Internet Mathematics Vol. 10: 263–286

Toward Quantifying Vertex Similarity in Networks

Charalampos E. Tsourakakis

Abstract. Vertex similarity is a major concept in network science with a wide range of applications. In this work we provide novel perspectives on finding (dis)similar vertices within a network and across two networks with the same number of vertices (graph matching). With respect to the former problem, we propose to optimize a geometric objective that allows us to express each vertex uniquely as a convex combination of a few extreme types of vertices. Our method has the important advantage of supporting efficiently several types of queries such as, which other vertices are most similar to this vertex? by using appropriate data structures and by mining interesting patterns in the network. With respect to the latter problem (graph matching) we propose the generalized condition number—a quantity widely used in numerical analysis— $\kappa(L_G, L_H)$ of the Laplacian matrix representations of G, H as a measure of graph similarity, where G, H are the graphs of interest. We show that this objective has a solid theoretical basis, and, we propose a deterministic and a randomized graph alignment algorithm. We evaluate our algorithms on both synthetic and real data. We observe that our proposed methods achieve high-quality results and provide us with significant insights into the network structure.

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uinm.

ISSN: 1542-7951 print

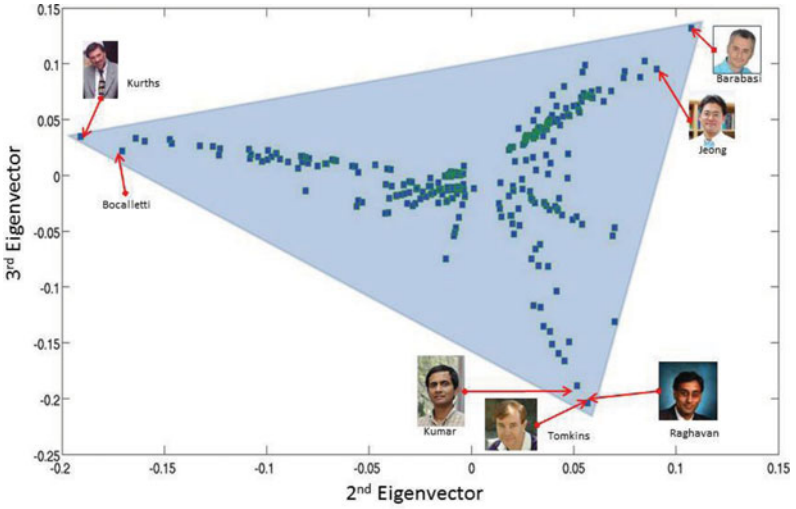
1. Introduction

Vertex similarity is an important network concept with a broad range of significant applications. Paradoxically, a major step toward the successful quantification of vertex similarity is finding a good definition of it. Typically, one is interested either in finding similar vertices in a given network or in finding similar vertices across two different networks. The former problem emerges in numerous applications in social networks such as link prediction and recommendation. It also emerges in the domain of privacy because the improved ability to predict an edge may be used for malicious purposes as well [Hay et al. 08]. The latter problem emerges also in various domains including graph mining, computer vision, and chemistry. The interested reader is urged to read [Zager 05] which contains a wealth of applications. In this work we provide novel perspectives on the two aforementioned problems. On purpose we state them abstractly using quotes in several places, in order to emphasize that two main contributions of our work are two novel formalizations of these problems. The first problem is: *given an undirected graph $G(V, E)$ and two vertices $u, v \in V$, how "similar" are u and v?* The second problem is: *given two graphs $G(V_G, E_G)$, $H(V_H, E_H)$ such that $|V_G| = |V_H|$, is there a permutation of the vertices of H that "reveals any similarities" between G and H? Can we find such a permutation efficiently?* We will refer to these problems as the *vertex similarity* and the *graph matching* problem, respectively.

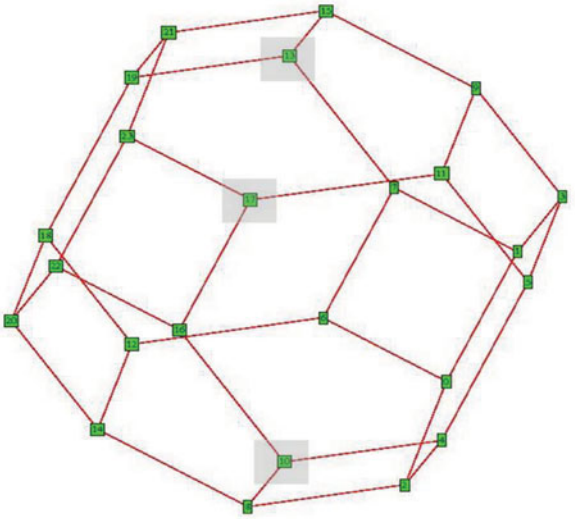
1.1. Paper Contributions and Road map

Our contributions are summarized as follows.

- For the vertex similarity problem:
  - We propose a novel approach, which is inspired by the concept of archetypal analysis [Cutler and Breiman 94]. We formalize our problem as optimizing a geometric objective, namely finding an enclosing simplex of minimum volume that is robust to outliers for a special cloud of points.
  - We propose an efficient algorithm for optimizing our objective. An output example of our method is shown in Figure 1(a), which shows a minimum-area 2-simplex $\mathcal{S}$ for a normalized Laplacian graph embedding [Belkin and Niyohi 03] of the largest component of the Netscience network, (see Table 1). Using the Euclidean distance between the mixture coefficients the smaller the distance is the more similar the vertices are we find that the vertices Prabhakar Raghavan, Ravi Kumar, and Andrew Tomkins are highly similar and that the three extreme points



(a) Vertex Similarity



(b) Graph Matching

Figure 1. (a) Minimum-area 2-simplex $\mathcal{S}$ for an informative embedding of the largest component of the Netscience network G (see Table 1 for the dataset details). $\mathcal{S}$ allows us to express each data point as a unique convex combination of its extreme points and, hence, call two vertices of G similar if their corresponding mixture coefficients are close. (b) Permutahedron $\mathcal{P}$ of the symmetric group S_4. The 3 shaded vertices of $\mathcal{P}$ define a hypothetical set of isomorphisms between G and H.

(archetypes) of $\mathcal{S}$ correspond to three influential groups of researchers. Specifically, the three vertices of the simplex lie close to Kurths and Bocalletti, Barabasi and Jeong, Kumar, and Raghavan, Tomkins, and Rajagopalan who are respectively, three authoritative groups of researchers on social networks.

- Our method has the advantage of supporting *efficiently* queries of the type, which other vertices are most similar to this vertex? and which are the most dissimilar vertices to this vertex? by the use of the appropriate data structures [Arya et al. 98].

- For the graph matching problem:
  - We introduce a novel criterion of similarity between two graphs G, H: the *generalized condition number* of their Laplacian matrix representations.
  - In contrast to frequently used heuristics, our criterion has a solid theoretical basis. Specifically, consider Figure 1(b), which shows the permutahedron $\mathcal{P}$ of the symmetric group S_4 [Burkard et al. 09] with three shaded vertices that correspond to three hypothetical permutations, which make's graph H identical to graph G. Theorem 4.1 proves that the global minimum of our proposed objective function occurs at the permutations that make H equal to G. Specifically, our proposed Metropolis chain in the limit of $\lambda \to +\infty$ ($\lambda \geq 1$ is a parameter in our chain) converges to the uniform distribution over the shaded set, i.e., $\pi(\text{Shaded Vertex}) = 1/3$, $\pi(\text{Nonshaded Vertex}) = 0$.
  - Despite the fact that in this work we focus on the restricted version of the graph matching problem where G, H have the same number of vertices, this does not make our work useless, for two main reasons. First, there exist applications where $|V_G| = |V_H|$ [Zager 05]. Second, our conceptual contribution is likely to be extendable to the more general case where $|V_G| \neq |V_H|$, as will be discussed in Section 6.
  - Our proposed method can also be used in conjuction with other graph matching methods, e.g., as a postprocessing tool. We explore this possibility in Section 5, where we show that this approach yields excellent practical performance.

