Internet Mathematics Vol. 8, No. 4: 456–481

Approximations of the Generalized Inverse of the Graph Laplacian Matrix

Enrico Bozzo and Massimo Franceschet

Abstract. We devise methods for finding approximations of the generalized inverse of the graph Laplacian matrix, which arises in many graph-theoretic applications. Finding this matrix in its entirety involves solving a matrix inversion problem, which is resource-demanding in terms of consumed time and memory and hence impractical whenever the graph is relatively large. Our approximations use only a few eigenpairs of the Laplacian matrix and are parametric with respect to this number, so that the user can compromise between effectiveness and efficiency of the approximate solution. We apply the devised approximations to the problem of computing current-flow betweenness centrality on a graph. However, given the generality of the Laplacian matrix, many other applications can be sought. We experimentally demonstrate that the approximations are effective already with a constant number of eigenpairs. These few eigenpairs can be stored with a linear amount of memory in the number of nodes of the graph, and in the realistic case of sparse networks, they can be efficiently computed using one of the many methods for retrieving a few eigenpairs of sparse matrices that abound in the literature.

1. Introduction

The graph Laplacian is an important matrix that can tell us much about graph structure. It places on the diagonal the degrees of the graph nodes and

ISSN: 1542-7951 print

elsewhere information about the distribution of edges among nodes in the graph. The graph Laplacian matrix, as well as its Moore–Penrose generalized inverse [Ben-Israel and Greville 03], are useful tools in network science that turn up in many different places, including random walks on networks, resistor networks, resistance distance among nodes, node centrality measures, graph partitioning, and network connectivity [Ghosh et al. 08, Newman 10]. In particular, among measures of centrality of graph nodes, betweenness quantifies the extent to which a node lies between other nodes. Nodes with high betweenness are crucial actors of the network, since they exert control over information (or over whatever else flows on the network) passing between others. Moreover, removal from the network of these brokers might seriously disrupt communications between other vertices [Newman 05, Brandes and Fleischer 05].

The computation of the generalized inverse of the Laplacian matrix is demanding in terms of consumed time and space, and thus it is infeasible on relatively large networks. On the other hand, there are today large databases from which real networks can be constructed, including technological, information, social, and biological networks [Brandes and Erlebach 05, Newman et al. 06, Newman 10]. These networks are voluminous and grow in time as more data are acquired. We have, therefore, the problem of running computationally heavy algorithms over large networks. The solution investigated in the present work is the use of approximation methods: algorithms that compute a solution close to the exact one and, as a compromise, that run using many fewer resources than the exact algorithm.

We propose a couple of approximation methods to compute the generalized inverse of the Laplacian matrix of a graph. Both methods are based on the computation of a few eigenpairs (eigenvalues and the corresponding eigenvectors) of the Laplacian matrix [Golub and Meurant 10], where the number of computed eigenpairs is a parameter of the algorithm. The first method, called cutoff approximation, uses the computed eigenpairs in a suitable way for the approximation of the actual entries of the generalized inverse matrix. The second method, named stretch approximation, takes advantage of the computed eigenpairs as well as of an estimation of the excluded ones. Both approximation methods can be applied to estimate current-flow betweenness centrality scores for the nodes of a graph. We experimentally show, using both random and scale-free network models, that the proposed approximations are both effective and efficient compared to the exact methods. In particular, the stretch method allows us to estimate, using a feasible amount of time and memory, a ranking of current-flow betweenness scores that strongly correlates with the exact ranking.

The layout of the paper is as follows. Section 2 introduces the notions of Laplacian matrix and its Moore–Penrose generalized inverse and recalls some

basic properties of these matrices. In Section 3, we define cutoff and stretch approximations. Moreover, we theoretically show that stretch approximation is more effective than cutoff approximation. We review in Section 4 the methods for inverting a matrix and for finding a few eigenpairs of a matrix, which are crucial operations in our contribution. Current-flow betweenness centrality is illustrated in Section 5. We formulate the definition in terms of the generalized inverse of the Laplacian matrix, which allows us to use cutoff and stretch approximations to estimate betweenness scores. A broad experimental analysis is proposed in Section 6 in order to investigate the effectiveness and efficiency of the devised approximation methods.

2. The Graph Laplacian and Its Generalized Inverse

Let $\mathcal{G} = (V, E, w)$ be an undirected weighted graph with V the set of nodes, E the set of edges, and w a vector such that $w_i > 0$ is the positive weight of edge i, for $i = 1, \ldots, |E|$. We denote by n the number of nodes and m the number of edges of the graph. The weighted Laplacian of $\mathcal{G}$ is the symmetric matrix

$$G = D - A,$$

where A is the weighted adjacency matrix of the graph and D is the diagonal matrix of the generalized degrees (the sum of the weights of the incident arcs) of the nodes.

In order to obtain more insight into the properties of the graph Laplacian, it is useful to express the matrix in another form. Let $B \in \mathbb{R}^{n \times m}$ be the incidence matrix of the graph such that if edge l connects two arbitrarily ordered nodes i and j, then $B_{i,l} = 1$, $B_{j,l} = -1$, while $B_{k,l} = 0$ for $k \neq i, j$. Given a vector v, the square diagonal matrix whose diagonal entries are the elements of v is denoted by $\mathrm{Diag}(v)$. Then we have $G = B\,\mathrm{Diag}(w)B^T$. Thus in addition to being symmetric, G is positive semidefinite, so that it has real and nonnegative eigenvalues that are useful to order as $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. If e denotes a vector of ones, then $De = Ae$, so that $Ge = 0$. It follows that $\lambda_1 = 0$ is the smallest eigenvalue of G. We will assume throughout the paper that $\mathcal{G}$ is connected. In this case, all other eigenvalues of G are strictly positive [Ghosh et al. 08]:

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n.$$

Since $\lambda_1 = 0$, the determinant of G is zero, and hence G cannot be inverted. As a substitute for the inverse of G we use the Moore–Penrose generalized inverse of G, which we call simply the generalized inverse of G [Ben-Israel and Greville 03]. As is customary, we denote this kind of generalized inverse by G^+. It is convenient

to define G^+ starting from the spectral decomposition of G. In fact, since G is symmetric, it admits the spectral decomposition

$$G = V\Lambda V^T,$$

where $\Lambda = \mathrm{Diag}(0, \lambda_2, \ldots, \lambda_n)$ and the columns of V are the eigenvectors of G. Observe that V is an orthogonal matrix, that is, $VV^T = I = V^T V$.

Using the spectral decomposition of G, its generalized inverse can be defined as follows:

$$G^+ = V\,\mathrm{Diag}\left(0, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}\right) V^T = \sum_{j=2}^{n} \frac{1}{\lambda_j} V(:,j)V(:,j)^T, \tag{2.1}$$

where $V(:,j)$ (respectively $V(j,:)$) denotes the jth column (respectively row) of matrix V.

