ be the minimizer of (2). Since the eigenvectors $v^1, \ldots, v^n$ form a basis of $\mathbb{R}^n$, we can decompose every v_0^k as a linear combination where $v_0^k = \sum_{l=1}^n \alpha_l^k v^l$. Lemma 2 allows us to assume without loss of generality that for every $2 \leqslant k \leqslant p$: v_0^k is orthogonal to the eigenvectors $v^1, \ldots, v^{k-1}$. We may therefore take $\alpha_l^k = 0$ for $l < k$, and write $v_0^k = \sum_{l=k}^n \alpha_l^k v^l$. Next, we use the constraint $(v_0^k)^T v_0^k = 1$ to obtain an equation for the coefficients α_l^k,

$$1 = \left(v_0^k\right)^T v_0^k = \left(\sum_{l=k}^n \alpha_l^k v^l\right)^T \left(\sum_{l=k}^n \alpha_l^k v^l\right) = \sum_{l=k}^n \left(\alpha_l^k\right)^2,$$

where the last equation stems from the orthonormality of $v^1, v^2, \ldots, v^n$.

Hence, $\sum_{l=k}^{n} \left(\alpha_l^k\right)^2 = 1$ (a generalization of Pythagoras' Law). Using this result, we can expand the quadratic form $\left(x^k\right)^T Ax^k$

$$\begin{aligned} (v_0^k)^T A v_0^k = \left(\sum_{l=k}^{n} \alpha_l^k v^l\right)^T A \left(\sum_{l=k}^{n} \alpha_l^k v^l\right) = \left(\sum_{l=k}^{n} \alpha_l^k v^l\right)^T \left(\sum_{l=k}^{n} \alpha_l^k A v^l\right) = \\ = \left(\sum_{l=k}^{n} \alpha_l^k v^l\right)^T \left(\sum_{l=k}^{n} \alpha_l^k \lambda_l v^l\right) = \sum_{l=k}^{n} \left(\alpha_l^k\right)^2 \lambda_l \geqslant \sum_{l=k}^{n} \left(\alpha_l^k\right)^2 \lambda_k = \lambda_k. \end{aligned} \tag{3}$$

Thus, the target function satisfies:

$$\operatorname{trace}(X^T AX) = \sum_{k=1}^{p} (x^k)^T Ax^k \geqslant \sum_{k=1}^{p} \lambda_k \,.$$

Since $\sum_{k=1}^{p} (v^k)^T Av^k = \sum_{k=1}^{p} \lambda_k$, we deduce that the low eigenvectors $v^1, \dots, v^p$ solve (1). $\square$

Now we state a more general form of Theorem 1.

Theorem 2. *Given a symmetric matrix $A_{n\times n}$ and a positive definite matrix B, denote by $v^1, \dots, v^n$ the generalized eigenvectors of (A, B), with corresponding eigenvalues $\lambda_1 \leqslant \dots \leqslant \lambda_n$. Then, $v^1, \dots, v^p$ are an optimal solution of the constrained minimization problem:*

$$\begin{aligned} &\min_{x^1,\dots,x^p} \quad \sum_{k=1}^{p} (x^k)^T Ax^k \\ &\text{subject to: } (x^k)^T Bx^l = \delta_{kl}, \quad k, l = 1, \dots p. \end{aligned} \tag{4}$$

Proof. Since B is positive definite it can be decomposed into: $B = C^T C$, where C is an $n \times n$ invertible matrix. Let us substitute in (4) $x^k = C^{-1} y^k$, so now we reformulate the problem as:

$$\begin{aligned} &\min_{y^1,\dots,y^p} \quad \sum_{k=1}^{p} (y^k)^T C^{-T} AC^{-1} y^k \\ &\text{subject to: } (y^k)^T y^l = \delta_{kl}, \quad k, l = 1, \dots p. \end{aligned} \tag{5}$$

Let $y_0^1, \dots, y_0^p$ be the minimizer of (5). By Theorem 1 these are simply the p lowest eigenvectors of $C^{-T}AC^{-1}$, obeying therefore$C^{-T}AC^{-1}y_0^k = \lambda_k y_0^k$. Using this equation and transforming back into $x_0^k = C^{-1}y_0^k$, we get $C^{-T}Ax_0^k = \lambda_k Cx_0^k$. This implies $Ax_0^k = \lambda_k Bx_0^k$, so the solvers of (4) are nothing but the lowest generalized eigenvectors of (A, B). $\square$

Frequently, the following version of Theorem 2 will be the most suitable:

Corollary 1. *Given a symmetric matrix $A_{n\times n}$ and a positive definite matrix $B_{n\times n}$, denote by $v^1, \ldots, v^n$ the generalized eigenvectors of (A, B), with corresponding eigenvalues $\lambda_1 \leqslant \cdots \leqslant \lambda_n$. Then, $v^1, \ldots, v^p$ are an optimal solution of the constrained minimization problem:*

$$\min_{x^1,\ldots,x^p} \quad \frac{\sum_{k=1}^{p}(x^k)^T A x^k}{\sum_{k=1}^{p}(x^k)^T B x^k} \tag{6}$$

$$\text{subject to: } (x^1)^T B x^1 = (x^2)^T B x^2 = \cdots = (x^p)^T B x^p \tag{7}$$

$$(x^k)^T B x^l = 0, \quad 1 \leqslant k \neq l \leqslant p. \tag{8}$$

Proof. Note that the value of (6) is invariant under scaling since for any constant $c \neq 0$:

$$\frac{\sum_{k=1}^{p}(x^k)^T A x^k}{\sum_{k=1}^{p}(x^k)^T B x^k} = \frac{\sum_{k=1}^{p}(cx^k)^T A (cx^k)}{\sum_{k=1}^{p}(cx^k)^T B (cx^k)}$$

Thus, we can always scale the optimal solution so that $(x^1)^T B x^1 = (x^2)^T B x^2 = \cdots = (x^p)^T B x^p = 1$. This reduces the problem to the form of (4). □

It is straightforward to prove the following corollary that deals with a case in which the solution must be B-orthogonal to the lowest generalized eigenvectors. In this case we just take the next low generalized eigenvectors.

Corollary 2. *Given a symmetric matrix $A_{n\times n}$ and a positive definite matrix $B_{n\times n}$, denote by $v^1, \ldots, v^n$ the generalized eigenvectors of (A, B), with corresponding eigenvalues $\lambda_1 \leqslant \cdots \leqslant \lambda_n$. Then, $v^{k+1}, \ldots, v^{k+p}$ are an optimal solution of the constrained minimization problem:*

$$\begin{aligned} &\min_{x^1,\ldots,x^p} \quad \frac{\sum_{i=1}^{p}(x^i)^T A x^i}{\sum_{i=1}^{p}(x^i)^T B x^i} \\ &\text{subject to: } (x^1)^T B x^1 = (x^2)^T B x^2 = \cdots = (x^p)^T B x^p \\ &\qquad (x^i)^T B x^j = 0, \quad 1 \leqslant i \neq j \leqslant p \\ &\qquad (x^i)^T B v^j = 0, \quad i = 1, \ldots, p,\ j = 1, \ldots, k\,. \end{aligned} \tag{9}$$

3 Spectral Graph Drawing

The earliest spectral graph-drawing algorithm was that of Hall [4]; it uses the low eigenvectors of the Laplacian. Henceforth, we will refer to this method as *the eigen-projection method*. Later, a similar idea was suggested in [11], where the results are shown to satisfy several desired aesthetic properties. A few other researchers utilize the top eigenvectors of the adjacency matrix instead of those of the Laplacian. For example, consider the work of [12], which uses the adjacency matrix eigenvectors to draw molecular graphs. Recently, eigenvectors of a modified Laplacian were used in [13] for the visualization of bibliographic networks.