The article is organized as follows: Section 2 briefly presents related work. Sections 3 and 4 present our proposed methods for the vertex similarity and the graph matching problem, respectively. Section 5 shows an experimental validation and evaluation of our proposed methods. Section 6 concludes the paper.

2. Related Work

In Sections 2.1, 2.2, and 2.3 we present work related to our proposed method in Section 3. In Sections 2.4, 2.5 we present work related to our proposed method in Section 4.

2.1. Vertex Similarity

The key idea that appears in different guises in the literature related to the vertex similarity problem is the following: two vertices are similar if their neighbors are similar. The recursive nature of this idea leads to recursive algorithms. It is worth pointing out that other measures of similarity exist: the number of common neighbors, Jaccard's coefficient, Salton's coefficient, the Adamic/Acar coefficient [Adamic and Adar 03], etc. These measures have significant shortcomings. For instance, two vertices may be highly similar even if they share no common neighbors [Leicht et al. 06].

The algorithm that has influenced and motivated a large and significant amount of research on vertex similarity is Kleinberg's Hypertext-Induced Topic Search (HITS) algorithm [Kleinberg 99]. Interestingly, [Blondel et al. 03] generalized HITS and provided a general scheme for finding the similarity of two vertices. Jeh and Widom proposed the Simrank algorithm [Jeh and Widom 02] to compute all-pairs vertex similarities in a graph. Leicht et al. propose another recursive measure of similarity closely resembling the centrality measure of Katz [Leicht et al. 06]. Recursive algorithms are closely connected to spectral graph theory. Additionally, spectral graph theory through random walks provides the basis for a rich set of similarity measures, including commute times and graph kernel methods [Fouss et al. 07]. Recently, nonnegative matrix factorization [Arora et al. 12] has been proposed in the context of role identification in social networks [Henderson et al. 12].

Vertex similarity has numerous applications such as link recommendation [Liben-Nowell and Kleinberg 03], schema matching [Melnik et al. 02], and privacy attacks [Hay et al. 08]. Our geometric perspective on the problem of vertex similarity in Section 3 has not been considered in the literature, to the best of our knowledge.

2.2. Archetypal Analysis

The idea of archetypal analysis was born by Breiman during his work on predicting the next-day ozone levels. Breiman proposed that each day could be quantified as a mixture of "extreme" or "archetypal" days [Cutler 10]. Culter

and Breiman introduced archetypal analysis and proposed an alternating minimization procedure [Cutler and Breiman 94]. Archetypal analysis has numerous applications in various fields including computational biology [Huggins et al. 07] and marketing [Riedesel 03].

2.3. Spectral Unmixing

Spectral unmixing is a central problem in spectral imaging. [Keshava 03] surveys existing algorithms for this problem. Of special interest to us is the geometric approach, inspired by Craig's seminal work [Craig 94], in which a minimum-volume simplex is fitted to the set of points.

The computational complexity of fitting a minimum-volume-enclosing simplex depends on the dimensionality k. Specifically, when $k = 2$ there exist efficient algorithms for finding the minimum-area enclosing triangle [O'Rourke et al. 86]. When $k = 3$, Zhou and Suri give an algorithm with complexity $O(n^4)$ [Zhou and Suri 00]. Packer showed that the problem is NP-hard when $k \geq \log(n)$ [Packer 02].

2.4. Graph Matching

The graph matching problem has attracted a lot of interest [Pattern Recognition Letters 03]. Umeyama proposed that instead of trying to find a permutation matrix P, i.e., one of the vertices of the Birkoff polytope [Burkard et al. 09] (it is a polytope whose vertices correspond to permutation matrices, see the closely related permutahedron in Figure 1(b)), which minimizes $||PA_GP^T - A_H||$, where A_G, A_H are the adjacency matrix representations of graphs G, H, one may relax the problem to finding an orthogonal matrix P' that minimizes the same objective [Umeyama 98]. Other methods relax the constraint of searching for a permutation matrix P to finding a doubly stochastic matrix.

Spectral approaches play a prominent role in matching two shapes, a key problem in computer vision. The problem of shape matching on preprocessing reduces to graph matching [Shapiro and Brady 92]. Spectral methods take as input two weighted graphs, each representing a shape and consisting typically of three steps [Bronstein et al. 10]. We believe that there may be fruitful connections between our proposed method in Section 4 and the first two steps of these methods. In the first step, the Laplacian embedding of the two shapes is computed. In the second step, a permutation matrix P and a sign matrix S, which matches the first k eigenvectors of these shapes to each other, are computed. Typically, this is done by finding the minimal cost assignment between these vectors. The final

step is the point registration of these two aligned embeddings, e.g., by using the expectation–minization (EM) algorithm.

When the two graphs of interest have a different number of vertices, the typical representation is the compatibility graph. Using this representation, the graph matching problem can be formulated as an integer quadratic problem (IQP), which can be tackled in various ways. Dominating approaches include semidefinite programming [Schellewald and Schnörr 05], spectral approaches [Bai et al. 04], linear programming relaxations [Klau 09] and the popular graduated assignment method [Gold and Rangarajan 96]. The latter relaxes the IQP into a nonconvex quadratic program and solves a sequence of convex optimization approximation problems. [Blondel et al. 03] use a generalization of the HITS method [Kleinberg 99] to find graph matchings. As we mentioned in Section 2.1, their method is also applicable to the vertex similarity problem. Other approaches include belief, propagation (BP) [Bayati et al. 09] and kernel-based methods [Smalter et al. 08].

2.5. Generalized Condition Number

A fundamental problem of linear algebra is solving the linear system of equations $Ax = b$ [Golub and Van Loan 96]. In the case of a preconditioned linear system, the corresponding quantity that determines the rate of convergence of the solver, e.g., preconditioned conjugate gradient, is the generalized condition number [Golub and Ye 02]. The definition follows:

Definition 2.1. [Gremban 96] Let A, B be two real matrices with the same null space $\mathbb{K}$; λ is a generalized eigenvalue of the ordered pair of matrices (A, B), also called "pencil", if there exists a vector $x \notin \mathbb{K}$ such that $Ax = \lambda Bx$. Let $\Lambda(A, B)$ be the set of generalized eigenvalues of the pencil (A, B). The generalized condition number $\kappa(A, B)$ is defined as the ratio of the maximum value $\lambda_{\max}(A, B)$ to the minimum value $\lambda_{\min}(A, B)$.

For every unit norm vector x, the following double inequality holds:

$$\lambda_{\min}(A, B)x^T Bx \leq x^T Ax \leq \lambda_{\max}(A, B)x^T Bx. \tag{2.1}$$

For the special case of interest where the pencil (L_G, L_H) is a pair of Laplacian matrices of two connected graphs G, H on the same vertex set, the condition number is given by the following expression:

$$\kappa(L_G, L_H) = \left(\max_{x^T 1=0} \frac{x^T L_G x}{x^T L_H x} \right) \left(\max_{x^T 1=0} \frac{x^T L_H x}{x^T L_G x} \right).$$

Notice that because G, H are connected, their null space is the same and specifically the span of the all-ones vector $\mathbf{1}$ [Golub and Ye 02]. Generalized eigenvalue problems of a special form have several important applications in computer science, see for instance [Belkin and Niyohi 03, Shi and Malik 00].

