Latent Semantic Analysis and Fiedler Embeddings[*]

Bruce Hendrickson[†]

Abstract

Latent semantic analysis (LSA) is a method for information retrieval and processing which is based upon the singular value decomposition. It has a geometric interpretation in which objects (e.g. documents and keywords) are placed in a low-dimensional geometric space. In this paper, we derive an alternative algebraic/geometric method for placing objects in space to facilitate information analysis. We show that our method is closely related to LSA, and essentially equivalent for particular choices of scaling parameters. We then show that our approach supports a number of generalizations and extensions that existing LSA approaches cannot handle.

1 Introduction

Latent semantic analysis (LSA) is a well–known tool for information retrieval and analysis. The canonical example of LSA begins with a term–document matrix in which matrix rows correspond to key-words or terms, and matrix columns are documents. A nonzero value in the matrix means that the corresponding document contains the corresponding term. This vector space model of information is due to Salton [14].

Instead of working directly with this matrix, LSA replaces it with a low rank approximation using the singular value decomposition [6]. A variety of interpretations of LSA have been proposed. It is a noise reduction technique in which only the most significant parts of the term–document matrix are retained. Alternatively, it is a method for mapping terms and documents into geometric spaces, after which geometric algorithms can facilitate analysis.

The purpose of this paper is to derive a novel algebraic algorithm for placing terms and documents in space. Our approach is closely related to LSA, and we show it to be essentially equivalent under certain scaling choices. Whereas traditional LSA is motivated by a matrix approximation argument, our alternative follows from a geometric optimization problem. Besides providing a fresh perspective on LSA, we show that our approach allows for novel generalizations and extensions that are not possible with traditional approaches. For instance, our methodology supports queries that involve both terms and documents, e.g. "with these terms AND similar to these documents". As another example, unlike existing LSA techniques, our approach allows for the consideration of term–term and document–document similarities in generating the geometric embedding. The former might come from co–citation or link analysis, while the latter could be provided by a thesaurus.

In §2 we review the intuition and mathematics underlying traditional LSA. In §3 we introduce a simple geometric optimization problem and derive an algebraic solution. Then in §4 we demonstrate that the solution to our geometric problem is essentially equivalent to LSA. Finally, in §5, we discuss some of the benefits of our alternative derivation, including some ideas for extending LSA in new ways. In particular, we show how it allows for new kinds of relationships to be included in the same mathematical framework.

As we will discuss in §3, the geometric problem we use to motivate our approach reduces to the calculation of eigenvectors of the Laplacian matrix of a graph. Laplacian eigenvectors have been used in a wide range of applications in combinatorial optimization including graph partitioning [10], clustering [7], and linear arrangement [11]. For some of the development in this paper, we will be interested in the Laplacian eigenvectors of bipartite graphs. The resulting matrix has a natural 2×2 block structure. Others have used Laplacian eigenvectors of bipartite graphs e.g. Berry, et al. for reordering term/document matrices [5], Dhillon [7] and Zha, et al. [15] for data clustering, and Newton, et al. for graph drawing [13]. But to our knowledge, the connection to LSA described in this paper is new. In a paper similar in spirit to this one, Bartell, et al. have shown that LSA is equivalent to a special case of a different geometric optimization problem known as multidimensional scaling [1].

2 Latent Semantic Analysis

In this section we briefly sketch the fundamental operations in latent semantic analysis (also known as latent semantic indexing). We follow the traditional derivation in which LSA is motivated by an optimal matrix approximation. More comprehensive presentations can

[*]This work was supported by the LDRD program at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE under contract number DE-AC-94AL85000.

[†]Discrete Algorithms & Math Dept., Sandia National Labs. Email: bah@sandia.gov

be found in some of the citations e.g. [6, 4, 2, 3].

The canonical example of LSA begins with a $t \times d$ term–document matrix A. Each row of A is associated with a keyword or term, and each column is a document. A matrix entry $A(i, j)$ is a non-negative value which encodes the importance of the term i in the document j. There is an extensive literature on methods for generating such a matrix from a corpus of documents, but the construction process is beyond the scope of the current paper. Often, the term–document matrix is scaled to achieve some attractive normalization property, or to weight some terms or documents more heavily than others. Thus, we will consider the more general *scaled* term–document matrix $B = D_t A D_d$, where D_t and D_d are non-negative diagonal matrices of size $d \times d$ and $t \times t$ respectively.