In fact, for regular graphs of uniform degree deg, the eigenvectors of the Laplacian equal those of the adjacency matrix, but in a reversed order, because $L = deg \cdot I - A$, and adding the identity matrix does not change eigenvectors. However, for non-regular graphs, use of the Laplacian is based on a more solid theoretical basis, and in practice also gives nicer results than those obtained by the adjacency matrix. Hence, we will focus on visualization using eigenvectors of the Laplacian.

3.1 Energy-based Derivation of the Eigen-projection Method

One of the most popular approaches to graph-drawing is the *force-directed* strategy [2, 3] that defines an energy function (or a force model), whose minimization determines the optimal drawing. Consequently, we will introduce the eigen-projection method as the solution of the following constrained minimization problem:

$$\min_{x^1,\ldots,x^p} \quad E(x^1,\ldots,x^p) \stackrel{\text{def}}{=} \frac{\sum_{\langle i,j\rangle \in E} w_{ij} d_{ij}^2}{\sum_{i<j} d_{ij}^2} \tag{10}$$

$$\text{subject to: } \mathrm{Var}(x^1) = \mathrm{Var}(x^2) = \cdots = \mathrm{Var}(x^p) \tag{11}$$

$$\mathrm{Cov}(x^k, x^l) = 0, \quad 1 \leqslant k \neq l \leqslant p. \tag{12}$$

Here, $\mathrm{Var}(x)$ is the *variance* of x, defined as usual by $\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(x(i) - \bar{x}\right)^2$, where $\bar{x}$ is the mean of x. Also, $\mathrm{Cov}(x^k, x^l)$ is the covariance of x^k and x^l defined as $\frac{1}{n}\sum_{i=1}^{n}\left(x^k(i) - \bar{x}^k\right)\left(x^l(i) - \bar{x}^l\right)$. Recall that $d_{ij}^2 = \sum_{k=1}^{p}\left(x^k(i) - x^k(j)\right)^2$.

The energy to be minimized, $E(x^1,\ldots,x^p)$, strives to make edge lengths short (to minimize the numerator) while scattering the nodes in the drawing area preventing an overcrowding of the nodes (to maximize the denominator). This way, we adopt a common strategy to graph drawing stating that adjacent nodes should be drawn closely, while, generally, nodes should not be drawn too close to each other; see, e.g., [14, 15]. Since the sum is weighted by edge-weights, “heavy” edges have a stronger impact and hence will be typically shorter. The first constraint (11) forces the nodes to be equally scattered along each of the axes. In this sense, the drawing has a perfectly balanced aspect ratio. The second constraint (12) ensures that there is no correlation between the axes so that each additional dimension will provide us with as much new information as possible[1].

The energy and the constraints are invariant under translation. We eliminate this degree of freedom by requiring that for each $1 \leqslant k \leqslant p$ the mean of x^k is 0, i.e.

[1] The strategy to require no correlation between the axes is used in other visualization techniques like Principal Components Analysis [16] and Classical Multidimensional Scaling [16].

$\sum_{i=1}^{n} x^k(i) = \left(x^k\right)^T \cdot 1_n = 0$. This is very convenient since now the no-correlation constraint, $\mathrm{Cov}(x^k, x^l) = 0$, is equivalent to requiring the vectors to be pairwise orthogonal $\left(x^k\right)^T x^l = 0$. Also, now $\mathrm{Var}(x^k) = \frac{1}{n}\left(x^k\right)^T x^k$ so the uniform variance constraint can be written in a simple form: $\left(x^1\right)^T x^1 = \left(x^2\right)^T x^2 = \cdots = \left(x^p\right)^T x^p$.

Using Lemma 1, we can write $\sum_{\langle i,j\rangle \in E} w_{ij} d_{ij}^2$ in a matrix form: $\sum_{k=1}^{p} \left(x^k\right)^T L x^k$. Now, the desired layout can be described as the solution of the following equivalent minimization problem:

$$\min_{x^1,\ldots,x^p} \frac{\sum_{k=1}^{p} \left(x^k\right)^T L x^k}{\sum_{i<j} d_{ij}^2} \tag{13}$$
$$\text{subject to: } \left(x^k\right)^T \left(x^l\right) = \delta_{kl}, \quad k,l = 1,\ldots,p$$
$$\left(x^k\right)^T \cdot 1_n = 0, \quad k = 1,\ldots,p.$$

We can simplify the denominator $\sum_{i<j} d_{ij}^2$ using the following lemma:

Lemma 3. *Let $x \in \mathbb{R}^n$ such that $x^T 1_n = 0$, then:*

$$\sum_{i<j} (x(i) - x(j))^2 = n \cdot \sum_{i=1}^{n} x(i)^2 \ (= n \cdot x^T x) \,.$$

Proof.

$$\sum_{i<j} (x(i) - x(j))^2 = \frac{1}{2} \sum_{i,j=1}^{n} (x(i) - x(j))^2 = \frac{1}{2} \left(2n \sum_{i=1}^{n} x(i)^2 - 2 \sum_{i,j=1}^{n} x(i)x(j) \right) =$$
$$= n \cdot \sum_{i=1}^{n} x(i)^2 - \sum_{i=1}^{n} x(i) \sum_{j=1}^{n} x(j) = n \cdot \sum_{i=1}^{n} x(i)^2$$

The last step stems from the fact that x is centered, so that $\sum_{j=1}^{n} x(j) = 0$. □

As a consequence we get

$$\sum_{i<j} d_{ij}^2 = \sum_{i<j} \sum_{k=1}^{p} \left(x^k(i) - x^k(j)\right)^2 = \sum_{k=1}^{p} \sum_{i<j} \left(x^k(i) - x^k(j)\right)^2 = \sum_{k=1}^{p} n \cdot \left(x^k\right)^T x^k$$

Therefore, we rewrite again the minimization problem in an equivalent form:

$$\min_{x^1,\ldots,x^p} \frac{\sum_{k=1}^{p} \left(x^k\right)^T L x^k}{\sum_{k=1}^{p} \left(x^k\right)^T x^k} \tag{14}$$
$$\text{subject to: } \left(x^k\right)^T x^l = \delta_{kl}, \quad k,l = 1,\ldots,p$$
$$\left(x^k\right)^T \cdot 1_n = 0, \quad k = 1,\ldots,p.$$

Let us substitute $A = L, B = I$ in Corollary 2. Using the fact that the lowest eigenvector of L is 1_n we obtain that the optimal layout is given by the lowest positive Laplacian eigenvectors $v^2, \ldots, v^{p+1}$. The resulting value of the energy is $\sum_{k=2}^{p+1} \lambda_k$, the sum of the corresponding eigenvalues.

Note that an interesting property of the eigen-projection is that the first $p-1$ axes of the optimal p-dimensional layout are nothing but the optimal $(p-1)$-dimensional layout.

4 Drawing using Degree-Normalized Eigenvectors

In this section we introduce a new, related spectral graph drawing method that associates the coordinates with some generalized eigenvectors of the Laplacian.