3. VertexSim: Vertex Similarity via Simplex Fitting

The main assumption of our proposed method is that each vertex is a "combination" of a few "extreme" types of vertices. This assumption lies conceptually close to archetypal analysis [Cutler 10]. We formalize mathematically the notions of "combination" and "extreme" geometrically in the following.

Our proposed algorithm is VertexSim shown as Algorithm 1. The algorithm takes as input the graph $G(V = [n], E)$, the dimension of the simplex we wish to fit, and a parameter γ, which tunes the sensitivity of the fitting algorithm to outliers. We assume that $k \ll n$. In the first step, we embed the graph G on the Euclidean space $\mathbb{R}^k$. Hence, each vertex is mapped to a k-dimensional point. It is worth emphasizing that typically real-world networks of small and medium size (up to several thousands of vertices and edges) have strong geometric structure. As the size grows, the geometry becomes less apparent (see also Section 6). There exist several methods to obtain an informative embedding of the graph. The majority of them are spectral [Lee and Verleysen 07]. In our experiments we choose the k smallest nontrivial eigenvectors, i.e., the eigenvectors corresponding the k smallest, nonzero eigenvalues of the normalized Laplacian [Belkin and Niyohi 03].

Algorithm 1. VertexSim

Require: Connected, undirected graph $G([n], E)$. Dimension k. Parameter γ.
(1) Embed the graph using the k smallest nontrivial eigenvectors of the normalized Laplacian of G.
(2) $[K, \{\theta_i\}_{i\in[n]}] \leftarrow$ Solve Optimization Problem 3.1 using gradient descent.
(3) For every pair of vertices (i, j) compute the Euclidean distance between the mixture coefficient vectors θ_i, θ_j.
(4) Add points $\{\theta_i\}_{i\in[n]}$ to a data structure supporting nearest neighbor search queries.

In the second step, we learn a simplex, i.e., a set of $k + 1$ affinely independent points, which encloses the cloud of points. The $k + 1$ vertices of the simplex are the extreme types of vertices, and each vertex is a convex combination of these types. The rationale behind the choice of a simplex is that each point is

expressed uniquely as a convex combination of the extreme points. This allows us to perform a quantitative analysis of vertex similarity and answer queries such as, which are the three vertices most similar to vertex v? with the use of appropriate data structures [Arya et al. 98]. Among all simplexes that fit the cloud of points we favor the one with the smallest volume, inspired by the seminal work of [Craig 94]. There exist a wide variety of off-the-shelf algorithms that compute a minimum-volume-enclosing simplex, and reliable implementations are publicly available. We use the method developed by [Tolliver et al. 10], which solves the following optimization problem, where $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^k$ is the cloud of points, $K = [v_0|\ldots|v_k]$ is a simplex in $\mathbb{R}^k$, and $\theta_i \in [0,1]^{k+1}$ for $i = 1, \ldots, n$ is the vector of mixture coefficients of point i:

$$\min_{K,\theta} : \sum_{i=1}^{s} |x_i - K\theta_i|_p + \gamma \log \texttt{vol}(K)$$
$$\forall \theta_i : \theta_i^T \mathbf{1} = 1,\ \theta_i \succeq 0 \tag{3.1}$$

The first term in the objective makes the formulation robust to outliers, see [Tolliver et al. 10]. We use $|x|_p$ to denote the p-norm of vector x. We choose $p = 1$. We need to derive the necessary partial derivatives. For completeness, we include here the computation. Let the simplex be represented by the vertex matrix $K = [v_0|...|v_k]$. Then,

$$\texttt{vol}(K) = c_k \cdot \texttt{det}\left(\Gamma^T K K^T \Gamma\right)^{1/2} = c_k \cdot \sqrt{\texttt{det}\ Q}, \tag{3.2}$$

where c_k is the volume of the unit simplex defined on $k+1$ points and Γ is a fixed vertexedge incidence matrix such that $\Gamma^T K = [v_1 - v_0|...|v_k - v_0]$. It follows that

$$\log \texttt{vol}(K) = \log c_k + \frac{1}{2} \log \texttt{det} Q \propto \log \prod_{d=1}^{k} \lambda_d(Q) = \sum_{d=1}^{k} \log \lambda_d(Q).$$

Hence, the gradient of $\log \texttt{vol}(K)$ is given by

$$\frac{\partial \log \texttt{vol(K)}}{\partial K_{ij}} = \sum_{d=1}^{k} \frac{\partial}{\partial K_{ij}} \lambda_d = \sum_{d=1}^{k} z_d^T (\Gamma^T E_{ij} E_{ij}^T \Gamma) z_d,$$

where the eigenvector z_d satisfies the equality $Q z_d = \lambda_d z_d$, and E_{ij} is the indicator matrix for the entry ij. To minimize the volume, we move the vertices along the paths specified by the negative log gradient of the current simplex volume.

Finally, in the third and fourth step's we compute the similarity between vertices based on the set of mixture coefficients. Specifically, we use the Euclidean

distance of mixture coefficients and a data structure that supports nearest neighbor queries [Arya et al. 98] to answer quickly typical queries as those we have mentioned.

It is worth emphasizing that the vertices of the fitted simplex may reveal structure in the network. Depending on the network, a domain expert can interpret their meaning. In the networks we use in Section 5, the interpretation is straightforward. Because of the special importance of the simplex vertices, we shall refer to them as *social network archetypes*. It is worth noticing that our proposed formulation compared to the k-community literature allows us to mine the graph even when there are no well-shaped clusters, see for instance Figures 1(a), 4 and 5.

4. CondSim: Graph Similarity and the Generalized Condition Number

4.1. Theoretical Result and Algorithms

Let $G([n], E_G), H([n], E_H)$ be connected graphs on n vertices (labeled for simplicity $\{1, 2, .., n\} = [n]$), and L_G, L_H be their Laplacian matrix representation, respectively. Also, let $\Lambda(L_G, L_H)$ and $\kappa(L_G, L_H)$ be the set of generalized eigenvalues and the generalized condition number of the pencil (L_G, L_H) [Golub and Ye 02, Golub and Van Loan 96]. We use S_n to denote the symmetric group, i.e., the group whose elements are all the permutations of the set $[n]$ and whose group operation is the composition of such permutations. We denote with $L_{G^{(\sigma)}}$ where $\sigma \in S_n$ the Laplacian matrix representation of the graph G whose vertex set has been renamed according to σ, i.e., $v \mapsto \sigma(v)$ for all $v \in [n]$. Our main result is the next theorem.

Theorem 4.1. *Let Ω be the state space representing the set of all permutations $\{\sigma : \sigma \in S_n\}$, and $f : \Omega \to \mathbb{R}^+$ be a function defined by $f(\omega) = \kappa(L_G, L_{H^{(\omega)}})$ for all $\omega \in \Omega$. Also, fix $\lambda \geq 1$ and define $\pi_\lambda(\omega) = \frac{\lambda^{-f(x)}}{Z(\lambda)}$ where $Z(\lambda) = \sum_{x \in \Omega} \lambda^{-f(x)}$ is the normalizing constant that makes π_λ a probability measure. We define a Metropolis chain, in which we allow transitions between two states if and only if they differ by a transposition as follows: if $f(\omega_1) < f(\omega_2)$ the Metropolis chain accepts the transition $\omega_1 \to \omega_2$ with probability $\lambda^{f(\omega_1)-f(\omega_2)}$, otherwise, always accept it.*

As $\lambda \to +\infty$ the stationary distribution π_λ of the Metropolis chain converges to the uniform distribution over the global minima of f. Furthermore, if $G \sim H$, i.e., G, H are isomorphic, then π_λ converges to the uniform distribution over the set of isomorphisms $\{\sigma : L_G = L_{H^{(\sigma)}}, \sigma \in S_n\}$.