Observe that G^+ inherits from G the property of being symmetric and positive semidefinite. Moreover, G^+ shares the same null space of G, as is true in general for the Moore–Penrose generalized inverse of a symmetric matrix. Thus, since $Ge = 0$, it turns out that $G^+ e = 0$. By setting $J = ee^T$, it follows that $GJ = JG = G^+ J = JG^+ = O$, where O is a matrix of all zeros. Using the eigendecompositions of G and G^+, it is easy to show that

$$\left(G + \frac{1}{n}J\right)\left(G^+ + \frac{1}{n}J\right) = I.$$

It follows that

$$G^+ = \left(G + \frac{ee^T}{n}\right)^{-1} - \frac{ee^T}{n}, \tag{2.2}$$

a formula that can be found in [Ghosh et al. 08] and is used implicitly in [Brandes and Fleischer 05]. Another useful consequence of the above equalities is

$$GG^+ = G^+ G = I - \frac{1}{n}J.$$

The generalized inverse of the graph Laplacian G is useful in solving linear systems of the form $Gv = b$ for some known vector b, which arises in many applications. The range of a matrix G is the linear space of vectors b for which the system $Gv = b$ has a solution. Since G is symmetric, its range is the space orthogonal to its null space. The null space of G is one-dimensional and spanned by the vector e with all components equal to unity. Hence, the range of G comprises the vectors x such that $e^T x = 0$, or equivalently, that sum to zero. It follows that the linear system $Gv = b$ has solutions if b sums to zero. By linearity, the difference of two solutions belongs to the null space of G, and this implies

that if we are able to find an arbitrary solution v^*, then all other solutions are of the form $v^* + \alpha e$, $\alpha \in \mathbb{R}$.

As is well known, $v^* = G^+ b$ is the minimum Euclidean norm solution of the system $Gv = b$, that is, it is the element having minimum Euclidean norm in the affine space of the solutions [Ben-Israel and Greville 03]. For completeness, we observe that regardless of whether b belongs to the range of G, $v^* = G^+ b$ is the minimum Euclidean norm solution of the problem $\min_x \|Gx - b\|_2$.

3. Approximations of the Generalized Inverse

In this section we propose two approximations of the the generalized inverse G^+ of the graph Laplacian matrix G. For $k = 2, \ldots, n$, we define the kth cutoff approximation of G^+ as

$$T^{(k)} = \sum_{j=2}^{k} \frac{1}{\lambda_j} V(:, j) V(:, j)^T. \tag{3.1}$$

In all computations, we do not directly utilize the matrix $T^{(k)}$, but we represent it by the $k-1$ eigenpairs that define it. This representation of $T^{(k)}$ can be stored using $O(kn)$ space, that is, $O(n)$ if k is constant. Moreover, computing an entry of $T^{(k)}$ using its eigenpair representation costs $O(k)$, that is, $O(1)$ if k is constant.

As k increases, the matrices $T^{(k)}$ are more and more accurate approximations of G^+. In fact, $G^+ = T^{(n)}$, and for $k < n$,

$$G^+ = T^{(k)} + \sum_{j=k+1}^{n} \frac{1}{\lambda_j} V(:, j) V(:, j)^T. \tag{3.2}$$

We have $\|G^+ - T^{(k)}\|_2 \leq \|G^+ - M\|_2$ for every $M \in \mathbb{R}^{n \times n}$ having rank less than or equal to $k-1$ [Golub and Van Loan 96]. Moreover, for $k = 2, \ldots, n-1$, the relative 2-norm error of the kth cutoff approximation is

$$\frac{\|G^+ - T^{(k)}\|_2}{\|G^+\|_2} = \frac{1/\lambda_{k+1}}{1/\lambda_2} = \frac{\lambda_2}{\lambda_{k+1}}.$$

The second approximation of G^+ exploits the following observation. If many of the excluded eigenvalues λ_j, for j larger than k, are close to one another, we might approximate them with a suitable value σ. We define the kth stretch approximation of G^+ as

$$S^{(k)} = T^{(k)} + \sum_{j=k+1}^{n} \frac{1}{\sigma} V(:, j) V(:, j)^T. \tag{3.3}$$

It is worth observing that the use of $S^{(k)}$ does not involve any significant additional cost with respect to the use of $T^{(k)}$. Indeed, since

$$\sum_{j=1}^{n} V(:,j)V(:,j)^T = VV^T = I,$$

it follows that

$$\begin{aligned} S^{(k)} &= T^{(k)} - \sum_{j=1}^{k} \frac{1}{\sigma} V(:,j)V(:,j)^T + \sum_{j=1}^{n} \frac{1}{\sigma} V(:,j)V(:,j)^T \\ &= \frac{1}{\sigma} I - \frac{1}{\sigma} V(:,1)V(:,1)^T + \sum_{j=2}^{k} \left(\frac{1}{\lambda_j} - \frac{1}{\sigma} \right) V(:,j)V(:,j)^T. \end{aligned}$$

Observe that the normalization of the eigenvector of G associated with the eigenvalue $\lambda_1 = 0$ yields $V(:,1) = e/\sqrt{n}$, where e is a vector with all components equal to unity. It follows that if one knows the value σ, the kth stretch approximation $S^{(k)}$ can be represented using $k-1$ eigenpairs; hence the space needed to store the representation of $S^{(k)}$ and the time needed to compute its entries do not increase with respect to the use of $T^{(k)}$.

On the other hand, the use of $S^{(k)}$ instead of $T^{(k)}$ allows us to improve the bound on the approximation error. Indeed, since

$$G^+ - S^{(k)} = G^+ - T^{(k)} - \sum_{j=k+1}^{n} \frac{1}{\sigma} V(:,j)V(:,j)^T,$$

from (3.2) we obtain

$$\|G^+ - S^{(k)}\|_2 = \left\| \sum_{j=k+1}^{n} \left(\frac{1}{\lambda_j} - \frac{1}{\sigma} \right) V(:,j)V(:,j)^T \right\|_2 = \max_{j=k+1,\ldots,n} \left| \frac{1}{\lambda_j} - \frac{1}{\sigma} \right|.$$

Assuming, as is reasonable, that $\lambda_{k+1} \leq \sigma \leq \lambda_n$, we have that

$$\frac{1}{\lambda_n} - \frac{1}{\lambda_j} \leq \frac{1}{\sigma} - \frac{1}{\lambda_j} \leq \frac{1}{\lambda_{k+1}} - \frac{1}{\lambda_j},$$

and hence

$$\max_{j=k+1,\ldots,n} \left| \frac{1}{\lambda_j} - \frac{1}{\sigma} \right| \leq \frac{1}{\lambda_{k+1}} - \frac{1}{\lambda_n} = \gamma,$$

so that

$$\frac{\|G^+ - S^{(k)}\|_2}{\|G^+\|_2} \leq \lambda_2 \gamma < \frac{\lambda_2}{\lambda_{k+1}} = \frac{\|G^+ - T^{(k)}\|_2}{\|G^+\|_2}.$$

Therefore, the relative 2-norm error of the stretch approximation $S^{(k)}$ is strictly less than the relative 2-norm error of the cutoff approximation $T^{(k)}$,

as soon as we choose σ within λ_{k+1} and λ_n. Moreover, the closer λ_{k+1} and λ_n are to each other, the better the stretch approximation.