Suppose that we weight nodes by their degrees, so the mass of node i is its degree — $\mathrm{deg(i)}$. Now if we take the original constrained minimization problem (14) and weight sums according to node masses, we get the following degree-weighted constrained minimization problem (where D is the degrees matrix):

$$\min_{x^1,\ldots,x^p} \frac{\sum_{k=1}^{p} \left(x^k\right)^T L x^k}{\sum_{k=1}^{p} \left(x^k\right)^T D x^k} \tag{15}$$
$$\text{subject to: } \left(x^k\right)^T D \left(x^l\right) = \delta_{ij}, \quad k,l = 1,\ldots,p$$
$$\left(x^k\right)^T D 1_n = 0, \quad k = 1,\ldots,p.$$

Substitute $A = L, B = D$ in Corollary 2 to obtain the optimal solution $u^2, \ldots, u^{p+1}$, the generalized eigenvectors of (L, D). We will show that in several aspects using these degree-normalized eigenvectors is more natural than using the eigenvectors of the Laplacian. In fact Shi and Malik [7] have already shown that the degree-normalized eigenvectors are more suitable for the problem of image segmentation. For the visualization task, the motivation and explanation are very different.

In problem (15) the denominator moderates the behavior of the numerator: The numerator strives to place those nodes with high degrees at the center of the drawing, so that they are in proximity to the other nodes. On the other hand, the denominator also emphasizes those nodes with high degrees, but in the reversed way: it strives to enlarge their scatter. The combination of these two opposing goals, helps in making the drawing more balanced, preventing a situation in which nodes with lower degrees are overly separated from the rest nodes.

Another observation is that degree-normalized eigenvectors unify the two common spectral techniques: the approach that uses the Laplacian and the approach that uses the adjacency matrix.

Claim. The generalized eigenvectors of (L, D) are also the generalized eigenvectors of (A, D), with a reversed order.

Proof. Utilize the fact that $L = D - A$. Take u, a generalized eigenvector of (L, D). The vector u satisfies $(D - A)u = \mu Du$, or equivalently, by changing sides, $Au = Du - \mu Du$. This implies that

$$Au = (1 - \mu)Du$$

and the claim follows. The proof in the other direction is performed similarly.
Thus, A and L have the same D-normalized eigenvectors, although the order of eigenvalues is reversed. $\square$

In this way, when drawing with degree normalized eigenvectors, we can take either the low generalized eigenvectors of the Laplacian, or the top generalized eigenvectors of the adjacency matrix, without affecting the result. (Remark: In this paper when referring to “top” or ”low” eigenvectors, we often neglect the topmost (or lowest) degenerate eigenvector $\alpha \cdot 1_n$.)

The degree-normalized eigenvectors are also the (non-generalized) eigenvectors of the matrix $D^{-1}A$. This can be obtained by left-multiplying the generalized eigen-equation $Ax = \mu Dx$ by D^{-1}, obtaining the eigen-equation

$$D^{-1}Ax = \mu x. \tag{16}$$

Note that $D^{-1}A$ is known as the *transition matrix* of a random walk on the graph G. Hence, the degree-normalized eigen-projection uses the top eigenvectors of the transition matrix to draw the graph.

Regarding drawing quality, for most unweighted graphs with which we experimented (that are probably close to being regular), we have observed not much difference between drawing using eigenvectors and drawing using degree-normalized eigenvectors. However, when there are marked deviations in node degrees (as is often the case when working with weighted graphs), the results are quite different. This can be directly seen by posing the problem as in (15). Here, we provide an alternative explanation based on (16). Consider the two edges e_1 and e_2. Edge e_1 is of weight 1, connecting two nodes, each of which is of degree 10. Edge e_2 is of weight 10, connecting two nodes, each of which is of degree 100. In the Laplacian matrix, the entries corresponding to e_2 are 10 times larger than those corresponding to e_1. Hence we expect the drawing obtained by the eigenvectors of the Laplacian, to make the edge e_2 much shorter than e_1 (here, we do not consider the effect of other nodes that may change the lengths of both edges). However, for the transition matrix in (16), the entries corresponding to these two edges are the same, hence we treat them similarly and expect to get the same length for both edges. This reflects the fact that the *relative* importance of these two edges is the same, i.e. $\frac{1}{10}$.

In many kinds of graphs numerous scales are embedded, which indicates the existence of dense clusters and sparse clusters. In a traditional eigen-projection drawing, dense clusters are drawn extremely densely, while the whole area of the drawing is used to represent sparse clusters or outliers. This might be the best way to minimize the weighted sum of square edge lengths, while scattering the nodes as demanded. A better drawing would allocate each cluster an adequate area. Frequently, this is the case with the degree normalized eigenvectors that adjust the edge weights in order to reflect their relative importance in the related local scale.

For example, consider Fig. 2, where we visualize 300 odors as measured by an electronic nose. Computation of the similarities between the odors is given in [17]. The odors are known to be classified into 30 groups, which determine the grayscale of each odor in the figure. Figure 2(a) shows the visualization of the odors by the eigenvectors of the Laplacian. As can be seen, each of the axes shows one outlying odor, and places all the other odors about at the same location. However, the odors are nicely visualized using the degree normalized eigenvectors, as shown in Fig. 2(b).

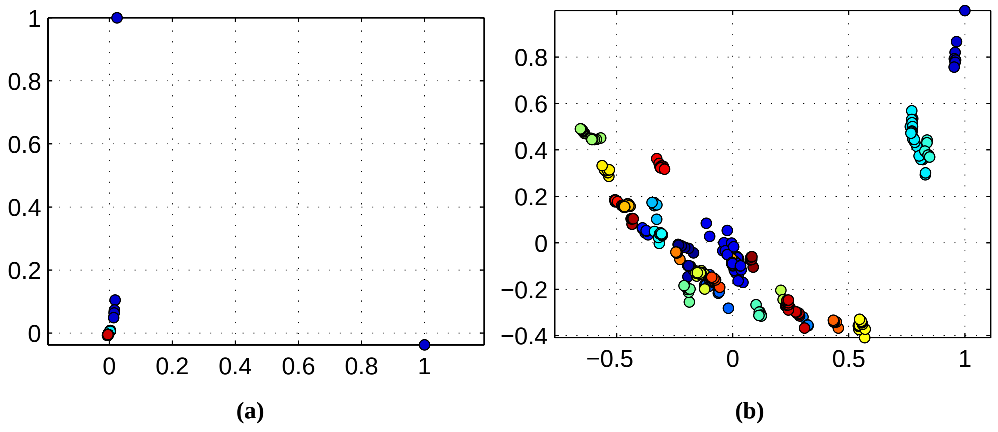

Fig. 2. Visualization of 300 odor patterns as measured by an electronic nose. **(a)** A drawing using the eigenvectors of the Laplacian. **(b)** A drawing using the degree-normalized eigenvectors.

5 A Direct Characterization of Spectral Layouts

So far, we have derived spectral methods as solutions of optimization problems. In this section we characterize the eigenvectors themselves, in a rather direct manner, to clarify the aesthetic properties of the spectral layout. The new derivation will show a very simple relation between layout aesthetics and the Laplacian (generalized) eigenvectors.

Here, the degree-normalized eigenvectors will appear as a more natural way for spectral graph drawing.

The quadratic form associated with the Laplacian $x^T Lx = \sum_{\langle i,j\rangle \in E} w_{ij}(x(i) - x(j))^2$, is tightly related to the aesthetic criterion that calls for placing each node at the weighted centroid of its neighbors. To conceive this note that for each node i differentiating $x^T Lx$ with respect to $x(i)$ gives

$$\frac{\partial x^T Lx}{\partial x(i)} = 2 \sum_{j \in N(i)} w_{ij}(x(i) - x(j)).$$

Equating this to zero and isolating $x(i)$ we get

$$x(i) = \frac{\sum_{j \in N(i)} w_{ij} x(j)}{\deg(\mathrm{i})} \ .$$

Hence, when allowing only node i to move, the location of i that minimizes $x^T Lx$ is the weighted centroid of i's neighbors.