Proof. Recall that the Laplacian representation of a connected graph is a symmetric, positive semidefinite matrix and that the dimension of its null space is 1 (the all-ones vector $\mathbf{1}$). Consider now the generalized eigenvalue problem $L_G x = \lambda L_H x$. The pencil (L_G, L_H) is Hermitian semidefinite. Therefore, there, exists a basis of generalized eigenvectors [Golub and Van Loan 96]. Notice that the all-ones vector $\mathbf{1}$ is a generalized eigenvector with corresponding generalized eigenvalue 0. Let $\Lambda(L_G, L_H) = \{0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{n-1}$ be the set of generalized eigenvalues. Then, $\kappa(L_G, L_H) = \frac{\lambda_{n-1}}{\lambda_1}$.[1] We prove that $\kappa(L_G, L_H) = 1$ if and only if $G \sim H$.

- $\kappa(L_G, L_H) = 1 \Rightarrow G \sim H$:

The generalized eigenvalues are $\lambda(L_G, L_H) = (0 = \lambda_0 < 1 = \lambda_1 = \ldots = \lambda_{n-1})$. Let $(\mathbf{1} = u_0, u_1, \ldots, u_{n-1})$ be the corresponding generalized eigenvectors which *form a basis*. Define $X = L_G - L_H$. Notice that $X u_i = 0$ for all $i = 0, .., n-1$. Hence, $X = 0$ and therefore $L_G = L_H \rightarrow G \sim H$.

- $G \sim H \Rightarrow \exists \sigma \in S_n$ s.t. $\kappa(L_G, L_{H^{(\sigma)}}) = 1$:

Since $G \sim H$ there exists a permutation $\sigma \in S_n$ such that $L_G = L_{H^{(\sigma)}}$. Simply, by substituting the eigenvectors $\{u_i\}_{i=0,..,n-1}$ of L_G in $L_G x = \lambda L_{H^{(\sigma)}} x = \lambda L_G x$, we obtain that the generalized eigenvalues are $(0, 1, 1, .., 1)$ and the corresponding eigenvectors $(\mathbf{1} = u_0, u_1, \ldots, u_{n-1})$. Hence, $\kappa(L_G, L_{H^{(\sigma)}}) = 1$.

Now, define $\Omega^* = \{\omega \in \Omega : f(\omega) = f^* = \min_{x \in \Omega} f(x)\}$. Because our chain is a Metropolis chain [Levin et al. 08], its stationary distribution is π_λ. Therefore,

$$\begin{aligned} \lim_{\lambda \rightarrow +\infty} \pi_\lambda(\omega) &= \lim_{\lambda \rightarrow +\infty} \frac{\lambda^{f(\omega)}/\lambda^{f^*}}{|\Omega^*| + \sum_{\omega \in \Omega - \Omega^*} \lambda^{f(\omega)}/\lambda^{f^*}} \\ &= \frac{I(\omega \in \Omega^*)}{|\Omega^*|}, \end{aligned} \tag{4.1}$$

where $I(\alpha \in A)$ is an indicator variable equal to 1 if element α belongs to set A, otherwise 0. If G, H are isomorphic, then $f^* = 1$ and, therefore, the above result suggests that the Metropolis chain converges to the uniform distribution over the set $\{\sigma : L_G = L_{H^{(\sigma)}}, \sigma \in S_n\}$. □

It is worth emphasizing that our result is valid even when the two Laplacians are cospectral. For instance, Figure 2 shows two cospectral graphs with respect to their Laplacians. Both Laplacians share the same set of eigenvalues

[1] Notice that despite the fact that our matrices are positive semidefinite, and not positive definite, this doesn't cause any real problem with respect to defining the generalized condition number since L_G, L_H have the same null space.

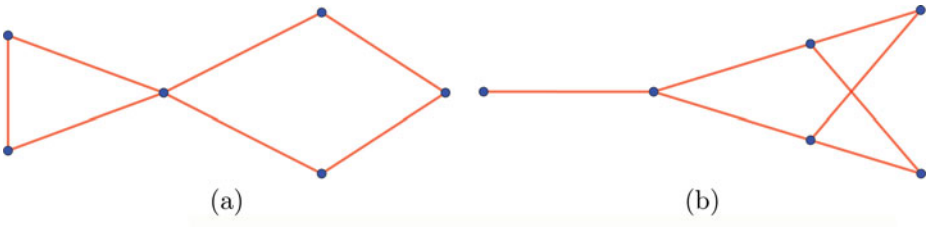


Figure 2. Cospectral Laplacians: The two nonisomorphic graphs have cospectral Laplacian matrix representation. The minimum generalized condition number over the space of 6! permutations is 6.1852.

$\{0, 0.76, 2, 3, 3, 5.24\}$. However, over the space of 6! permutations, the minimum generalized condition number is 6.19.

Our proposed algorithm CondSimGradDescent is shown as Algorithm 2. It is a greedy, efficient heuristic. The algorithm takes two parameters, the maximum number of iterations q and a parameter $\epsilon > 0$ which quantifies the least amount of progress required by the algorithm to keep iterating. This may help to avoid extremely incremental improvements that do not significantly improve the graph matching but cost a great deal computationally. Algorithm 2 performs gradient descent with respect to the generalized condition number using transpositions. On the one hand, algorithm 2 tends to be computationally more aggressive than the Metropolis chain in the sense that it always moves to a state/permutation that results in a smaller generalized condition number. On the other hand, the Metropolis chain, because of the randomization, is likely to avoid local minima. Our algorithm returns the permutation that defines the best graph alignment found and the corresponding condition number.

The complexity of our Algorithm 2 depends on the choice of algorithm that solves the generalized eigenvalue problem. Specifically, let $f(L_G, L_H)$ be the corresponding running time as a function of the two Laplacians. Also let q abbreviate the maximum number of iterations MAXITER. Then the total running time is upper bounded by $O(qn^2 f(L_G, L_H))$ because we perform q steps and at each step we compute the generalized condition number for the $\binom{n}{2}$ possible transpositions. In our experiments we use the algorithm of [Golub and Ye 02]. The speed of convergence is given in Lemma 1 [Golub and Ye 02]. For our purposes, since we set the number of iterations (which are matrix-vector multiplications) of the Golub–Ye algorithm to a constant, we may assume that the running time that computing the smallest nontrivial and the largest generalized eigenvalue of the pencil (L_G, L_H) is linear in the total number of edges $|E_G| + |E_H| = O(m)$, where $m = \max(|E_G|, |E_H|)$. It is worth noticing that, using a series of transpositions, we can reach any permutation from any starting permutation. If m

is large, e.g., $m \gg n \log n$, one can use the developed theory of spectral sparsifiers to speed up the generalized condition number computations. Specifically, one may perform first the Spielman–Srivastava sparsification [Spielman and Srivastava 08] on both Laplacians L_G, L_H, obtain spectrally equivalent matrices $\tilde{L}_G, \tilde{L}_H$, and apply our algorithm on the latter Laplacians.

Algorithm 2. CondSimGradDescent.