The optimal choice for σ, that is, the value that minimizes the 2-norm relative error, is the harmonic mean of λ_{k+1} and λ_n:

$$\frac{1}{\sigma} = \frac{1}{2}\left(\frac{1}{\lambda_{k+1}} + \frac{1}{\lambda_n}\right).$$

With this choice, we reduce by one-half the bound of the approximation error:

$$\max_{j=k+1,\ldots,n} \left|\frac{1}{\lambda_j} - \frac{1}{\sigma}\right| \leq \frac{\gamma}{2}.$$

We have used this choice of σ in all our experiments. Note that the computation of σ implies computing two additional eigenvalues, namely λ_{k+1} and λ_n, but not the corresponding eigenvectors. To avoid this additional cost, we might reasonably assume that λ_n is big, so that its reciprocal is small, and that λ_{k+1} is close to λ_k, so that the optimal value of σ is approximately $2\lambda_k$.

4. Methods for Matrix Inversion and for Finding a Few Eigenpairs

Finding the generalized inverse of the graph Laplacian matrix involves solving a matrix inversion problem (2.2). Inverting a matrix is, however, computationally demanding in terms of time and memory. Given a matrix A, the columns of A^{-1} can be computed by solving the linear systems $Ax = e_i$ for $i = 1, \ldots, n$, where e_i is the vector whose ith entry is equal to one and the other entries are equal to zero. If a direct method is used, then A is factored, and the factorization is used to solve the systems. This typically costs $O(n^3)$ floating-point operations and $O(n^2)$ memory locations (the inverse of a matrix is almost invariably dense even if the input matrix is sparse) [Aho et al. 74]. In particular, the complexity of matrix product and matrix inversion are the same, and the best known lower bound for matrix product, obtained for bounded arithmetic circuits, is $\Omega(n^2 \log n)$ [Raz 03]. If an iterative method is used, then in the case that A has a conditioning independent from the dimension, or a good preconditioner can be found, the number of iterations becomes independent of the dimension. Since every iteration costs $O(m)$, the cost is $O(mn)$ floating-point operations and $O(n^2)$ memory locations to store the inverse. If the matrix is sparse, then the number of operations is quadratic. Otherwise, the number of operations has to be multiplied by an additional factor that depends on the conditioning of A. For an introduction to iterative methods to solve linear systems and to preconditioning, see [Saad 03].

Instead of computing the entire generalized inverse of the Laplacian matrix, our approximation methods compute and store only a few eigenpairs of the Laplacian matrix. If a matrix A is big and sparse, then the computation of a few eigenpairs of A can be made by means of iterative methods whose basic building block is the product of A by a vector, which has linear complexity if A is sparse. One of the simplest among these methods is orthogonal iteration [Golub and Van Loan 96], a generalization of the power method. The method, while simple, can be quite slow, since the number of iterations depends on the distance between the sought eigenvalues of A, and experimental evidence shows that the eigenvalues nearest to zero are clustered, in particular for sparse networks [Zhan et al. 10].

On the other hand, one of the most widely used algorithms is the Lanczos method with implicit restart, implemented by ARPACK [Lehoucq et al. 98]. This is the method we have used in our experiments. For the computation of a few smallest eigenpairs of a matrix A, the method works in the so-called shift and invert mode. In other words, the Lanczos method is applied to $(A - \sigma I)^{-1}$, where σ is a suitable shift. To do this, the matrix $A - \sigma I$ is factored before the iteration begins, and the factorization is used to solve the sequence of linear systems that arises during the calculation. This accelerates the method, but the factors are surely much less sparse than the matrix itself. This, combined with the clustering of the eigenvalues near zero, leads to a nonlinear scaling, which was observed also in our experiments.

Alternative approaches are Jacobi–Davidson and deflation accelerated conjugate gradient [Bergamaschi and Putti 02], which seem to be highly competitive with the Lanczos method. In particular, in the Jacobi–Davidson method, it is still necessary to solve inner linear systems, but the factorization is avoided and substituted by the use of preconditioned iterative methods based on Krylov spaces. Deflation accelerated conjugate gradient sequentially computes the eigenpairs by minimizing the Rayleigh quotient $q(z) = z^T Az/z^T z$ over the subspace orthogonal to the eigenvectors previously computed. Finally, we mention the multilevel algorithm implemented in the `HSL_MC73` routine of the HSL mathematical software library, which, however, computes only the second-smallest eigenpair [Hu and Scott 03].

5. Current-Flow Betweenness Centrality

A large volume of research on networks has been devoted to the concept of centrality [Sabidussi 66, Freeman 79, Borgatti 05, Newman 10]. This research addresses the following fundamental question: What are the most important or central vertices in a network? There are four measures of centrality that

are widely used in network analysis: degree centrality, eigenvector centrality, closeness, and betweenness. Here, we focus on betweenness centrality.

Betweenness measures the extent to which a node lies on paths between other nodes. Nodes with high betweenness might have considerable influence within a network by virtue of their control over information (or over whatever else flows on the network) passing between other nodes. They are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths between other nodes.

Typically, only geodesic paths are considered in the definition of betweenness, yielding a measure that is called shortest-path betweenness. However, the shortest-path approach, as well as any other approach based on optimal paths, has a serious drawback. Paths even slightly longer than the shortest one give no contribution to the measure. This, as effectively shown in [Newman 05], can produce some odd effects. In real networks, on the other hand, it is not rare that information (or whatever else flows on the network) takes a more circuitous route, either by chance or because it is intentionally channeled through many intermediaries [Stephenson and Zelen 89]. In a social network, for instance, a fad does not know the optimal route to move among actors; it simply wanders around more or less randomly.