When the graph is connected, placing each node at the weighted centroid of its neighbors can be strictly achieved only by the degenerate solution that puts all nodes at the same location. Hence, to incorporate this aesthetic criterion into a graph drawing algorithm, it should be modified appropriately.

Presumably the earliest graph drawing algorithm, formulated by Tutte [18], is based on placing each node on the weighted centroid of its neighbors. To avoid the degenerate solution, Tutte arbitrarily chose a certain number of nodes to be anchors, i.e. he fixed their coordinates in advance. Those nodes are typically drawn on the boundary. This, of course, prevents the collapse; however it raises new problems, such as which nodes should be the anchors, how to determine their coordinates, and why after all such an anchoring mechanism should generate nice drawings. An advantage of Tutte's method is that in certain cases, it can guarantee achieving a planar drawing (i.e., without edge crossings).

Tutte treats in different ways the anchored nodes and the remaining nodes. Whereas the remaining nodes are located exactly at the centroid of their neighbors, nothing can be said about anchored nodes. In fact, in several experiments we have seen that the anchored nodes are located quite badly.

Alternatively, we do not use different strategies for dealing with two kinds of nodes, but rather, we treat all the nodes similarly. The idea is to gradually increase the deviations from centroids of neighbors as we move away from the origin (that is the center of the drawing). This reflects the fact that central nodes can be placed exactly at their neighbors' centroid, whereas boundary nodes must be shifted outwards.

More specifically, node i, which is located in place $x(i)$, is shifted from the center toward the boundary by the amount of $\mu \cdot |x(i)|$, for some $\mu > 0$. Formally, we request the layout x to satisfy, for every $1 \leqslant i \leqslant n$

$$x(i) - \frac{\sum_{j \in N(i)} w_{ij} x(j)}{\mathrm{deg(i)}} = \mu \cdot x(i)\,.$$

Note that the deviation from the centroid is always toward the boundary, i.e. toward $+\infty$ for positive $x(i)$ and toward $-\infty$ for negative $x(i)$. In this way we prevent a collapse at the origin.

We can represent all these n requests compactly in a matrix form, by writing

$$D^{-1}Lx = \mu x\,.$$

Left-multiplying both sides by D, we obtain the familiar generalized eigen-equation

$$Lx = \mu D x\,.$$

We conclude with the following important property of degree-normalized eigenvectors:

Proposition 1. *Let u be a generalized eigenvector of (L, D), with associated eigenvalue μ. Then, for each i, the exact deviation from the centroid of neighbors is*

$$u(i) - \frac{\sum_{j \in N(i)} w_{ij} u(j)}{\mathrm{deg(i)}} = \mu \cdot u(i)\,.$$

Note that the eigenvalue μ is a scale-independent measure of the amount of deviation from the centroids. This provides us with a fresh new interpretation of the eigenvalues that is very different from the one given in Subsection 3.1, where the eigenvalues were shown as the amount of energy in the drawing.

Thus, we deduce that the second smallest degree-normalized eigenvector produces the non-degenerate drawing with the smallest deviations from centroids, and that the third smallest degree-normalized eigenvector is the next best one and so on.

Similarly, we can obtain a related result for eigenvectors of the Laplacian:

Proposition 2. *Let v be an eigenvector of L, with associated eigenvalue λ. Then, for each i, the exact deviation from the centroid of neighbors is*

$$v(i) - \frac{\sum_{j \in N(i)} w_{ij} v(j)}{\mathrm{deg(i)}} = \lambda \cdot \mathrm{deg(i)}^{-1} \cdot v(i)\,.$$

Hence for eigenvectors of the Laplacian, the deviation between a node and the centroid of its neighbors gets larger as the node's degree decreases.

6 An Optimization Process

An attractive feature of the degree-normalized eigenvectors is that they can be computed by an intuitive algorithm, which is directly related to their aesthetic properties. This is unlike the (non generalized) eigenvectors, which are computed using methods that are difficult to interpret in aesthetic terms. We begin by deriving an algorithm for producing a 1-D layout $x \in \mathbb{R}^n$, and then we show how to compute more axes.

Our algorithm is based on iteratively placing each node at the weighted centroid of its neighbors (simultaneously for all nodes). The aesthetic reasoning is clear; as explained in Section 5 this principle unifies the Tutte layout [18] and the eigen-projection.

A rather impressive fact is that when initialized with a vector D-orthogonal to 1_n, such an iterative placement algorithm converges *in the direction* of a non-degenerate degree-normalized eigenvector of L. More precisely, this algorithm converges either in the direction of u^2 or that of u^n. We can prove this surprising fact by observing that the action of putting each node at the weighted centroid of its neighbors is equivalent to multiplication by the transition matrix — $D^{-1}A$. Thus, the process we have described can be expressed in a compact form as the sequence

$$\begin{cases} x_0 &= \text{random vector, s.t. } \ x_0^T D 1_n = 0 \\ x_{i+1} &= D^{-1} A x_i \ . \end{cases}$$

This process is known as the Power-Iteration [19]. In general, it computes the "dominant" eigenvector of $D^{-1}A$, which is the one associated with the largest-in-magnitude eigenvalue. In our case, all the eigenvectors are D-orthogonal to the "dominant" eigenvector — 1_n, and also the initial vector, x_0, is D-orthogonal to 1_n. Thus, the series converges in the direction of the next dominant eigenvector, which is either u^2, which has the largest positive eigenvalue, or u^n, which possibly has the largest negative eigenvalue. (We assume that x_0 is not D-orthogonal to u^2 or to u^n, which is nearly always true for a randomly chosen x_0.)

In practice, we want to ensure convergence to u^2 (avoiding convergence to u^n). We use the fact that all the eigenvalues of the transition matrix are in the range $[-1, 1]$. This can be proved directly using the Gershgorin bound on eigenvalues [19], since in $D^{-1}A$ all entries on the diagonal are 0, and the sum of each row is 1. Now it is possible to shift the eigenvalues by adding the value 1 to each of them, so that they are all positive, thus preventing convergence to an eigenvector with a large negative eigenvalue. This is done by working on the matrix $I + D^{-1}A$ instead of the matrix $D^{-1}A$. In this way the eigenvalues are in the range $[0, 2]$, while eigenvectors are not changed. In fact, it would be more intuitive to scale the eigenvalues to the range $[0, 1]$, so we will actually work with the matrix $\frac{1}{2}(I + D^{-1}A)$. If we use our initial "intuitive" notions, this means a more careful process. In each iteration, we put each node at the average between its old place and the centroid of its neighbors. Thus, each node absorbs its new location not only from its neighbors, but also from its current location.

The full algorithm for computing a multidimensional drawing is given in Fig. 3. To compute a degree-normalized eigenvector u^k, we will use the principles of the power-iteration and the D-orthogonality of the eigenvectors. Briefly, we pick some random x, such that x is D-orthogonal to $u^1, \ldots, u^{k-1}$, i.e. $x^T D u^1 = 0, \ldots, x^T D u^{k-1} = 0$. Then, if $x^T D u^k \neq 0$, it can be proved that the series $\frac{1}{2}(I + D^{-1}A)x, (\frac{1}{2}(I + D^{-1}A))^2 x, (\frac{1}{2}(I + D^{-1}A))^3 x, \ldots$ converges in the direction of u^k. Note that in theory, all the vectors in this series are D-orthogonal to $u^1, \ldots, u^{k-1}$. However, to improve numerical stability, our implementation imposes the D-orthogonality to previous eigenvectors in each iteration. The power iteration algorithm produces vectors of diminishing (or exploding) norms. Since we are only interested in convergence *in direction*, it is customary to re-scale the vectors after each iteration. Here, we will re-scale by normalizing the vectors to be of length 1.