In this *vector space* model of information, a document is described as a (weighted) vector of terms of which it is comprised. Two documents are similar if the inner product of their vectors is large. Thus, the matrix $B^T B$ describes the set of inter-document similarities. A query q is also a (weighted) vector of terms. The answer to a query is a set of documents that are similar to it, e.g. documents whose vectors have large inner-products with the query vector, that is large values in $B^T q$. (Most commonly, similarity is measured in angular distance, that is, as a direction cosine between the two vectors, which is just a normalized version of the inner product.)

Unfortunately, this simple model has a variety of well-known shortcomings. Most notably, small differences in vocabulary (e.g. *car* instead of *automobile*) can make documents look different from queries, even if their topics are overlapping. LSA attempts to address this problem through compression and noise reduction. Specifically, LSA uses matrix transformations to retain only the most significant portions of B, and then performs queries in this transformed space.

More formally, LSA is constructed around the singular value decomposition (SVD) of B,

$$B = U \Sigma V^T,$$

Where U and V are orthogonal matrices and Σ is diagonal and non-negative. The diagonal values of Σ are ordered to be non-increasing. In LSA, the matrix is approximated by a truncated SVD in which the first k diagonal values of Σ are retained, but the rest are set to zero. That is,

$$B \approx B_k = U_k \Sigma_k V_k^T,$$

where U_k is $t \times k$, Σ_k is $k \times k$ and V_k is $d \times k$. The truncated SVD is the best rank k approximation to B in the Frobenius norm.

The truncated SVD can be thought of as generating a k-dimensional embedding of the terms and documents. However, it is important to note that the term coordinates and the document coordinates are distinct entities. The notation in the field is inconsistent with respect to the scaling factors, but we choose to define the columns of $\Sigma_k^{1/2} V_k^T$ as the *document vectors* and $\Sigma_k^{1/2} U_k^T$ as the *term vectors.*

With the truncated SVD approximation to B, the inner-products required for document-document similarities can now be approximated as $B^T B \approx B_k^T B_k = (\Sigma_k^{1/2} V_k^T)^T \Sigma_k (\Sigma_k^{1/2} V_k^T)$, that is, as the inner product of document vectors, scaled by Σ_k.

Given a query vector q, we want to embed q into the document space in such a way that inner products with document vectors approximate $B^T q$. It is straightforward to see that this is achieved by letting the transformed query vector $\hat{q}$ be $\Sigma_k^{1/2} U_k^T q$, with a standard, unscaled inner product. (N.B. Alternatively, we could have used an inner product scaled by Σ_k as for document-document comparisons, in which case $\hat{q}$ would be $\Sigma_k^{-1/2} U_k^T q$. Both approaches lead to interpretation challenges since either the scaling of queries differs from that of documents, or the inner product differs for different kinds of questions.)

These procedures are summarized in Figure 1.

LSA document Embedding:
Given scaled term/document matrix B.
(1) Compute truncated SVD of B, $U_k \Sigma_k V_k^T$.
(2) The position of document i is $\Sigma_k^{1/2} V_k^T e_i$.

Querying:
Given Document embedding, and query vector q.
(1) Compute query location $= \Sigma_k^{1/2} U_k^T q$.
(2) Return documents nearest to query point
(nearest in angular distance).

Figure 1: LSA algorithms for term/document embeddings and querying.

3 Graphs, Laplacian Eigenvectors, and Fiedler Embeddings

As sketched above in §2, the traditional derivation of LSA is based upon optimal matrix approximations. But informally, the method succeeds in mapping documents into geometric space in such a way that similar documents are close to each other. In this section, we will pick up on this geometric closeness objective and develop an alternative algebraic method that explicitly tries to optimize closeness. For reasons that will be clear shortly, we will call our approach a *Fiedler embedding* algorithm. In §4, we will discuss the mathematical relationship between traditional LSA and our methodology.

Our approach begins with a graph $G = (V, E)$ in which V is a set of vertices and E is a set of vertex pairs known as edges. An edge (i, j) connecting vertices i and j has a non-negative weight $w_{i,j}$ which describes how similar the two vertices are. Larger weights correspond to a greater degree of similarity. In §4 we will look at the special case where the vertices are terms and documents, and the similarities are entries of the scaled term–document matrix B. But for now, we will consider the more abstract and general problem.