Current-flow betweenness centrality is a version of betweenness that includes contributions of all paths, although longer paths give a lesser contribution [Newman 05, Brandes and Fleischer 05]. For a given node, current-flow betweenness measures the current flow that passes through the vertex when a unit of current is injected in a source node and removed from a target node, averaged over all source–target pairs. Equivalently, it is equal to the net number of times that a random walk on the graph passes through the node on its journey, averaged over a large number of trials of the random walk. As observed in [Newman 10], current-flow betweenness centrality is appropriate for traffic that traverses a network with no idea of where it is going, while shortest-path betweenness is appropriate for information that knows exactly where it is going and takes the most direct path to get there.

We next give the precise definition of current-flow betweenness centrality in terms of resistor networks. Consider a network in which the edges are resistors and the nodes are junctions between resistors. Each edge is assigned a positive weight indicating the conductance of the edge. The resistance of an edge is the inverse of its conductance. Outlets are particular nodes where current enters and leaves the network. A vector u called *supply* defines them: a node i such that $u_i \neq 0$ is an outlet. In particular, if $u_i > 0$, then node i is a source, and current enters the network through it, while if $u_i < 0$, then node i is a target, and current leaves the network through it. Since there should be as much current entering the

network as leaving it, we have that $\sum_i u_i = 0$. We consider the case that a unit of current enters the network at a single source s and leaves it at a single target t. That is, $u_i^{(s,t)} = 0$ for $i \neq s, t$, $u_s^{(s,t)} = 1$, and $u_t^{(s,t)} = -1$. We are interested in how current flows through the network for an arbitrary choice of source and target outlets.

Let $v_i^{(s,t)}$ be the potential of node i, measured relative to any convenient reference potential, for source s and target t outlets. Kirchhoff's law of current conservation implies that the node potentials satisfy the following equation for every node i:

$$\sum_j A_{i,j} \left(v_i^{(s,t)} - v_j^{(s,t)} \right) = u_i^{(s,t)}, \tag{5.1}$$

where A is the weighted adjacency matrix of the network. The current flow through edge (i, j) is the quantity $A_{i,j}(v_i^{(s,t)} - v_j^{(s,t)})$, that is, the difference of potentials between the involved nodes multiplied by the conductance of the edge. A positive value indicates that the current flows in a certain direction (say from i to j), and a negative value means that the current flows in the opposite direction. Hence, Kirchhoff's law states that the current flowing in or out of any node is zero, with the exception of the source and target nodes.

In matrix form, (5.1) reads

$$(D - A)v^{(s,t)} = Gv^{(s,t)} = u^{(s,t)}, \tag{5.2}$$

where D is a diagonal matrix such that the ith element of the diagonal is equal to $\sum_j A_{i,j}$, that is, it is the (generalized) degree of node i. Recall that $G = D - A$ is the graph Laplacian matrix. As observed in Section 2, if G^+ is the generalized inverse of the Laplacian matrix G, then the potential vector is

$$v^{(s,t)} = G^+ u^{(s,t)}. \tag{5.3}$$

This means that the potential of node i with respect to source s and target t outlets is given by $v_i^{(s,t)} = G^+_{i,s} - G^+_{i,t}$. Therefore, the generalized inverse matrix G^+ contains information to compute all node potentials for any pair of source–target nodes.

An example of a resistor network with node potential solution is provided in Figure 1. Observe that Kirchhoff's law is satisfied for each node. For instance, the current entering at node B is 0.47 (from node A), which equals the current leaving node B, which is again 0.47 (0.13 to E, 0.27 to F, and 0.07 to C). Moreover, the current leaving the source node A is 1, and the current entering the target node H is also 1. Note that there is no current on the edge from C to D, since both nodes have the same potential. Any other potential vector obtained from the given solution by adding a constant is also a solution, since the potential

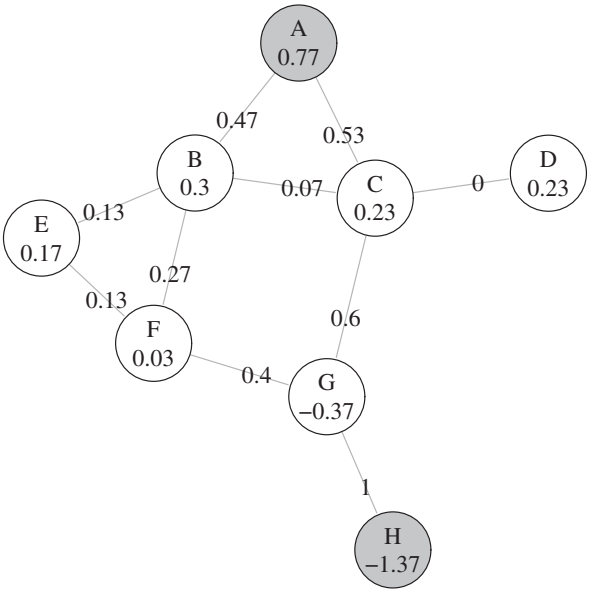

Figure 1. A resistor network with all resistances equal to unity. Each node is identified with a letter and is labeled with the value of its potential when a unit current is injected at node A and removed from node H. Each edge is labeled with the absolute current flowing on it.

differences remain the same, and hence Kirchhoff's law is satisfied. The given potential vector is, however, the solution with minimum Euclidean norm.

We have now all the ingredients to define current-flow betweenness centrality. As observed above, given a source s and a target t, the absolute current flow through edge (i, j) is the quantity $A_{i,j}|v_i^{(s,t)} - v_j^{(s,t)}|$. By Kirchhoff's law, the current that enters a node is equal to the current that leaves the node. Hence, the current flow $F_i^{(s,t)}$ through a node i different from the source s and a target t is half of the absolute flow on the edges incident in i:

$$F_i^{(s,t)} = \frac{1}{2} \sum_j A_{i,j} \left| v_i^{(s,t)} - v_j^{(s,t)} \right|. \tag{5.4}$$

Moreover, the current flows $F_s^{(s,t)}$ and $F_t^{(s,t)}$ through both s and t are set to 1 if endpoints of a path are considered part of the path (this is our choice in the rest of this paper), and to 0 otherwise. Figure 2 gives an example. Note that the flow from A to H through node G is 1 (all paths from A to H pass eventually through G), the flow through F is 0.4 (a proper subset of the paths from A to H go through F, and these paths are generally longer than for G), and the flow

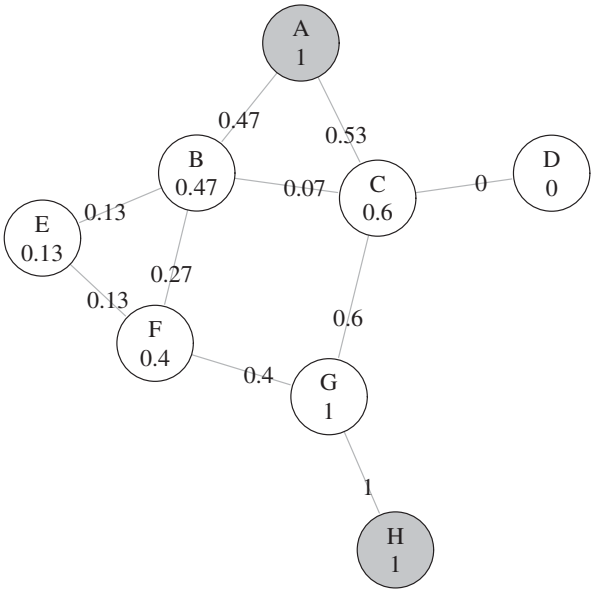

Figure 2. A resistor network with all resistances equal to unity (this is the same network of Figure 1). Each node is now labeled with the value of flow through it when a unit current is injected at node A and removed from node H. Each edge is labeled with the absolute current flowing on it.

through E is 0.13 (a proper subset of the paths from A to H go through E, and these paths are generally longer than for F).