The convergence rate of this algorithm when computing u^k is dependent on the ratio μ_k/μ_{k+1}. Running time is significantly improved when replacing the random initialization of the vectors with some well-designed initialization. A way to obtain such a smart initialization is described in Section 7. An alternative approach is to embed this algorithm in a multi-scale construction. This is done by approximating the original graph using a *coarse graph* about half the size, and then computing recursively the eigenvectors of the coarse graph. These eigenvectors are used to obtain an initial prediction of the eigenvectors of the original graph. Our experience with such a multi-scale strategy shows an extremely rapid convergence; further details are given in [20].

7 Spectral Drawings within a Subspace

Most graph drawing methods suffer from lengthy computation times when applied to really large graphs. Hence, a particularly challenging problem is drawing large graphs containing 10^3–10^6 nodes, which has gained much interest recently because of the rapid growth of data collections. (Recent work on accelrating force-directed layout algorithms includes [21–24].) In general, when using standard eigen-solvers the eigen-projection method is very fast compared to almost all other graph-drawing algorithms. However, calculating the first few eigenvectors of L is a difficult task that presents a real problem for standard algorithms when n becomes around 10^5. Consequently, we describe an approach that significantly reduces running times, enabling the drawing of huge graphs in a reasonable time.

```
Function SpectralDrawing ($G$ – the input graph, $p$ – dimension)
% This function computes $u^2, \ldots, u^p$, the top (non-degenerate) eigenvectors of $D^{-1}A$.
  const $\epsilon \leftarrow 10^{-8}$   % tolerance
  for $k = 2$ to $p$ do
    $\hat{u}^k \leftarrow$ random   % random initialization
    $\hat{u}^k \leftarrow \frac{\hat{u}^k}{\|\hat{u}^k\|}$
    do
      $u^k \leftarrow \hat{u}^k$
      % $D$-Orthogonalize against previous eigenvectors:
      for $l = 1$ to $k - 1$ do
        $u^k \leftarrow u^k - \frac{(u^k)^T D u^l}{(u^l)^T D u^l} u^l$
      end for
      % multiply with $\frac{1}{2}(I + D^{-1}A)$:
      for $i = 1$ to $n$ do
        $\hat{u}^k(i) \leftarrow \frac{1}{2} \cdot \left( u^k(i) + \frac{\sum_{j \in N(i)} w_{ij} u^k(j)}{\mathrm{deg(i)}} \right)$
      end for
      $\hat{u}^k \leftarrow \frac{\hat{u}^k}{\|\hat{u}^k\|}$   % normalization
    while $\hat{u}^k \cdot u^k < 1 - \epsilon$   % halt when direction change is negligible
    $u^k \leftarrow \hat{u}^k$
  end for
  return $u^2, \ldots, u^p$
```

Fig. 3. The algorithm for computing degree-normalized eigenvectors

Whereas in eigen-projection we chose the drawing axes as arbitrary vectors in $\mathbb{R}^n$, now we would like to constrain the drawing axes to lie within some subspace (vector space) $\mathbb{S} \subseteq \mathbb{R}^n$. Thus, we require $x^1, \ldots, x^p \in \mathbb{S}$. We define a subspace $\mathbb{S}$ by some $n \times m$ matrix $\mathcal{X}$ whose columns span $\mathbb{S}$ (so $\mathbb{S} = \mathcal{R}ange(\mathcal{X})$). Consequently, we can always denote the axes of the drawing by the vectors $y^1, \ldots, y^p \in \mathbb{R}^m$ so that $x^1 = \mathcal{X}y^1, \ldots, x^p = \mathcal{X}y^p$.

Clearly, constraining the drawing to lie within a subspace might result in arbitrarily bad layouts. However, as we are going to show, we can construct subspaces that contain reasonably nice layouts, so we are not loosing much by working only within such subspaces.

7.1 The high dimensional embedding subspace

An appropriate subspace is based on the high-dimensional embedding (HDE) that we have already used in [25]. This HDE comprises m axes $\mathcal{X}^1, \ldots, \mathcal{X}^m \in \mathbb{R}^n$. In order to construct it, we choose m *pivot* nodes $p_1, p_2, \ldots, p_m$ that are uniformly distributed over the graph and link each of the m axes with a unique node. The axis $\mathcal{X}^i$, which is associated with pivot node p_i, represents the graph from the "viewpoint" of p_i. This is done by assigning the j-th component of $\mathcal{X}^i$ to the graph-theoretical distance between nodes p_i and j (i.e., the length of the shortest path connecting the two nodes). Henceforth, we denote this graph-theoretical distance by $\mathcal{D}_{p_i j}$, so in symbols $\mathcal{X}^i(j) = \mathcal{D}_{p_i j}$.

The resulting algorithm for constructing the high-dimensional embedding is given in Fig. 4. The graph-theoretical distances are computed using breadth-first-search (BFS) [26]. The pivots $p_1, p_2, \ldots, p_m$ are chosen by a heuristic for the k-centers problem, as follows. The first member, p_1, is chosen at random. For $j = 2, \ldots, m$, node p_j is a node that maximizes the shortest distance from $\{p_1, p_2, \ldots, p_{j-1}\}$. This method is mentioned in [27] as a 2-approximation[2] to the *k-center* problem, where we want to choose k nodes of V, such that the longest distance from V to these k centers is minimized. The time complexity of this algorithm is $O(m \cdot |E|)$, since we perform BFS in each of the m iterations.

We have previously shown that the HDE is a kind of m-dimensional layout of the graph and that PCA projections of the high-dimensional embedding typically yield nice layouts; see [25]. Projections are just a type of linear combinations, so restricting the layout to lie inside $\mathrm{span}(\mathcal{X}^1, \ldots, \mathcal{X}^m)$ is plausible. In practice, we have found that choosing $m \sim 50$ serves very well for producing a nice layout.

It is important that $\mathcal{X}^1, \ldots, \mathcal{X}^m$ are orthogonal, or at least linearly independent (so they form a valid basis). Of course, this characteristic can be obtained without altering $\mathrm{span}(\mathcal{X}^1, \ldots, \mathcal{X}^m)$. Moreover, it will be convenient for us that all vectors are orthogonal to 1_n, in order to eliminate the redundant translation degree-of-freedom. We achieve all these requirements by a variant of the Gram-Schmidt orthonormalization procedure shown in Fig. 5. Note that vectors that are (almost) linearly dependent will get the value 0 and therefore should be removed. The entire orthonormalization process takes $O(m^2 n)$ time. After completing this process we take all the (non-zero) vectors, and arrange them, column-wise, in the matrix $\mathcal{X}$.