Require: L_G, L_H the Laplacian matrix representation of G, H respectively. q (Maximum number of iterations). $\epsilon > 0$ (Tolerance). {σ initialized to the identity permutation}
$\sigma \leftarrow (1, 2, .., n)$
$i \leftarrow 0$
while $i \leq q$ **do**
  $i \leftarrow i + 1$
  $\sigma^* \leftarrow \arg\max_{\sigma' \in S'} \kappa(L_G, L_{H^{(\sigma)}}) - \kappa(L_G, L_{H^{(\sigma')}})$ where S' is the set of all permutations that differ from σ. a single transposition.
  if $\kappa(L_G, L_{H^{(\sigma)}}) - \kappa(L_G, L_{H^{(\sigma')}}) > \epsilon$ **then**
    $\sigma \leftarrow \sigma^*$
    CN$\leftarrow \kappa(L_G, L_{H^{(\sigma)}})$
  else
    break
  end if
  if CN $= 1$ **then**
    break
  end if
end while
Return (σ, CN)

4.2. Further Insight: Theory of Support Trees

We proved in Theorem 4.1 that when the generalized condition number is 1, then indeed G, H can be perfectly matched, i.e., G, H are isomorphic. To complete the justification of our rationale behind the choice of our similarity measure we need to explain why a value close to 1 implies a good graph alignment. The answer lies in the theory of support preconditioners [Gremban 96, Bahendr n.d.]. In the following, let A, B be Laplacian matrices.

Definition 4.2. Support The support $\sigma(A, B)$ of matrix B for A is the greatest lower bound over all τ such that $\tau B - A$ is positive semidefinite, i.e., $\sigma(A, B) = \liminf\{\tau : \tau B - A \succeq 0\}$.

Name (Abbr.)	Nodes (n)	Edges (m)
⊙ Netscience	1589	2742
⊙ Football	115	613
⊙ Political Books	105	441
⋆ Erdös '72	5488	7085
⋆ Erdös '82	5822	7375
⋆ Erdös '02	6927	8472
⋆ Roget Thesaurus	1022	3648

Table 1. Datasets.

Definition 4.3. Congestion & Dilation An embedding of H into G is a mapping of vertices of H onto vertices of G, and edges of H onto paths in G. The dilation $d(G, H)$ of the embedding is the length of the longest path in G onto which an edge of H is mapped. The congestion $g_e(G, H)$ of an edge e in G is the number of paths of the embedding that contain e. The congestion $g(G, H)$ of the embedding is the maximum congestion of the edges in G.

The following facts have been proved in Gremban's PhD thesis [Gremban 96]: (a) The support number $\sigma(A, B)$ is bounded above by the maximum product of dilation and congestion over all embedding of A into B; (b) $\kappa(A, B) \leq \sigma(A, B)\sigma(B, A)$, Lemma 4.8 [Gremban 96]. Fact (b), in combination with fact (a), shows that the generalized condition number is closely related to the goodness of two embeddings, i.e., of H into G and vice versa. When both the dilation and the congestion of the embeddings—or in our terminology of the alignments—are small then the generalized condition number is small.

5. Experiments

In Sections 5.1, 5.2 we describe the datasets we used in our experiments and the experimental setup. In Sections 5.3, 5.4 we provide an experimental evaluation of our proposed methods respectively.

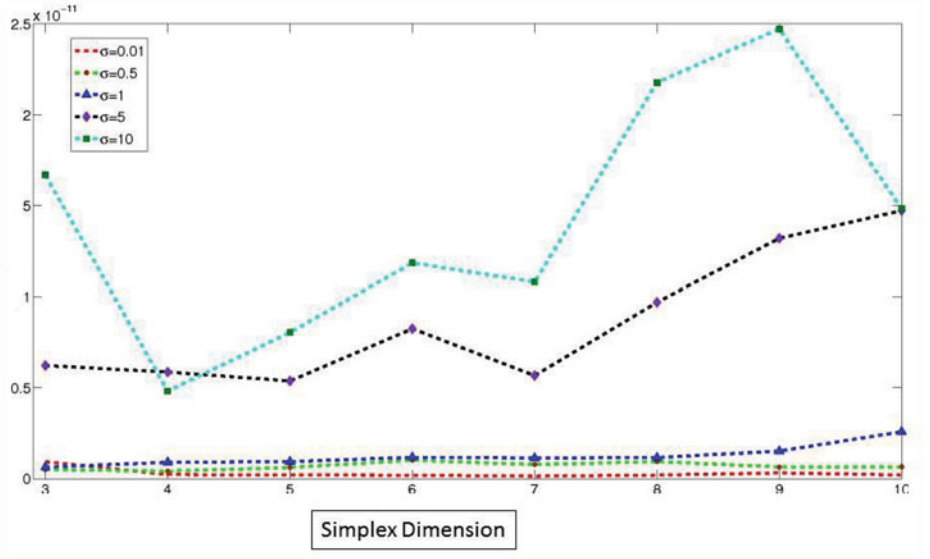


Figure 3. Performance of simplex fitting on 1000 points drawn uniformly at random from a randomly generated k-simplex perturbed by Gaussian noise $N(0, \sigma^2)$. Figure plots the sum of Euclidean distances of the $k+1$ reconstructed simplex vertices from the $k+1$ true vertices as a function of the dimensionality k of the simplex for five different standard deviations $\sigma = 0.01, 0.5, 1, 5, 10$. Notice that the simplex fitting method (essentially) perfectly recovers the true simplex in all cases.

5.1. Datasets

Table 1 summarizes the real-world datasets we used for our experiments. Whenever neccessary, graphs were made undirected, unweighted, and self-edges were removed. Datasets annotated with $\odot$ and $\star$ are available online from [Newman n.d., Pajek n.d.] respectively. We picked small and medium sized networks deliberately because the geometric structure in such networks is striking.

We also generated several synthetic datasets. Specifically, for Section 5.3.1 we generated a cloud of points, where each point was chosen uniformly at random from a random k-simplex (see Appendix [Levin et al. 08]) and a random stratified social network, see Section IIIA of [Leicht et al. 06]. Notice that the first type of synthetic data involves no graph and its goal is to test the goodness of the simplex fitting method. Stratified networks model the phenomenon according to which individuals make connections with individuals similar to themselves with respect to some criterion, e.g., income, age. For each vertex, we picked an age from 1 to 10, chosen uniformly at random. Two vertices with age i and

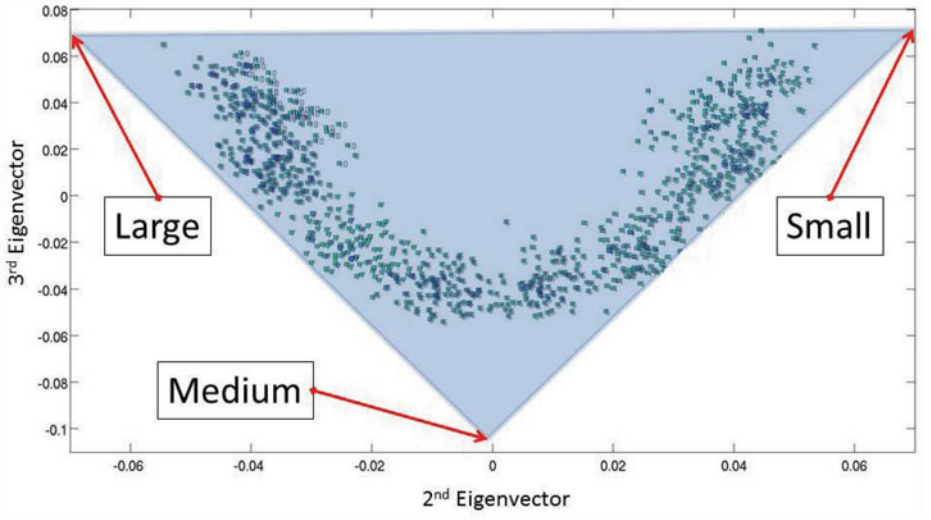


Figure 4. 2-simplex fitted on a random stratified network. VertexSim correctly assigns higher similarity values to vertices of the same age. The three vertices of the fitted 2-simplex conceptually represent the concepts "senior/large age" (8–10), "middle-aged/medium age"(4–7) and "young/small age" (1–3).