Finally, the current-flow betweenness centrality b_i of node i is the flow through i averaged over all source–target pairs (s, t):

$$b_i = \frac{\sum_{s<t} F_i^{(s,t)}}{\frac{1}{2}n(n-1)}. \tag{5.5}$$

Since the potential $v_i^{(s,t)}$ is equal to $G^+_{i,s} - G^+_{i,t}$, with G^+ the generalized inverse of the graph Laplacian, (5.4) can be expressed in terms of elements of G^+ as follows:

$$F_i^{(s,t)} = \frac{1}{2}\sum_j A_{i,j}\left|G^+_{i,s} + G^+_{j,t} - G^+_{i,t} - G^+_{j,s}\right|. \tag{5.6}$$

Hence, if we replace matrix G^+ in (5.6) with its kth cutoff approximation $T^{(k)}$ (or its kth stretch approximation $S^{(k)}$), we get an approximate value of current-flow betweenness centrality for node i.

The computational complexity of (5.5) is as follows. We denote by k_i the number of neighbors of node i, that is, the number of edges incident in i. Assuming that we have matrix G^+, computing (5.6) for given i, s, and t costs $O(k_i)$ if i has at least one neighbor, and $O(1)$ otherwise. Hence, (5.5) for a specific i costs $O(n^2 \cdot \min(k_i, 1))$. Since $\sum_i \min(k_i, 1) = O(n + m)$, (5.5) for all nodes costs $O(n^2(n + m))$, that is, $O(n^3)$ if the graph is sparse.[1] Furthermore, the complexity of computing the equation using either cutoff or stretched approximations increases by a factor of k, since an entry (i, j) of $T^{(k)}$ or of $S^{(k)}$ can be obtained in $O(k)$. If k is constant with respect to n, then there is no asymptotic increase of complexity. Clearly, if n is large, the cost of (5.5) is prohibitive.

A possible solution for the bottleneck of (5.5) is the adoption of a probabilistic approach by computing (5.6) for only a sample of source–target pairs [Brandes and Fleischer 05]. If we choose at random αn source nodes, with $0 < \alpha < 1$, and for each source, we choose at random αn target nodes, then the sample of source–target pairs has size $\alpha^2 n^2$. For instance, if $\alpha = 0.1$, then $\alpha^2 = 0.01$, and hence the cost of computing (5.5) declines by two orders of magnitude.

An alternative solution to avoid the bottleneck of (5.5) is to use cutoff approximation with $k = 2$ (only the second-smallest eigenpair is used). In this case, (5.5) significantly simplifies. Indeed, if $k = 2$, then $T^{(2)}_{i,j} = (1/\lambda_2)V_{i,2}V^T_{j,2}$. It follows that flow $F^{(st)}_i$ can be approximated by

$$\begin{aligned}
\hat{F}^{(s,t)}_i &= \frac{1}{2}\sum_j A_{i,j}\left|T^{(2)}_{i,s} + T^{(2)}_{j,t} - T^{(2)}_{i,t} - T^{(2)}_{j,s}\right| \\
&= \frac{1}{2}\sum_j A_{i,j}\frac{1}{\lambda_2}\left|V_{s,2} - V_{t,2}\right|\left|V_{i,2} - V_{j,2}\right| \\
&= \frac{1}{2\lambda_2}\left|V_{s,2} - V_{t,2}\right|\sum_j A_{i,j}\left|V_{i,2} - V_{j,2}\right|.
\end{aligned}$$

Observe that the sum in the formula for the flow is now independent of the source–target pair. Hence, the approximate betweenness of i is

$$\hat{b}_i = \frac{2}{n(n-1)}\sum_{s<t}\frac{1}{2\lambda_2}|V_{s,2} - V_{t,2}|\sum_j A_{i,j}|V_{i,2} - V_{j,2}| = C\sum_j A_{i,j}|V_{i,2} - V_{j,2}|,$$

where C is a constant. This means that the approximate betweenness $\hat{b}_i$ is proportional to the quantity $\sum_j A_{i,j}|V_{i,2} - V_{j,2}|$. Note that $\hat{b}_i$ is a local version of b_i that depends only on the neighborhood of i. If we ignore the

[1] This cost can be improved to $O(mn \log n)$, hence $O(n^2 \log n)$ for sparse networks, as shown in [Brandes and Fleischer 05].

multiplicative constant C, the approximate betweenness $\hat{b}_i$ can be computed in $O(n+m)$ for all nodes, hence in linear time with respect to the size of the network.

We conclude this section by computing exact and approximate betweenness centrality on a real network. The instance is a social network of dolphins (*Tursiops truncatus*) belonging to a community that lives in the fjord of Doubtful Sound in New Zealand. The unusual conditions of this fjord, with relatively cool water and a layer of fresh water on the surface, have limited the departure of dolphins and the arrival of new individuals in the group, facilitating a strong social relationship within the dolphin community. The network is an undirected unweighted graph containing 62 nodes (dolphins) and 159 nondirectional connections between pairs of dolphins. Two dolphins are joined by an edge if during the observation period, which lasted from 1994 to 2001, they were spotted together more often than expected by chance. This network has been extensively studied by David Lusseau and coauthors; see, for instance, [Lusseau and Newman 04].

The ranking obtained with the cutoff approximation method using only three eigenpairs of the graph Laplacian correlates at 0.92 with the exact ranking, and the mean change of rank between the two compilations is 5.4. Moreover, the stretch approximation method performs even better. Using the same number of eigenpairs (three), the approximate and exact rankings correlate at 0.99, and the mean change of rank between the two compilations is just two positions. Figure 3 depicts the dolphin social network, where the size of the node is proportional to its exact current-flow betweenness (graph on the left) and to its approximate current-flow betweenness (graph on the right). Moreover, Table 1 shows the top ten of both compilations, and Figure 4 gives a scatter plot of the two rankings. Nine dolphins out of ten are present in both top-ten rankings: the missing dolphins (DN63 and Grin) both rank 11th in the other ranking. Six dolphins have the same rank in both compilations (notably, the top four are the same). In fact, the stretch approximation method is already effective using just one eigenpair (correlation at 0.98 and mean change at 2.9). These outcomes suggest that the proposed approximation methods for betweenness, in particular the stretch version, are effective.