[2] A δ-approximation algorithm delivers an approximate solution guaranteed to be within a constant factor δ of the optimal solution.

```
Function HDE ($G(V = \{1, \ldots, n\}, E), m$)
% This function finds an $m$-dimensional high-dimensional embedding of $G$

  Choose node $p_1$ randomly from $V$
  $d[1, \ldots, n] \leftarrow \infty$
  for $i = 1$ to $m$ do
      % Compute the $i - th$ coordinate using BFS
      $\mathcal{D}_{p_i *} \leftarrow \mathrm{BFS}(G(V, E), p_i)$
      for every $j \in V$
          $\mathcal{X}^i(j) \leftarrow \mathcal{D}_{p_i j}$
          $d[j] \leftarrow \min\{d[j], \mathcal{X}^i(j)\}$
      end for
      % Choose next pivot
      $p_{i+1} \leftarrow \arg\max_{\{j \in V\}}\{d[j]\}$
  end for
  return $\mathcal{X}^1, \ldots, \mathcal{X}^m$
```

Fig. 4. Constructing an m dimensional HDE

```
Function Orthonormalize ($\{u^1, \ldots, u^m\}$)
% This function orthonormalizes a set of vectors.
% Also, it orthogonalizes the vectors against $1_n$

  const $u^0 \leftarrow \frac{1_n}{\|1_n\|}, \ \epsilon \leftarrow 0.001$
  for $i = 1$ to $m$ do
    for $j = 0$ to $i - 1$ do
      $u^i \leftarrow u^i - \left(\left(u^i\right)^T u^j\right) u^j$
    end for
    if $\|u^i\| < \epsilon$ then
      % a linearly dependent vector
      $u^i \leftarrow 0$
    else
      $u^i \leftarrow \frac{u^i}{\|u^i\|}$
    end if
  end for
```

Fig. 5. Gram-Schmidt orthonormalization

7.2 Eigen-projection in a subspace

As shown in Subsection 3.1, eigen-projection defines the axes as the minimizers of (14):

$$\begin{aligned} &\min_{x^1,\ldots,x^p \in \mathbb{R}^n} \frac{\sum_{k=1}^{p} \left(x^k\right)^T L x^k}{\sum_{k=1}^{p} \left(x^k\right)^T x^k} \\ &\text{subject to: } \left(x^k\right)^T x^l = \delta_{kl}, \quad k,l = 1,\ldots,p \\ &\qquad\qquad\quad \left(x^k\right)^T \cdot 1_n = 0, \quad k = 1,\ldots,p. \end{aligned}$$

In our case, we want to optimize $x^1,\ldots,x^p$ within the subspace $\mathcal{R}ange(\mathcal{X})$, and this can be achieved by replacing them with $\mathcal{X}y^1,\ldots,\mathcal{X}y^p$. Hence, (14) becomes

$$\begin{aligned} &\min_{y^1,\ldots,y^p \in \mathbb{R}^m} \frac{\sum_{k=1}^{p} \left(y^k\right)^T \mathcal{X}^T L \mathcal{X} y^k}{\sum_{k=1}^{p} \left(y^k\right)^T \mathcal{X}^T \mathcal{X} y^k} \\ &\text{subject to: } \left(y^k\right)^T \mathcal{X}^T \mathcal{X} y^l = \delta_{kl}, \quad k,l = 1,\ldots,p. \end{aligned}$$

Note, that here we do not impose orthogonality of the axes to 1_n, as it is already achieved by the fact that $1_n \notin \mathcal{R}ange(X)$. We can further simplify the problem by utilizing the fact that $\mathcal{X}^T\mathcal{X} = I$, obtaining the equivalent problem:

$$\begin{aligned} &\min_{y^1,\ldots,y^p \in \mathbb{R}^m} \frac{\sum_{k=1}^{p} \left(y^k\right)^T \mathcal{X}^T L \mathcal{X} y^k}{\sum_{k=1}^{p} \left(y^k\right)^T y^k} \\ &\text{subject to: } \left(y^k\right)^T y^l = \delta_{kl}, \quad k,l = 1,\ldots,p. \end{aligned} \tag{17}$$

By Corollary 1 the minimizers of (17), are the lowest eigenvectors of $\mathcal{X}^T L \mathcal{X}$.

To summarize, let us be restricted to a subspace spanned by the columns of the orthogonal matrix $\mathcal{X}$. The drawing can be obtained by first computing the p lowest eigenvectors of $\mathcal{X}^T L \mathcal{X}$, denoted by $y^1,\ldots,y^p$, and then taking the coordinates to be $\mathcal{X}y^1,\ldots,\mathcal{X}y^p$.

Computation of the product $\mathcal{X}^T L \mathcal{X}$ is done in two steps: first, we compute $L\mathcal{X}$ in time $O(m|E|)$ utilizing the sparsity of L, and then we compute $\mathcal{X}^T(L\mathcal{X})$ in time $O(m^2 n)$. Note that $\mathcal{X}^T L \mathcal{X}$ is an $m \times m$ matrix, where typically $m \sim 50$, so the eigenvectors' calculation takes negligible time (about a millisecond). This is of course a significant benefit of optimizing within a subspace. We recommend that a very accurate calculation be performed; this improves the layout quality with an insignificant affect on running time. In practice, we invert the order of the eigenvectors by using the matrix $B = \mu \cdot I - \mathcal{X}^T L \mathcal{X}$, and compute the highest eigenvectors of B using the power-iteration [19]. The scalar μ is the highest eigenvalue of $\mathcal{X}^T L \mathcal{X}$ that can be computed directly by the power-iteration. Alternatively, one can set μ to the Gershgorin bound [19], which is a theoretical upper bound for (the absolute value of) the largest eigenvalue of a matrix. Specifically, for our matrix this bound is given by:

$$\max_i \left(\left(\mathcal{X}^T L \mathcal{X}\right)_{ii} + \sum_{j \neq i} | \left(\mathcal{X}^T L \mathcal{X}\right)_{ij} | \right) .$$

Remarks

1. In a case that we want to use the notion of "degree-normalized" generalized eigenvectors (i.e., to optimize (15) within the HDE subspace), all we have to do is to D-orthonormalize $\mathcal{X}$'s columns, instead of orthonormalizing them (this is a slight change to the Gram-Schmidt process).
2. When the nodes represent multi-dimensional points, we can take the original coordinates as the subspace within which we optimize. One of the benefits is that the resulting layout is a linear combination of the original multi-dimensional data; for more details see [28].

7.3 Results of subspace-constrained optimization

We have validated the performance of optimization within the HDE subspace by experimenting with many graphs. The results are very encouraging – we obtained quality layouts in a very rapid time. For example we provide in Fig. 6 the layouts of the four graphs that were previously shown in Fig. 1.

Here, we would like to mention some general comments about the nature of drawings produced by the eigen-projection. On the one hand, we are assured to be in a global minimum of the energy, thus we might expect the global layout of the drawing to faithfully represent the structure of the graph. On the other hand, there is nothing that prevents nodes from being very close. Hence, the drawing might show dense local arrangement. These general claims are nicely demonstrated in the examples drawn in Fig. 1. Comparing the layouts in Fig. 1 with those in Fig. 6, it is clear that when optimizing within the HDE subspace the layouts are somewhat less globally-optimal and the symmetries are not perfectly shown. However, an aesthetical advantage of working within the HDE subspace is that the undesirable locally dense regions are less common, and the nodes are more evenly distributed over the layout. This can be observed by comparing the boundaries of the layouts in the two figures or by observing the folded part of the grid in layout **(d)**.