Our goal is to place the vertices of the graph into a low-dimensional geometric space in such a way that similar vertices are close to each other (i.e., edge lengths will be short). Geometric embeddings of graphs can be useful for a variety of reasons, but for the purposes of this paper, our eventual goal is the same of the goals of LSA. We hope to use geometric proximity as a way to identify vertices that are similar to each other, even if they don't have an edge between them.

The geometric embedding problem can be posed as an algebraic minimization. There are many ways to mathematically describe such an embedding, but one will be particularly useful. Specifically, we choose to find points in k space that minimize the weighted sum of the square of edge lengths. That is, if p_r is the location of vertex r, then

$$\text{Minimize} \sum_{(r,s)\in E} w_{r,s}|p_r - p_s|^2$$

If the number of vertices is n, and the geometric space has dimensionality k, then the positions of the vertices can be considered to be an $n \times k$ matrix X. Define the *Laplacian matrix* L as follows.

$$L(i,j) = \begin{cases} -w_{i,j} & \text{if } e_{ij} \in E \\ \sum_k w_{i,k} & \text{if } i = j \\ 0 & \text{Otherwise.} \end{cases}$$

That is, the Laplacian is the negative of the matrix of weights, except that the diagonal values are chosen to make row-sums zero. Note that L is symmetric and positive semi-definite. After a bit of algebra, our minimization problem can now be rewritten as

$$\text{Minimize Trace}(X^T L X) \tag{3.1}$$

This minimization problem is poorly posed for three reasons. First, it is invariant under translations. To avoid this problem we can add a constraint to make the median of the point set be the origin. That is,

$$\text{(Constraint 1)} \quad X_i^T 1^n = 0 \ \text{ for } i = 1 \ldots k, \tag{3.2}$$

where 1^n denotes the vector of n ones.

Second, even with this constraint the minimization problem has the trivial solution of placing all the vertices at the origin. To avoid this, we can simply insist that the sum of squares of each coordinate value is nonzero. That is,

$$\text{(Constraint 2)} \ \text{ for } \ i = 1, \ldots, k \quad X_i^T X_i = \delta_i \tag{3.3}$$

for some positive values δ_i. Without loss of generality, we will choose to order the axes so that the δ_i values are non-increasing. We will have more to say about these values when we compare the Laplacian approach to LSA in §4.

Finally, we want to ensure that each coordinate conveys distinct information. We accomplish this by imposing the constraint that the vector of coordinate values in each dimension is orthogonal to the coordinate values from any other dimension.

$$\text{(Constraint 3)} \ \text{ for } \ i \neq j \quad X_i^T X_j = 0. \tag{3.4}$$

Denoting the diagonal matrix of δ values by Δ, we can combine constraints 2 and 3, resulting in the following optimization problem.

$$\begin{aligned} &\text{Minimize Trace}(X^T L X) \\ &\text{Subject to :} \\ &\qquad (i)\ X_i^T 1^n = 0 \ \text{ for } i = 1 \ldots k, \\ &\qquad (ii)\ X^T X = \Delta \end{aligned} \tag{3.5}$$

L is positive semi-definite, and it has an eigenvector proportional to 1^n with eigenvalue 0. If the graph is connected, then all other eigenvalues are positive [8, 9]. Now sort the eigenvalues λ_i in non-decreasing order, and let the corresponding normalized eigenvectors form a matrix Q so that $L = Q^T \Lambda Q$. Note that constraint *(i)* disallows columns of X from having any component in Q_1, but since the eigenvectors of a symmetric matrix are orthogonal, all other eigenvectors of L satisfy this constraint.

We can represent X in terms of Q; that is, let $X = QZ$ for some $n \times k$ matrix Z. Our constrained minimization problem can now be written as

$$\begin{aligned} &\text{Minimize Trace}(Z^T Q^T L Q Z) = \text{Trace}(Z^T \Lambda Z) \\ &\text{Subject to :} \\ &\qquad (i) Z(1,*) = 0 \\ &\qquad (ii) Z^T Z = \Delta \end{aligned} \tag{3.6}$$

The first constraint is a consequence of the exclusion of any contribution from the first eigenvector of L bh(the constant vector). It is not hard to see that the solution to (3.6) is to have the columns of Z span the space spanned by the unit vectors $e_2, \dots, e_{k+1}$. Furthermore, the large values of Δ should be paired with the small values of Λ. Recall that the diagonal entries in Λ are non-decreasing, while those of Δ are non-increasing. If these diagonal matrices have non-repeating values, then $Z = \Delta^{1/2}[e_2, \dots, e_{k+1}]$. (If Δ or Λ have repeated values, then the solution is degenerate and any basis for the subspace spanned by the corresponding columns of repeated values are also minimizers.)