The dolphin network, while interesting, is quite a small network. Potentially, our approximation methods are applicable to much larger networks, in particular when some efficient method for finding the partial eigenspectrum of large sparse matrices is exploited. In Section 6.4, we test the approximation methods on larger graphs and use interpolation of the results to estimate the time needed to run the methods on relatively large networks.

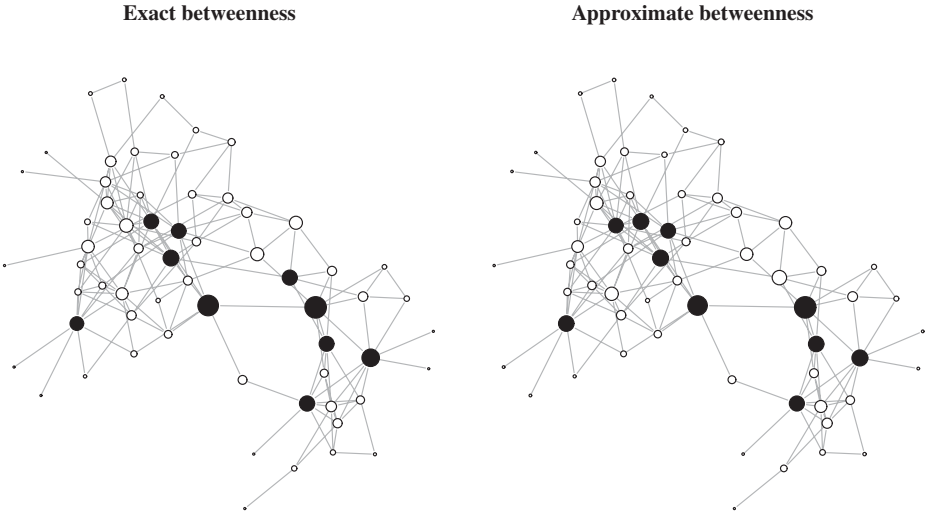

Figure 3. A dolphin social network. The size of the nodes is proportional to the current-flow betweenness. The exact solution is shown on the left, and the approximate one is shown on the right. Black nodes are the top ten leaders in both networks (nine of which are shared by the two networks). The approximation uses the stretch method with three eigenpairs.

6. Experimental Evaluation

This section is devoted to the experimental evaluation of effectiveness and efficiency of the proposed approximations for the generalized inverse of the

Dolphin	Exact	Dolphin	Approximate
Beescratch	0.254	Beescratch	0.290
SN100	0.244	SN100	0.266
Jet	0.209	Jet	0.222
SN9	0.189	SN9	0.222
Web	0.183	SN4	0.220
DN63	0.181	Trigger	0.217
Upbang	0.179	Upbang	0.215
SN4	0.177	Web	0.211
Kringel	0.176	Kringel	0.206
Trigger	0.165	Grin	0.204

Table 1. The top-ten rankings according to exact betweenness (column Exact) and approximate betweenness (column Approximate). The approximation uses the stretch method with three eigenpairs.

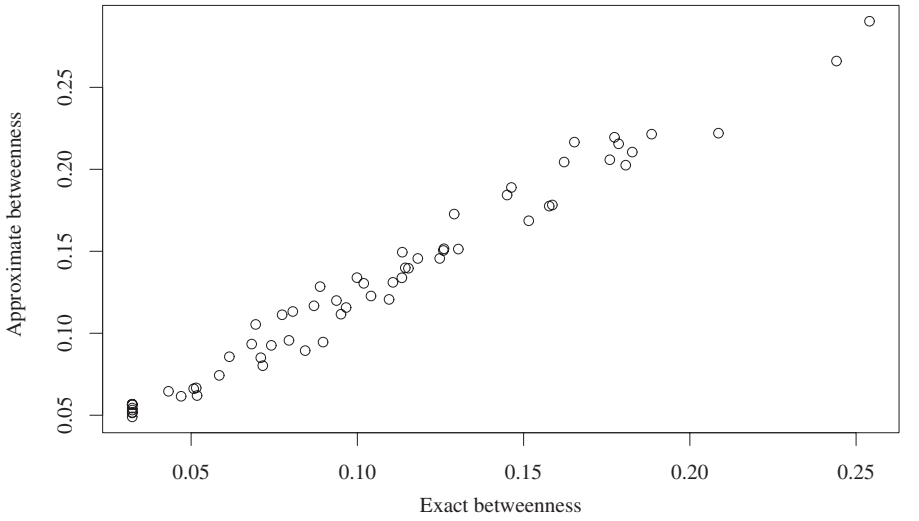

Figure 4. A scatter plot of exact and approximate betweenness rankings: the rank of a dolphin in the exact ranking is plotted against the rank of the same dolphin in the approximate ranking. The approximation uses the stretch method with three eigenpairs.

Laplacian of a graph, and in particular for the current-flow betweenness centrality scores of nodes.

6.1. Experimental Setting

In our experiments, we took advantage of the following two well-known network models. The first is the random model, also known as the Erdős–Rényi model (ER model for short) [Solomonoff and Rapoport 51, Erdős and Rényi 60]. According to this model, a network is generated by laying down a number of nodes and adding edges between them with independent probability for each node pair. This model generates a small-world network with a binomial node degree distribution, which is peaked at a characteristic scale value. We will denote a graph drawn according to the ER model by $\mathrm{ER}(n, q)$, where n is the number of nodes and $p = q/n$ is the edge probability. Such a graph has roughly $m = q(n-1)/2$ edges. An ER graph is not necessarily connected, but it contains a giant connected component containing most of the nodes as soon as $q \geq 1$. We extracted the giant component from the generated graphs and used this component in our experiments.

However, many real networks are scale-free, that is, the degree distribution has a long tail that roughly follows a power law [Newman 10]. A network model

that captures the power-law distribution of node degrees is known as cumulative advantage [Simon 55, de Solla Price 76], which was later rediscovered and further investigated under the name preferential attachment or Barabási–Albert model (BA model for short) [Barabási and Albert 99, Barabási et al. 99, Bollobás et al. 01]. Such a model, as described by [Barabási and Albert 99], has the following main ingredients:

Startup: The graph has a single isolated node.

Growth: Additional nodes are added to the network one at a time.