The ability of the HDE-constrained optimization to show delicate details that are often hidden in eigen-projection layouts stems from the fact that all coordinates in the

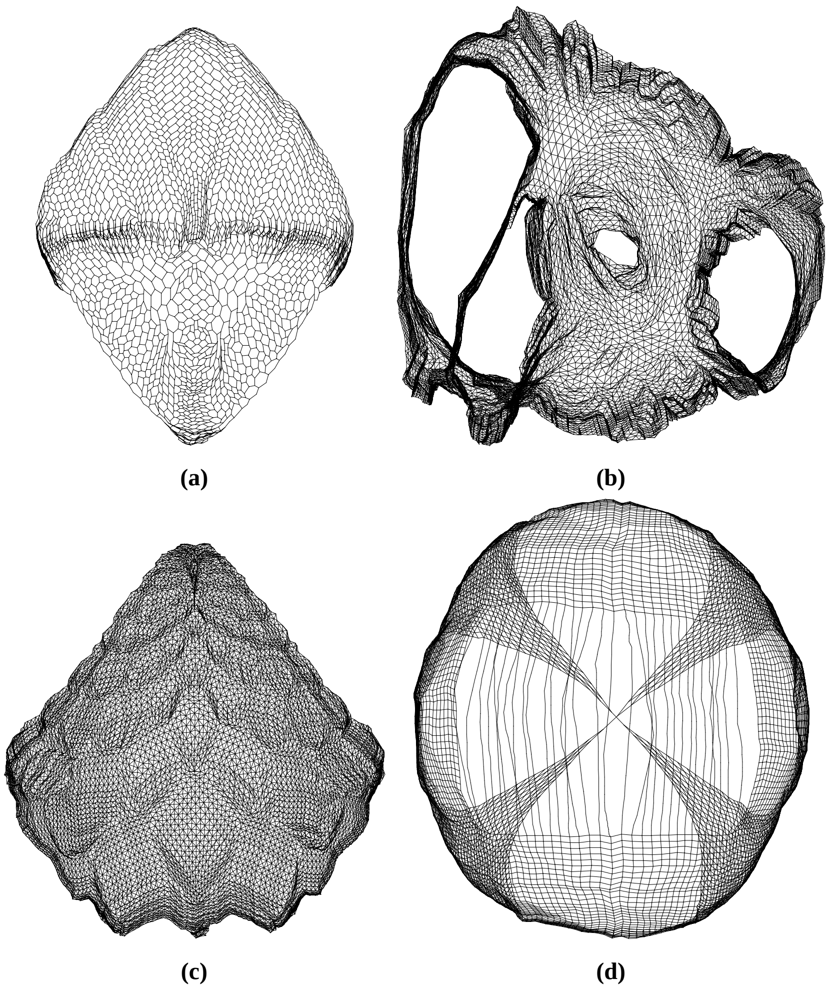

Fig. 6. Drawings obtained by HDE-constrained eigen-projection. **(a)** The 4970 graph. $|V| = 4,970$, $|E| = 7,400$. **(b)** The 4elt graph [10]. $|V| = 15,606$, $|E| = 45,878$. **(c)** The Crack graph [10]. $|V| = 10,240$, $|E| = 30,380$. **(d)** A 100×100 folded grid with central horizontal edge removed. $|V| = 10,000$, $|E| = 18,713$.

HDE subspace are integral (up to translation and orthogonalization). We further demonstrate this ability in Fig. 7, where we compared side-by-side the results of constrained and unconstrained eigen-projection. Clearly, the new method agrees with the eigen-projection regarding the global structure of the layout, but provides additional finer details.

An interesting issue is whether eigen-projection always produces cross-free layouts of planar graphs. In general, the answer is no. For example, consider the 4elt graph, which is planar as mentioned in [24]. The eigen-projection layout of this graph, which is displayed in Fig. 1(b), contains clear crossings in its top-right portion. An easier example is obtained by connecting two corners of a squared grid. The resulting graph is planar, but eigen-projection will draw it with edge crossings. An open question is whether can we define a family of planar graphs for which eigen-projection produces crossing-free layouts.

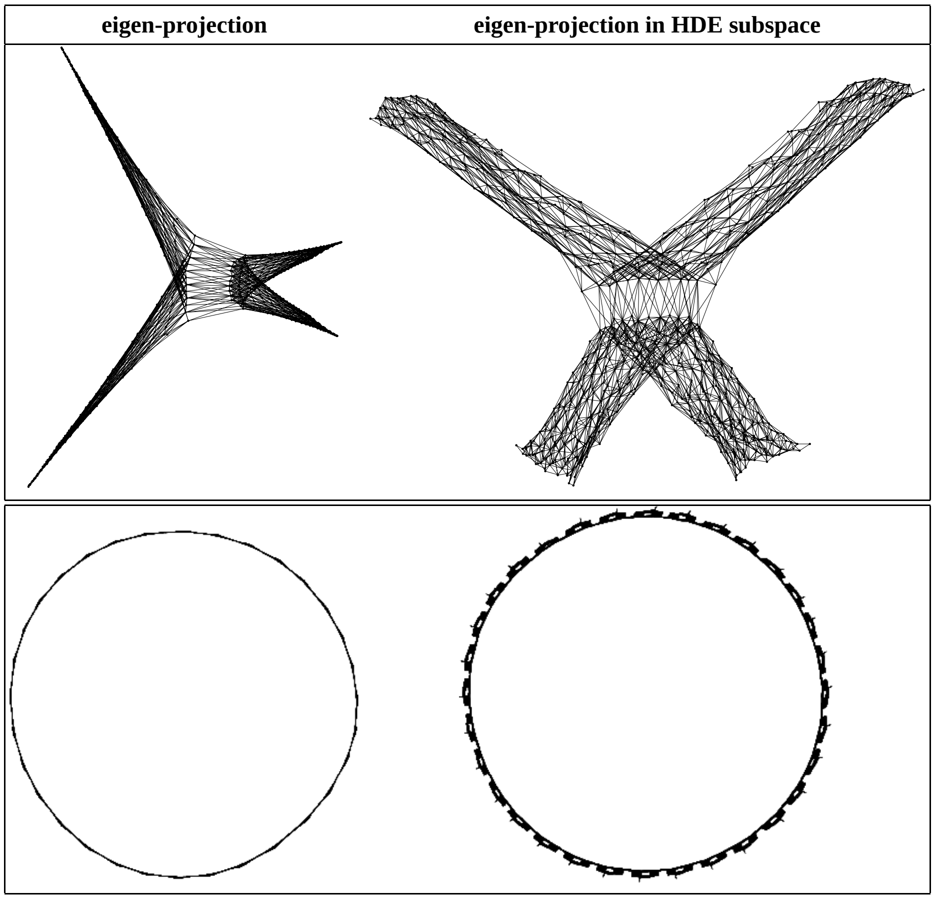

Fig. 7. Comparison of eigen-projection (left) and eigen-projection optimized in HDE subspace (right). The results are given for the two graphs: **top:** the Bfw782a graph [29] (|V|=782, |E|=3,394), and **bottom:** the Finan512 graph [10] (|V|=74,752, |E|=261,120).

Running time

A distinct advantage of optimization within the HDE subspace is the substantial reduction of running time thanks to replacing the $n \times n$ eigen-equation with an $m \times m$ eigen-equation. We cannot provide a recipe for the value of m, but in all our experiments $m = 50$ served us well, regardless of the graph's size. Note that unlike all iterative eigen-solvers (including the rapid algebraic-multigrid implementation in [20]) for which the number of iterations depends on the structure of the matrix, the running time of our algorithm depends only on the graph's size and is $O(m^2 n + m|E|)$. Moreover, the dimensionality of the drawing has virtually no effect on the running time, whereas for (unconstrained) eigen-projection running time is linear in the dimensionality of the layout (e.g., time for drawing a graph in 3-D will grow by ~50% relative to drawing it in 2-D).