j, respectively, are connected with probability $p_0 e^{-\alpha \Delta t}$ where $\Delta t = |i - j|$. The parameters are set to $\alpha = 0.8$ and $p_0 = 0.1$. Finally, for Section 5.4 we generated random graphs of two types. Erdös-Rényi-Gilbert graphs [Bollobás 01] and R-MAT graphs [Chakrabarti et al. 04]. For the former we use $p = 0.5$ and for the latter the parameters are set to $[a = 0.55, b = 0.1; c = 0.1, d = 0.25]$.

5.2. Experimental Setup and Implementation Details

The experiments were performed on a single machine, with Intel Xeon CPU at 2.83 GHz, 6144KB cache size and and 50GB of main memory. Our algorithms are implemented in MATLAB. The results we show are obtained by setting the parameter γ to 1 and the dimensionality k of the embedding equal to 2. Clearly our method is valuable when k is larger than 3 where visualization is impossible (see also Section 2.3 for a discussion of the computational complexity as a function of k). Here we report results for $k = 2$ for visualization purposes. It is worth pointing out two more facts concerning our experimental section: first, VertexSim in our experiments was not affected by the value of the parameter γ because there were no outliers in any of the embeddings, and second, we experimented with higher values of k (from 3 to 5) obtaining interpretable results.

The wall-clock times we report use the Golub–Ye algorithm [Golub and Ye 02] as a subroutine to compute condition numbers. In order to use this algorithm,

which is designed for positive definite pencils, we shift slightly the spectrum of the Laplacians: we set $L' = L + \epsilon \frac{\mathbf{1}\mathbf{1}^T}{n}$ where ϵ is a small positive constant. This is a natural "trick" to compute the generalized condition number. We use the default settings of the *eigifp* software.[2] It is worth mentioning that given that our graphs are small- and medium-sized, we checked the quality of this "trick." We observed that when we set $\epsilon = 0.01$, we obtain essentially accurate condition numbers. For instance, assume we permute the set of vertices of the Football network according to a randomly generated permutation. We compute the generalized condition number using the shifting trick and the Golub–Ye algorithm and exactly by computing the eigenvalues of $(L_b)^{\dagger} L_a$. In the former case we obtain 52.592 and in the latter 52.591. This is a representative example of what we observe in practice, which also explains why this shifting heuristic is frequently used, see Section 6 in [Sun et al. 09].

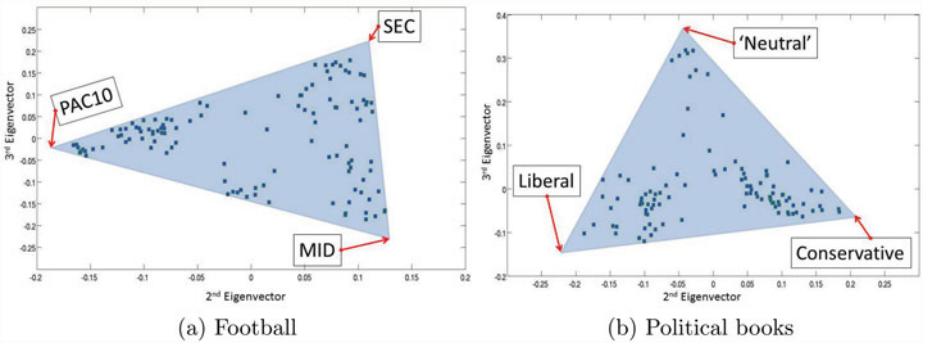
The parameters of CondSimGradDesc were set to $q = 200$, $\epsilon = 0$ for all experiments in Section 5.4.

A final remark with respect to the experiments of CondSim in Section 5.4: it is a well-known fact that a permutation can be decomposed in cycles and that a random permutation has $O(\log n)$ cycles in expectation [Wilf 99]. Therefore, if we generate only permutations chosen uniformly at random, we are restricting ourselves, with high probability, to permutations that share a common structure. To avoid a potential artifact in our experimental results, we generate permutations with a different number of cycles. We use a simple recursive algorithm [Wilf 99] to generate a permutation with k cycles uniformly at random in our experiments [see p. 33, Wilf 99]. Finally, we use third-party software and specifically the code of [Arya et al. 98, Bayati et al. 09] and Jeremy Kepner's R-MAT code implementation.

5.3. VertexSim at Work

5.3.1. Synthetic Data. We validate the VertexSim algorithm in two ways: first we verify that it can successfully recover the simplex $\mathcal{S}$ and, hence, the mixture coefficients of data points sampled uniformly at random from $\mathcal{S}$; and second, we evaluate its performance on a stratified network. Figure 3 shows the performance of our fitting method as a function of the simplex dimension (x-axis) for five different standard deviations $\sigma = 0.01, 0.5, 1, 5, 10$ (5 lines) for a randomly generated k-simplex. The quality of the performance (y-axis) is quantified as the sum $\sum_{i=1}^{k+1} ||v_i - \tilde{v}_i||$, where $\tilde{v}_i$ is the reconstructed vertex of the k-simplex. The performance is excellent, as Figure 3 shows. The average running time for four

[2] Available online at http://www.ms.uky.edu/ qye/software.html



(a) Football (b) Political books

Figure 5. Minimum-area 2-simplexes for the (a) Football network (b) Political books network. In both cases, VertexSim provides significant mining capabilities for extracting pairs of highly similar vertices and concepts. For more details, see Section 5.3.2.

executions is 0.0071 and the variance 1.4×10^{-6}. It is worth mentioning that we also tried the Chan et al. algorithm [Chan et al. 09] obtaining exactly the same simplex.

Figure 4 shows the performance of VertexSim for a stratified network with $\alpha = 0.8$ and $p_0 = 0.1$ and ages ranging from 1 to 10, picked uniformly at random for every vertex. Specifically, Figure. 4 shows the fitted 2-simplex. Upon performing Step 3 of Algorithm 1, it becomes apparent that pairs of vertices with the same age are significantly more similar than vertices with different ages, which is what a good vertex similarity algorithm should have as its output. Furthermore, the three vertices of the 2-simplex correspond to the three concepts, "senior/large age," "middle-aged/medium age," and "young/small age."