It follows that a solution to Eq. 3.5 is

$$X = \Delta^{1/2}[Q_2, \dots, Q_{k+1}]. \tag{3.7}$$

That is, the coordinates of vertex i are just the ith entries of eigenvectors $2, \dots, k+1$ of L, scaled by the square roots of the δ values. We will call this solution a *Fiedler embedding* in honor of Miroslav Fiedler's pioneering work exploring the relationships between graphs and Laplacian eigenvectors. It is worth noting that these eigenvectors are also low energy oscilations for certain mass and spring systems.

3.1 Queries Once we have embedded the graph in space, we can use geometry to identify pairs of similar vertices. Two vertices might not have an edge between them, but they might still be similar if they have many neighbors in common. Since the Fiedler embedding tries to keep edge lengths short, vertices sharing neighbors should be placed close to each other. So given a vertex, its geometrically nearest neighbors are natural candidates for similarity. In this discussion, Euclidean distance is the most natural metric, but as noted above, LSA traditionally uses angular distance, measured by direction cosines.

Now suppose we want to add a new vertex to the geometric space, and this vertex is known to be similar to some of the vertices we have already placed. In the language of LSA, this new vertex corresponds to a query and its known similarity values comprise a query vector. Once the new vertex is given coordinates, we could use geometric algorithms to find nearby vertices, and these would be the output of the query.

As with the derivation of the Fiedler embedding, we wish to place this new vertex into the space in such a way that it is near to vertices it is known to be similar to. Mimicking the development above, we will position the new vertex to minimize the (weighted) sum of squares of distances to the vertices it has an edge to. That is, we wish to find a position p_x which solves

$$\text{Minimize} \sum_{(s,x)\in E} w_{s,x}|p_s - p_x|^2.$$

Setting the derivative to zero, it is easy to see that the solution to this problem is

$$p_x = \sum_{(s,x)\in E} w_{s,x}p_s / \sum_{(s,x)\in E} w_{s,x} = \Delta^{1/2}Q^T q/||q||_1.$$

where q is the vector of similarity values for the new vertex (values of w in the derivation above).

To summarize, the Fiedler procedure for constructing coordinates and querying are sketched in Figure 2.

Fiedler Embedding:
Given a graph, W, a non-negative, symmetric matrix of vertex similarities, and Δ a diagonal, non-negative matrix of coordinate scalings.
(1) Generate Laplacian matrix L from W.
(2) Compute eigenvectors u_j corresponding to the $k+1$ smallest eigenvalues of L.
(3) Let $Q = [u_2, \dots, u_{k+1}]$.
(4) Position of vertex i is $\Delta^{1/2}Q^T e_i$.

Querying:
Given Graph embedding and new entity with similarity/query vector q.
(1) Compute query point $= \Delta^{1/2}Q^T q/||q||_1$.
(2) Return vertices nearest to query point (nearest in Euclidean distance).

Figure 2: Fiedler algorithms for graph embedding and querying.

4 Relationship Between LSA and Fiedler Embeddings

Comparing Figures (2) and (1), two differences stand out. First, one is using Laplacian eigenvectors and the other uses left and right singular vectors. Below we argue for the essential equivalence of these two decompositions. Second, the Fiedler approach involves scaling by Δ (the normalization constraints from the minimization problem) while LSA uses scaling by the singular values Σ_k. As the normalization values were unspecified in the derivation of the Laplacian approach, we can, if desired, choose to set them equal to the singular values.