Preferential attachment: Each node connects to the existing nodes with a fixed number r of links. The probability that it will choose a given node is proportional to the number of links the chosen node already has.

The resulting network is a small-world graph with a power-law degree distribution: most of the nodes (the trivial many) will have low degree, and a small but significant share of nodes (the vital few or hubs) will have an extraordinarily high degree. We will denote a graph drawn according to the BA model by $\mathrm{BA}(n, r)$, where n is the number of nodes and r is the number of edges attached to each node added during the preferential attachment step. Such a graph has $m = r(n-1)$ edges. Note that the resulting graph is connected, but it is not necessarily simple: multiple edges might exist between the same pair of nodes. We simplified the graph in our experiments, removing multiple edges, if any.

In this section we give results for unweighted networks only. We also experimented with weighted graphs with edge weights drawn from a random uniform distribution, but we noticed no particular discrepancy with respect to the unweighted case.

Our experiments were performed within the R computing environment, taking advantage of the `igraph` package for the generation of networks. We used the R interface to `LAPACK` (Linear Algebra Package) [Anderson et al 99] for matrix inversion and eigenpairs generation, as well as the R interface to `ARPACK` (Arnoldi Package) [Lehoucq et al. 98] when a few eigenpairs of large sparse matrices were necessary. Moreover, we exploited the C programming language for a faster implementation of the computation of betweenness scores using (5.5), calling the C compiled code from the R environment (R is rather slow at iterative algorithms that require loops iterated many times). All experiments were run on a MacBook Pro equipped with a 2.3-GHz Intel Core i5 processor, 4 GB 1333-MHz DDR3 memory, running Mac OS X Lion 10.7.2.

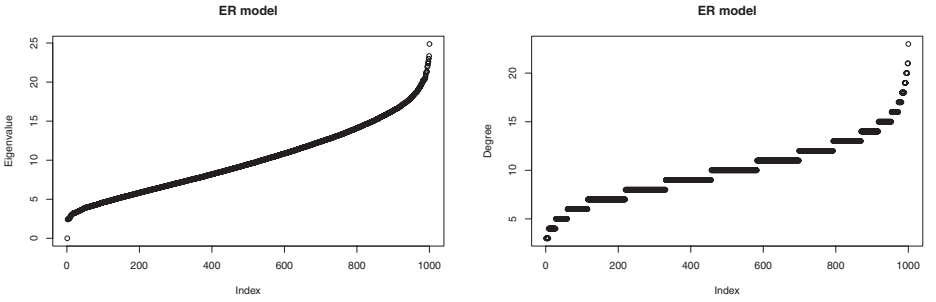

Figure 5. Plot of sorted eigenvalues (left plot) and sorted degrees (right plot) on ER($n = 1000, q = 10$).

6.2. Distribution of Eigenvalues and Node Degrees

We begin the experimental investigation with an exploratory analysis of the distribution of eigenvalues of the Laplacian, comparing with the distribution of node degrees on both ER and BA networks. This study is useful in understanding the behavior of the proposed approximations.

The exploratory experiment uses graphs ER($n = 1000, q = 10$) and BA($n = 1000, r = 5$). Observe that the graphs used have the same number of nodes and approximately the same number of edges, hence a similar edge density. Nevertheless, they differ in the way the edges are placed among nodes.

Figures 5 and 6 explore the distribution of eigenvalues of the Laplacian as well as the distribution of node degrees on the ER and BA graphs, respectively. We observe that the eigenvalues of the Laplacian are well approximated by the degrees of the nodes, an interesting phenomenon already investigated in [Zhan et al. 10]. For the ER model, eigenvalues and degrees have a correlation of 0.989;

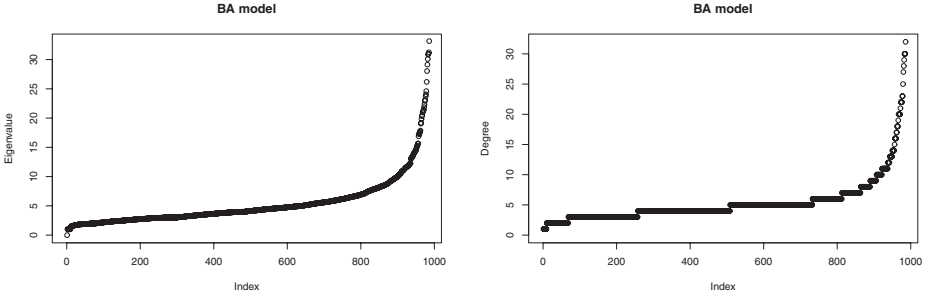

Figure 6. Plot of sorted eigenvalues (left plot, up to index 980) and sorted node degrees (right plot, up to index 980) on BA($n = 1000, r = 5$).

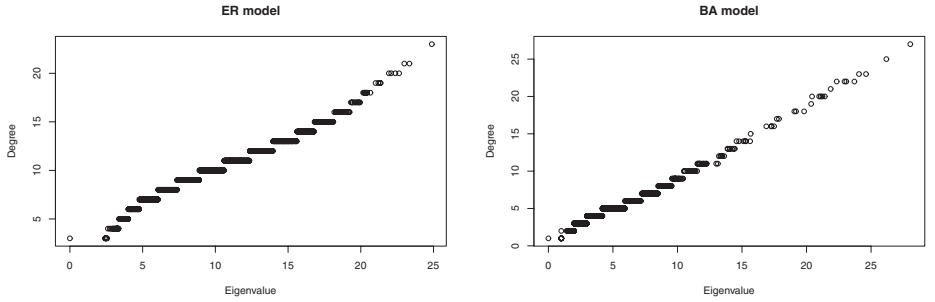

Figure 7. Scatter plot of eigenvalues versus node degrees on ER($n = 1000, q = 10$) (left plot) and on BA($n = 1000, r = 5$) (right plot).

the Euclidean distance between the two sorted vectors, relative to the 2-norm of the eigenvalue vector, is 0.13. Using a BA model, eigenvalues and degrees are correlated at 0.983, and the relative Euclidean distance between the two sorted vectors is 0.02. Figure 7 shows a scatter plot of Laplacian eigenvalues versus node degrees.