5.3.2. Real-world Data. Figure 5(a) shows the minimum-area-fitted 2-simplex for the American football college network, whose vertices correspond to teams and whose edges correspond to games among them. According to [Girvan and Newman 02] the teams are divided into conferences containing around 8–12 teams each and the frequency of games between members of the same conference is higher than between members of different conferences. The three vertices of the fitted simplex correspond to three conferences PAC 10, SEC, and MID. Furthermore, VertexSim, using the fitted mixture coefficients, assigns higher similarity to vertices of the same conference. The fitting algorithm took 7.5 seconds to find the simplex. Similar remarks hold for Figure 5(b) which shows the minimum-area-fitted 2-simplex for the political books network whose vertices represent books-and-edges copurchasing by the same buyer. The three vertices of the simplex correspond to liberal, conservative, and "neutral" books. For both datasets,

	Erdös-Renyi			R-MAT		
$\|V\|$	8	16	32	8	16	32
CondSimGradDescent	6/7	5/15	7/31	5/7	5/15	11/31
Belief Propagation (BP) [Bayati et al. 09]	0/7	0/15	0/31	0/7	0/15	0/31

Table 2. Results of our method versus the belief propagation method of [Bayati et al. 09] on various random networks for permutations whose number of cycles ranges from 1 to $|V| - 1$. The fractions indicate how many times an algorithm found a permutation makes the original graph and its permuted version exactly the same.

the vectors of mixture coefficients provide us a novel way to determine vertex similarities in an interpretable way. The fitting algorithm needs 6 seconds to compute the simplex. We obtain highly interpretable results for other datasets as well. Indicatively we report a few highly similar pairs of vertices according to VertexSim: (musician, poetry), (melody, poetry), (voice, hearing) from the Rogets Thesaurus network and (Vojtech Rödl, Noga Alon), (Joel Spencer, Janòs Pach) from the Erdös collaboration network, 1972.

5.4. CondSimGradDescent at Work

5.4.1. Synthetic Data. We compare CondSimGradDescent with the belief propagation method of [Bayati et al. 09] for a few synthetic datasets in the following way. We generate a random graph of n vertices and a permutation with k cycles, where k ranges from 1 to $n - 1$. We do not consider the identity permutation with n cycles. We permute the graph according to the random permutation and see whether the graph matching methods can align perfectly the original graph and its permuted version. We use two types of random graphs, namely binomial random graphs [Bollobás 01] and R-MAT graphs with 8, 16, 32 vertices. Table 2 shows the results. As we see, CondSimGradDescent outperforms significantly BP [Bayati et al. 09]. It is worth pointing out again that this is a validation test and that fast graph isomorphism tests exist [Bollobás 01]. An interesting trend we observe is that the fewer the cycles of the permutation, the easier CondSimGradDescent gets trapped in local minima. On the positive side, when the number of cycles is small, CondSimGradDescent typically finds an optimal alignment efficiently.

5.4.2. CondSimGradDescent as a Post-Processing Tool. Due to the computational cost of CondSimGradDescent, a realistic use of it in large networks is as a postprocessing tool. We describe a typical use of CondSimGradDescent as a postprocessing tool that significantly improves the graph alignment in combination with the beliefpropagation-based method of [Bayati et al. 09]. We perform the following experiment: we consider the *Football* network. We generate a permutation σ uniformly at random and permute the labels of G accordingly. The number of fixed points of the permutation we obtain is 0. We apply the function *netalignbp()*, which is open-sourced [Bayati et al. 09]. The alignment produced by [Bayati et al. 09] has recognized correctly 50 out of the 115 correct vertices to vertex assignments. The generalized condition number equals 29.4. Applying the CondSimGradDesc method to the alignment obtained from belief propagation, we obtain a generalized condition number of value 4.16, resulting in 74 correct assignments. Each iteration of CondSimGradDesc lasts on average, 90 seconds with standard deviation equal to 2 seconds over the 200 iterations. It is worth outlining that the belief-propagation-based method of [Bayati et al. 09] is designed for the setting where there exists a reasonable guess for the graph alignment. One should conclude only that our proposed method is useful as a postprocessing tool, and not draw any negative conclusions on the performance of [Bayati et al. 09].

6. Conclusion

6.1. Summary

In this work we contribute to the important problem of quantifying vertex similarity in networks by introducing novel approaches to the problems of vertex similarity within and across two graphs with the same number of vertices, respectively. We observed an excellent performance of our algorithms both on synthetic and on real-world networks. This verifies empirically that both the proposed conceptual approaches as well as the algorithmic solutions are valuable.

6.2. Discussion and Open Problems

Concerning our first algorithm, VertexSim, two natural questions arise: Is there always geometric structure in social networks? Can we fit other geometric objects such as simplicial complexes to capture more complex geometric structure? Concerning the first question, [Leskovec et al. 08] have studied extensively properties of large scale networks and it appears that there exists strong geometric

structure in small- and medium-sized networks such as those we studied in Section 5, but the structure typically decays as the size of the network grows. The answer to the second question is an interesting research problem.

Concerning our second algorithm, CondSimGradDescent, an interesting research direction is to extend it to cases where the two graphs have a different number of vertices. The theory of Steiner tree preconditioners [Koutis 07] is a promising approach. Also, understanding the performance of CondSimGradDescent in the isomorphism setting is another interesting problem. Finally, understanding its performance in simple graphs, e.g., trees, remains open.

Acknowledgments. I would like to thank Ioannis Koutis, Gary Miller for several discussions on the theory of support tree preconditioners and Jure Leskovec for discussions concerning the geometry of social networks. Also, I would like to thank the reviewers for their thorough comments.

Funding. Research supported by NSF Grant No. CCF-1013110.

References

[Adamic and Adar 03] L. Adamic and E. Adar. "Friends and Neighbors on the Web." *Social Networks* 25:3(2003), 211–230.

[Arora et al. 12] S. Arora, R. Ge, R. Kannan, and A. Moitra. "Computing a Nonnegative Matrix Factorization – Provably." In *Proceedings of the 44th Symposium on Theory of Computing (STOC '12)*, pp. 145–162. New York, NY: ACM, 2012.

[Arya et al. 98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. "An Optimal Algorithm for Approximate Nearest Neighbor Searching." *Journal of the ACM* 45:6 (1998), 891–923.

[Bahendr n.d.] Bahendr. http://www.sandia.gov/_bahendr/support.html.

[Bai et al. 04] X. Bai, H. Yu, and E. R. Hancock. "Graph Matching Using Spectral Embedding and Alignment." In *Proceedings of the 17th International Conference on Pattern Recognition*, vol 3, pp. 398–401. Cambridge: IEEE, 2004.

[Bayati et al. 09] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang. "Algorithms for Large, Sparse Network Alignment Problems." In *Proceedings of the IEEE International Conference on Data Mining*, pp. 705–710. Washington, D.C.: IEEE, 2009.

[Belkin and Niyohi 03] M. Belkin and P. Niyohi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation* 15:6 (2003), 1373–1396.

[Blondel et al. 03] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. A measure of similarity between graph vertices. *SIAM Review* 46:4 (2003), 647–666.

[Bollobás 01] Bollobás, B. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge, UK: Cambridge University Press, 2001.

[Bronstein et al. 10] A. Bronstein, M. Bronstein, U. Castellani, A. Dubrovina, L. Guibas, et al.: "SHREC 2010: Robust Correspondence Benchmark." Paper presented at Eurographics Workshop on 3D Object Retrieval, Norrkoping, Sweden, May 2, 2010.

[Burkard et al. 09] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM, 2009.

[Chakrabarti et al. 04] D. Chakrabarti, Y. Zhan, and C. Faloutsos. "R-MAT: A Recursive Model for Graph Mining." Paper presented at SIAM Data Mining, Orlando, FL, April 22–24, 2004.

[Chan et al. 09] T.-H. Chan, C.-Y. Chi, Y. M. Huang, and K. Ma. "Convex Analysis Based Minimum-Volume Enclosing Simplex Algorithm for Hyperspectral Unmixing." In *Acoustics, Speech and Signal Processing, (ICASSP 2009)*, pp. 1089–1092: IEEE, 2009.

[Craig 94] M. Craig. "Minimum-Volume Transforms for Remotely Sensed Data." *IEEE Transactions on Geoscience and Remote Sensing* 3:3 (1994), 542–552.