The Fiedler embedding from §3 involves small eigenvectors of a Laplacian matrix, while the LSA embedding uses large singular vectors from the scaled term–document matrix. Although seemingly quite different, these spaces are very closely related. To see this, interpret the scaled term–document matrix to express similarities between terms and documents. Now imagine a graph in which terms and documents are both treated as vertices, and similarity values come from the corresponding entries in the scaled term–document matrix. The Laplacian matrix of this graph will have a block structure, reflecting the bipartite graph that comes from having only term/document similarities but no term/term or document/document entries. That is,

$$L = \begin{pmatrix} D_1 & -B \\ -B^T & D_2 \end{pmatrix}, \tag{4.8}$$

where D_1 and D_2 are diagonal matrices which make the row sums zero. The Fiedler embedding consists of eigenvectors of this matrix with small eigenvalues.

Now consider the space used in traditional LSA. Recall that the left and right singular vectors of B are closely related to the eigenvectors of a larger matrix M, where

$$M = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}. \tag{4.9}$$

That is, if (σ, u, v) comprises a singular triplet of B, then $(\sigma, v : u)$ is an eigen pair of M, where $u : v$ indicates concatenation of u and v. So the space used in LSA consists of eigenvectors of M with large eigenvalues. The matrix $I - M$ will have identical eigenvectors to M, but the ordering of the eigenvalues will be reversed. Consequently, LSA can be considered to be a projection into the space of the eigenvectors of the smallest eigenvalues of

$$I - M = \begin{pmatrix} I & -B \\ -B^T & I \end{pmatrix}. \tag{4.10}$$

A comparison of Eq. 4.10 with Eq. 4.8 illustrates the close similarity between the spaces utilized by LSA and by the Fiedler embedding. The matrices are identical in the case where D_1 and D_2 are identity matrices. This will happen when the term–document matrix has been scaled to be doubly stochastic, that is, to have unity row and column sums. Recall, however, that LSA uses the first k eigenvectors, while the Fiedler embedding skips the first and uses the next k since it knows that the first is a constant vector.

Now consider querying operations in the LSA and Fiedler approaches. In the term–document setting, a query will be a vector of terms, so in the language of eigenvectors of $I = M$ it will only have nonzeros in the first t entries of a larger $t + d$ vector. The query operations from Figs. 1 and 2 are structurally similar. As discussed above the spaces represented by U_k in LSA and Q^T in the Fiedler embedding are very closely related. The scaling values Δ in the Fiedler approach are free parameters, and so, if desired, can be chosen to be equal to the singular values Σ_k. With this equivalence, the only remaining difference in the query operation has to do with normalization of the output vector. Since LSA typically uses direction cosines, this normalization is irrelevant. The normalization by $||q||_1$ in the Fiedler algorithm is appropriate for a Euclidean distance metric.

In summary, the Fiedler embedding approach to information retrieval is very closely related to LSA in the case where the data is represented as a term–document matrix. When B is scaled to be doubly stochastic, the eigenspaces uses by the two approaches are identical, except that LSA uses the first k vectors while the Fiedler embedding uses the second through the $k + 1$st. If the normalization values in the Fiedler embedding are chosen to be the singular values of B, then the query operations are also very closely related.

However, there are no obvious reasons to choose these particular scalings. In fact, a doubly stochastic scaling of the term–document matrix forces all terms and documents to be treated equally, which may be undesirable. Instead, consider both LSA and the Fiedler embedding to be classes of algorithms with some free parameters. The discussion above argues that these algorithms are closely related and overlap for some choices of parameter settings.

5 Benefits of Fiedler Retrieval

As explained in §2, the Fiedler retrieval approach introduced in §3 can be viewed as a close cousin to traditional LSA. We believe it provides several distinct advantages over traditional LSA formulations. It provides a simple and intuitive way to explain the power of LSA. But it also has some more concrete advantages.

- The Fiedler embedding is an explicit attempt to place similar objects near to each other. This objective is often alluded to in the LSA literature, but with traditional presentations of LSA the mathematical underpinnings of nearness are opaque.
- Unlike traditional LSA, in Fiedler retrieval terms and documents are treated equivalently and colocated in the same space. As itemized below, this creates opportunities for several extensions to traditional LSA. It is worth noting that algorithms for symmetric eigenvalue computations are closely related to SVD algorithms, so the cost of Fiedler retrieval should be similar to that of the SVD computation. With terms and documents being equivalent, algorithms for queries are simplified and unified. This disproves a claim made by Deerwester, et al. [6]. "... it is not possible to make a single configuration of points in space that will allow both between [term/document] and within [term/term] comparisons." Among the extensions

made possible by the Fiedler viewpoint are the following.