Table 1 provides the actual running time of the various components of the subspace-constrained algorithm, as measured on a single-processor Pentium IV 2GHz PC. In addition to the total running time, we also provide the time needed for computing and orthogonalizing the HDE subspace (in the HDE-titled column), and the time needed for calculating the matrix $\mathcal{X}^T L \mathcal{X}$ (in the last column).

graph	$\mathbf{\|V\|}$	$\mathbf{\|E\|}$	**running time (sec.)**		
			total	**HDE**	$\mathcal{X}^T L \mathcal{X}$
516 [10]	516	729	0.02	0.00	0.00
Bfw782a [29]	782	3,394	0.06	0.02	0.00
Fidap006 [29]	1651	23,914	0.06	0.03	0.02
4970 [10]	4970	7400	0.77	0.09	0.64
3elt [10]	4720	13,722	0.77	0.09	0.64
Crack [10]	10,240	30,380	1.80	0.25	1.45
4elt2 [10]	11,143	32,818	1.84	0.28	1.52
4elt [10]	15,606	45,878	2.59	0.44	2.13
Sphere [10]	16,386	49,152	2.91	0.55	2.33
Fidap011 [29]	16,614	537,374	3.28	0.73	2.52
Finan512 [10]	74,752	261,120	8.17	2.83	5.30
Sierpinski (depth 10)	88,575	177,147	13.89	3.19	10.56
grid 317 $\times$ 317	100,489	200,344	7.59	3.28	4.24
Ocean [10]	143,437	409,593	25.73	8.00	17.50

Table 1. Running time (in seconds) of the various components of the algorithm

8 Discussion

In this paper we have presented a spectral approach for graph drawing, and justified it by studying three different viewpoints for the problem. The first viewpoint is based on solving a constrained energy minimization problem. We shaped the problem so that it shares much resemblance with force directed graph drawing algorithms (for a survey refer to [2, 3]). Compared with other force-directed methods, the spectral approach has two major advantages: **(1)** Its global optimum can be computed efficiently. **(2)** The energy function contains only $O(|E|)$ terms, unlike the $O(n^2)$ terms appearing in almost all the other force-directed methods. A unique feature of our energy-based derivation is that it optimizes a multidimensional layout, whereas previous energy-based approaches to spectral layouts [4, 1, 11] deal with optimizing only a 1-D layout.

A second viewpoint shows that spectral methods place each node at the centroid of its neighbors with some well defined deviation. This new interpretation provides an accurate description of the aesthetic properties of spectral drawing and also explains the relation between "nice" layouts and eigenvectors in a very direct manner.

We have also introduced a third viewpoint, showing that a kind of spectral drawing is the limit of an iterative process, in which each node is placed at the centroid of its neighbors. This viewpoint does not only sharpen the nature of spectral drawing, but also provides us with an aesthetically-motivated algorithm. This is unlike other algorithms for computing eigenvectors, which are rather complicated and far from having an aesthetic interpretation.

In addition to the theoretical analysis of spectral layouts, we suggested a novel practical algorithm that significantly accelerates the layout computation based on the notion of optimizing within a subspace. To this end, we described how to construct an appropriate subspace with a relatively low dimensionality that captures the "nice" layouts of the graph. This way, each axis of the drawing is a linear combination of a few basis vectors, instead of being an arbitrary vector in $\mathbb{R}^n$ (n is the number of nodes). The resulting layout might be the final result of serve as a smart initialization for an iterative eigen-solver.

References

1. Y. Koren, "On Spectral Graph Drawing", *Proc. 9th Inter. Computing and Combinatorics Conference (COCOON'03)*, LNCS 2697, Springer-Verlag, 496–508, 2003.
2. G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall, 1999.
3. M. Kaufmann and D. Wagner (Eds.), *Drawing Graphs: Methods and Models*, LNCS 2025, Springer Verlag, 2001.
4. K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Management Science* **17** (1970), 219–229.

5. F.R.K. Chung, *Spectral Graph Theory*, CBMS Reg. Conf. Ser. Math. 92, American Mathematical Society, 1997.
6. B. Mohar, "The Laplacian Spectrum of Graphs", *Graph Theory, Combinatorics, and Applications* **2** (1991), 871–898.
7. J. Shi and J. Malik, "Normalized Cuts and Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22** (2000), 888–905.
8. A. Pothen, H. Simon and K.-P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs", *SIAM Journal on Matrix Analysis and Applications,* **11** (1990), 430–452.
9. M. Juvan and B. Mohar, "Optimal Linear Labelings and Eigenvalues of Graphs", *Discrete Applied Math.* **36** (1992), 153–168.
10. Walshaw's collection: `www.gre.ac.uk/~c.walshaw/partition`
11. J. Shawe-Taylor and T. Pisanski, "Characterizing Graph Drawing with Eigenvectors", Technical Report CSD-TR-93-20 (Royal Holloway, University of London, Department of Computer Science, Egham, Surrey TW20 0EX, England).
12. D.E. Manolopoulos and P.W. Fowler, "Molecular Graphs, Point Groups and Fullerenes", *J. Chem. Phys.* **96** (1992), 7603–7614.
13. U. Brandes and T. Willhalm, "Visualizing Bibliographic Networks with a Reshaped Landscape Metaphor", *Proc. 4th Joint Eurographics - IEEE TCVG Symp. Visualization (VisSym'02)*, pp. 159-164, ACM Press, 2002.
14. P. Eades, "A Heuristic for Graph Drawing", *Congressus Numerantium* **42** (1984), 149-160.
15. T. M. G. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement", *Software-Practice and Experience* **21** (1991), 1129-1164.
16. A. Webb, *Statistical Pattern Recognition*, Arnold, 1999.
17. L. Carmel, Y. Koren and D. Harel, "Visualizing and Classifying Odors Using a Similarity Matrix", *Proc. 9th International Symposium on Olfaction and Electronic Nose (ISOEN'02)*, IEEE, to appear, 2003.
18. W. T. Tutte, "How to Draw a Graph", *Proc. London Math. Society* **13** (1963), 743–768.
19. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1996.
20. Y. Koren, L. Carmel and D. Harel, "Drawing Huge Graphs by Algebraic Multigrid Optimization", *Multiscale Modeling and Simulation* **1** (2003), 645–673, SIAM.
21. P. Gajer, M. T. Goodrich and S. G. Kobourov, "A Multi-dimensional Approach to Force-Directed Layouts of Large Graphs", *Proc. 8th Graph Drawing (GD'00)*, Lecture Notes in Computer Science, Vol. 1984, pp. 211–221, Springer-Verlag, 2000.
22. D. Harel and Y. Koren, "A Fast Multi-Scale Method for Drawing Large Graphs", *Journal of Graph Algorithms and Applications* **6** (2002), 179–202. Earlier version: *Proc. 8th Graph Drawing (GD'00)*, Lecture Notes in Computer Science, Vol. 1984, Springer-Verlag, pp. 183–196, 2000.
23. A. Quigley and P. Eades, "FADE: Graph Drawing, Clustering, and Visual Abstraction", *Proc. 8th Graph Drawing (GD'00)*, Lecture Notes in Computer Science,Vol. 1984, pp. 197–210, Springer Verlag, 2000.

24. C. Walshaw, “A Multilevel Algorithm for Force-Directed Graph Drawing”, *Journal of Graph Algorithms and Applications* **7** (2003), 253-285.
25. D. Harel and Y. Koren, “Graph Drawing by High-Dimensional Embedding”, *Proc. 10th Graph Drawing (GD’02)*, LNCS 2528, pp. 207–219, Springer-Verlag, 2002.
26. T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
27. D. S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, 1996.
28. Y. Koren and L. Carmel, “Robust Linear Dimensionality Reduction”, *IEEE Transactions on Visualization and Computer Graphics* **10** (2004), 459–470.
29. The Matrix Market collection: `math.nist.gov/MatrixMarket`