[Cutler 10] A. Cutler. "Remembering Leo Breiman." *Annals of Applied Statistics* 4:4 (2010), 1621–1633.

[Cutler and Breiman 94] A. Cutler, L. Breiman. "Archetypal Analysis." *Technometrics* 36:4 (1994), 338–347.

[Fouss et al. 07] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. "Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation." *IEEE Trans. Knowl. Data Eng.* 19:3 (2007), 355–369.

[Girvan and Newman 02] M. Girvan, M. E. J. Newman. "Community Structure in Social and Biological Networks." In *Proceedings of the National Academy of Sciences (PNAS)* 99:12 (2002), 7821–7826

[Gold and Rangarajan 96] S. Gold, and A. Rangarajan. "A Graduated Assignment Algorithm for Graph Matching." IEEE Trans. Pattern Anal. Mach. Intell. 18:4 (1996), 377–388.

[Golub and Ye 02] G. Golub and Q. Ye. "An Inverse Free Preconditioned Krylov Subspace Method for Symmetric Generalized Eigenvalue Problems." *SIAM J. on Scientific Computing* 24:1 (2002), 312–334.

[Golub and Van Loan 96] G. Golub, and C. F. Van Loan. *Matrix Computations*. Baltimore, MD: JHU Press, 1996.

[Gremban 96] K. Gremban. "Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems." PhD Thesis, CMU, 1996.

[Hay et al. 08] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P Weis. "Resisting Structural Re-Identification in Anonymized Social Networks." *PVLDB* 1:1 (2008), 102–114.

[Henderson et al. 12] K. Henderson, B. Gallagher, and T. Eliassi-Rad, et al. "RolX: Structural Role Extraction & Mining in Large Graphs." In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 1231–1239. New York, NY: ACM, 2012.

[Huggins et al. 07] P. Huggins, L. Pachter, and B. Strumfels. "Toward the Human Genotope." *Bulletin of Math. Biology* 69:8 (2007), 2723–2725.

[Jeh and Widom 02] G. Jeh and J. Widom. "SimRank: A Measure of Structural-Context Similarity." In *ACM SIGKDD*, pp. 538–543. ACM Press, 2002.

[Keshava 03] N. Keshava. "A Survey of Spectral Unmixing Algorithms." *Lincoln Laboratory Journal* 14:1 (2003), 55–78.

[Klau 09] G. Klau. "A New Graph-Based Method for Pairwise Global Network Alignment." *BMC Bioinformatics* 10:Suppl 1(2009), S59.

[Kleinberg 99] J. Kleinberg. "Authoritative Sources in a Hyperlinked Environment." *J. of the ACM* 46:5 (1999), 614–632.

[Koutis 07] I. Koutis. "Combinatorial and Algebraic Tools for Multigrid Algorithms." PhD Thesis, CMU, 2007.

[Lee and Verleysen 07] Lee, J.A., Verleysen, M: *Nonlinear Dimensionality Reduction*. Information Science and Statistics. New York, NY: Springer-Verlag, 2007.

[Leicht et al. 06] E. A. Leicht, P. Holme, and M. E. J. Newman. "Vertex Similarity in Networks." *Phys. Rev.* E 73 (2006), 026120.

[Leskovec et al. 08] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. "Statistical Properties of Community Structure in Large Social and Information Networks." In *Proceedings of the 17th International Conference on the WWW*, pp. 695–704. New York, NY: ACM, 2008.

[Levin et al. 08] D. Levin, and Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. Providence, RI: American Mathematical Society, 2008.

[Liben-Nowell and Kleinberg 03] D. Liben-Nowell, and J. Kleinberg. "The Link Prediction Problem for Social Networks." In *Proceedings of the 12th International Conference on Information and Knowledge management,* pp. 556–559. New York, NY: ACM, 2003.

[Melnik et al. 02] S. Melnik, H. Garcia-Molina, and E. Rahm. "Similarity Flooding: A Versatile Graph Matching Algorithm." In *Proceedings of the 18th International Conference on Data Engineering,* p. 17. Washington, D.C.: IEEE Computer Society, 2002.

[Newman n.d.] Newman. http://www.cise.ufl.edu/research/sparse/matrices/Newman/index.html

[O'Rourke et al. 86] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. "An Optimal Algorithm for Finding Minimal Enclosing Triangles." *J. Algorithms* 7:2 (1986), 258–269.

[Packer 02] A. Packer. "NP-Hardness of Largest Contained and Smallest Containing Simplices for V - and H-Polytopes." *Discrete Comput. Geom* 28, (2002), 349–377.

[Pajek n.d.] Pajek. http://www.cise.ufl.edu/research/sparse/matrices/Pajek/index.html

[Pattern Recognition Letters 03] *Pattern Recognition Letters - Special Issue: Graph-Based Representations in Pattern Recognition* 24(8), May 2003. New York, NY: Elsevier, 2003.

[Riedesel 03] P. Riedesel. "Archetypal Analysis in Marketing Research: A New Way of Understanding Consumer Heterogeneity." Action Marketing Research, 2003.

[Schellewald and Schnörr 05] C. Schellewald, and C. Schnörr. "Probabilistic Subgraph Matching Based on Convex Relaxation." In *Energy Minimization Methods in Computer Vision and Pattern Recognition,* pp. 171–186. Lecture Notes in Computer Science 3757. Berlin, Heidelberg: Springer, 2005.

[Shapiro and Brady 92] L. S. Shapiro and M. J. Brady. "Feature-Based Correspondence: An Eigenvector Approach." *Image and Vision Computing* 10:5 (1992), 283–288.

[Shi and Malik 00] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation." *IEEE Pattern Analysis and Machine Intelligence* 22:8 (2000), 888–905.

[Smalter et al. 08] A. Smalter, J. Huan, and G. Lushington. "CPM: A graph pattern matching kernel with diffusion for accurate graph classification." Technical Report, ITTC-FY2009-TR-45910-01, 2008.

[Spielman and Srivastava 08] D. Spielman and N. Srivastava. "Graph Sparsification by Effective Resistances." In *Proceedings of the 40th Symposium on Theory of Computing (STOC '08)*, pp. 563–568. New York, NY: ACM, 2008.

[Sun et al. 09] L. Sun, S. Ji, and J. Ye. "A Least Squares Formulation for a Class of Generalized Eigenvalue Problems in Machine Learning." In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09*), pp. 977–984, New York, NY: ACM, 2009.

[Tolliver et al. 10] D. Tolliver, C. E. Tsourakakis, A. Subramanian, S. Shackney, and R. Schwartz. "Robust Unmixing of Tumor States in Array Comparative Genomic Hybridization Data." *Bioinformatics ISMB* 26:12 (2010), 106–114.

[Umeyama 98] S. Umeyama. An Eigendecomposition Approach to Weighted Graph Matchings Problems. *Transactions of Pattern Analysis and Machine Intelligence* 10:5 (1998), 695–703.

[Wilf 99] H. S. Wilf. *East Side, West Side...* Lecture Notes. Philadelphia, PA: University of Pennsylvania, 1999.

[Zager 05] L. Zager. "Graph Similarity and Matching." Master's thesis, Massachusetts Institute of Technology, 2005.

[Zhou and Suri 00] Y. Zhou and S. Suri. "Algorithms for Minimum Volume Enclosing Simplex in R3." In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, pp. 500–509. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000.

Charalampos E. Tsourakakis, Department of Mathematical Sciences, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA (ctsourak@math.cmu.edu)