- In the standard development of LSA it is assumed that the only usable information is term/document connections. LSA does not naturally allow for inclusion of any additional information related to term/term or document/document similarities. However, nothing in the derivation of the Fiedler retrieval algorithm from §3 exploited the bipartite nature of the graph. That is, the diagonal blocks of the Laplacian matrix need not be diagonal matrices. In other words, we could have explicitly added information about document/document or term/term similarities into our construction. For instance, citation analysis or hyperlinks could have provided document/document similarities or a thesaurus or dictionary could have offered term/term similarities. In a web setting, link and text analysis can be combined into a common algebraic framework. In a recommender system, product/product similarities could be included, enhancing current methods that just include consumer/product information. Fiedler retrieval allows for a principled inclusion of such information.
- Similarly, Fiedler retrieval supports queries that include both term and document similarities. For example, one could search for documents similar to a few particular documents and a few specific terms. These kinds or cross-queries are problematic for traditional LSA.
- LSA is traditionally constrained to capturing the relationships between two classes of objects, e.g. terms and documents. It is unclear how to extend the standard methodology to more object classes, e.g. terms, documents and authors. Fiedler retrieval is not limited in this way. The Laplacian matrix will have a logical block structure, with as many block-rows and block-columns as object classes. That is, the rows and colums of the matrix will have entries for terms, documents and authors. The diagonal blocks will capture similarities between objects of the same type (e.g. authors with authors), while the off-diagonal blocks will encode similarities between disparated types. Several researchers have recently proposed to solve this problem by extending the term–document matrix to higher dimension and using multilinear algebra techniques (e.g. the TOPHITS approach of Kolda and Bader [12]). Although these approaches are mathematically elegant, computations on tensors are much more challenging than those on matrices. With Fiedler retrieval, we retain many of the advantages of tensors, while retaining the algorithmic and mathematical benefits of working with two-dimensional matrices.

Acknowledgements

The ideas in this paper have benefitted from discussions with Tammy Kolda, Liz Jessup, Mike Berry, Inderjit Dhillon, Brett Bader, Erik Boman, Chris Ding and Petros Drineas.

References

[1] B. T. Bartell, G. W. Cottrell, and R. K. Belew, *Latent semantic indexing is an optimal special case of multidimensional scaling*, in Proc. Annual International ACM SIGIR Conf. Research and Development in Information Retrieval, ACM, 1992, pp. 161–167.

[2] M. Berry and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, SIAM, Philadelphia, 1999.

[3] M. Berry, Z. Drmac, and E. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335–362.

[4] M. Berry, S. Dumais, and G. O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review, 37 (1995), pp. 573–595.

[5] M. W. Berry, B. Hendrickson, and P. Raghavan, *Sparse matrix reordering schemes for browsing hypertext*, in Lectures in Applied Mathematics, vol. 32, AMS, 1996, pp. 99–123.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41 (1990), pp. 391–407.

[7] I. S. Dhillon, *Co-clustering documents and words using bipartite spectral graph partitioning*, in Proc. 7th Intl. Conf. Knowledge Discovery & Data Mining, ACM, 2001.

[8] M. Fiedler, *Algebraic connectivity of graphs*, Czech. Math. Journal, 23 (1973), pp. 298–305.

[9] ———, *A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory*, Czech. Math. Journal, 25 (1975), pp. 619–633.

[10] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.

[11] M. Juvan and B. Mohar, *Optimal linear labelings and eigenvalues of graphs*, Discrete Appl Math., 36 (1992), pp. 153–168.

[12] T. G. Kolda, B. W. Bader, and J. P. Kenny, *Higher-order web link analysis using multilinear algebra*, in Proc. IEEE Intl. Conf. Data Mining, November 2005.

[13] M. C. Newton, O. Sykora, and I. Vrto, *Two new heuristics for the 2-sided bipartite crossing number*, in Proc. 10th Intl. Symp. Graph Drawing, Lecture Notes in Computer Science 2528, Springer, 2002, pp. 312–319.

[14] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw Hill, New York, 1983.

[15] H. Zha, X. He, C. Ding, M. Gu, and H. Simon, *Bipartite graph partitioning and data clustering*, in Proc. 10th Intl. Conf. Info. & Knowledge Mgmt., ACM, 2001.