The plots of the sorted eigenvalues of the Laplacian on the ER and BA models are visibly different (Figure 8, left plot). In the initial part (up to index 800), ER eigenvalues grow faster than BA eigenvalues; however, in the tail of the plot, BA eigenvalues rapidly catch up, overtaking ER eigenvalues around index 960. From here to the end of the sequence, BA eigenvalues literally explode (observe that in order to show the behavior of smaller eigenvalues, the figure shows the eigenvalue curve up to index 980). The eigenvalues of the generalized inverse of the Laplacian, which are the inverses of the Laplacian eigenvalues, are shown in Figure 8, right plot. Both curves decrease rapidly at the beginning, but the ER curve has a more significant drop; after this initial fall, both curves

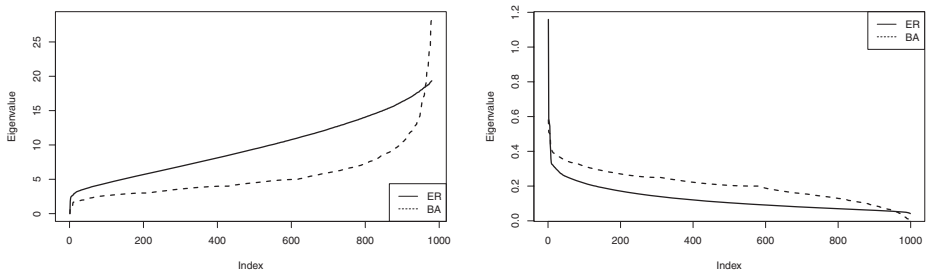

Figure 8. Plot of eigenvalues of Laplacian (left plot, up to index 980) and of generalized inverse Laplacian (right plot, up to index 980) on ER($n = 1000, q = 10$) and BA($n = 1000, r = 5$).

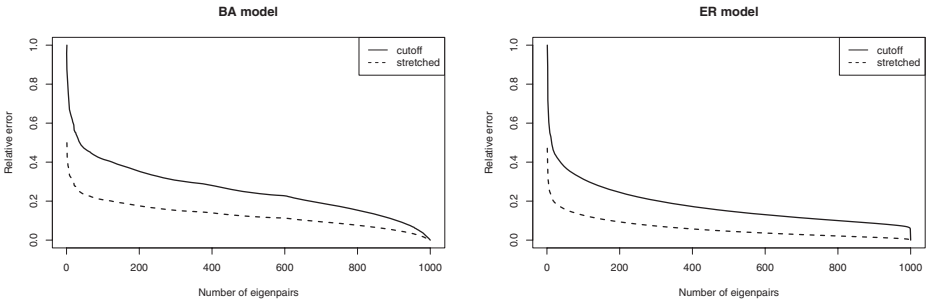

Figure 9. Relative 2-norm error of cutoff and stretch approximations on BA($n = 1000, r = 5$) (left plot) and ER($n = 1000, q = 10$) (right plot).

decreases gently, although the BA line declines faster. This might be a hint of the effectiveness of the stretch approximation method: the tail of the eigenvalue curve has a small slope; hence the eigenvalues of the tail can be well approximated by a single middle value.

6.3. Effectiveness of Approximations

In this section we investigate the effectiveness of approximations of the generalized inverse of the Laplacian and, in particular, of the current-flow betweenness rankings. We first study the relative 2-norm error of the kth cutoff and stretch approximations, defined as the ratio between $\|G^+ - T^{(k)}\|_2$ and $\|G^+\|_2$ (cutoff approximation error), and the ratio between $\|G^+ - S^{(k)}\|_2$ and $\|G^+\|_2$ (stretch approximation error). Figure 9 shows the approximation error as the number of eigenpairs k grows from 1 to n. In both network models, the stretch approximation is significantly more effective than the cutoff one: on average, the stretch approximation errors are 30% and 50% of the cutoff approximation error on the ER and BA graphs, respectively. In particular, the stretch approximation is more effective on ER graphs, as predicted by the spectral behavior highlighted in Figure 8.

We next test the effectiveness of the approximations applied to current-flow betweenness. The quality of the approximations is established by computing the correlation coefficient among the approximate and exact node rankings. We used the Pearson product-moment correlations, whereby the input data is logarithmically transformed when it is not normally distributed. This coefficient is largely used in the natural sciences to measure the similarity of two rankings; it runs from -1 to 1, where values close to -1 indicate negative correlation (one ranking is the reverse of the other), values close to 0 correspond to null correlation (the two rankings are independent), and values close to 1 denote positive correlation

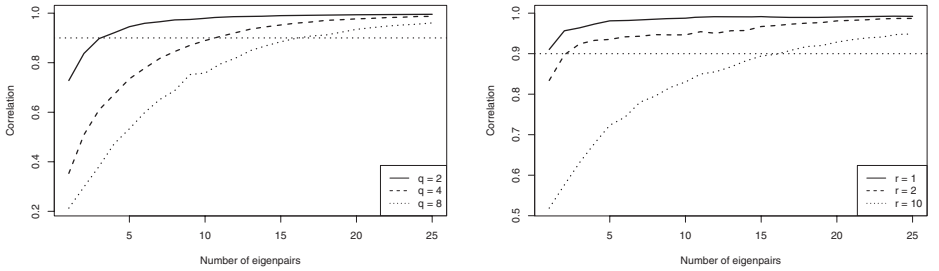

Figure 10. Effectiveness of cutoff approximation for betweenness, increasing the number of eigenpairs from 1 to 25 on ER($n = 100, q$) for different values of q (plot on the left) and on BA($n = 100, r$) for different values of r (plot on the right).

(the two rankings are similar). A good approximation method, in our assessment, is one that approaches the exact ranking with correlation of at least 0.9.

The effectiveness experiments were run on the ER and BA graphs with 100 nodes and an increasing number of edges: we used $q = 2, 4, 8$ for the the ER model and $r = 1, 2, 10$ for the BA model. The corresponding ER and BA graphs have roughly the same density. Notice that a BA graph with $r = 1$ is a tree, hence an acyclic graph, while $r > 1$ generates graphs with loops. We always generated a sample of 100 such graphs and took the average of the observed correlation coefficients.

Figure 10 shows the effectiveness of the cutoff approximation for betweenness on ER and BA graphs with increasing density. The quality of the approximation increases with the number of eigenpairs that are used in the approximation and decreases, for both network models, as the graphs become denser. A cumulative comparison between cutoff and stretch approximations as well as among ER and BA network models is given in Figure 11. The stretch approximation performs neatly better than the cutoff one. Regardless of the graph model and of the graph density, the effectiveness of the stretch method is well above the quality threshold of 0.9 already with a single eigenpair.

Finally, we tested a probabilistic approximation of betweenness that uses only a sample of node pairs for solving (5.5). In Figure 12 we compare the probabilistic approach on both the exact version of betweenness, which uses the full generalized Laplacian inverse matrix for the computation of betweenness scores, and the stretch version of betweenness, which uses the stretch approximation method with one eigenpair only. The plot shows the effectiveness of the combined approximation algorithm, which uses the stretch method (with a single eigenpair) for the approximation of the generalized Laplacian inverse and the probabilistic approach for the computation of betweenness scores: less than 10%

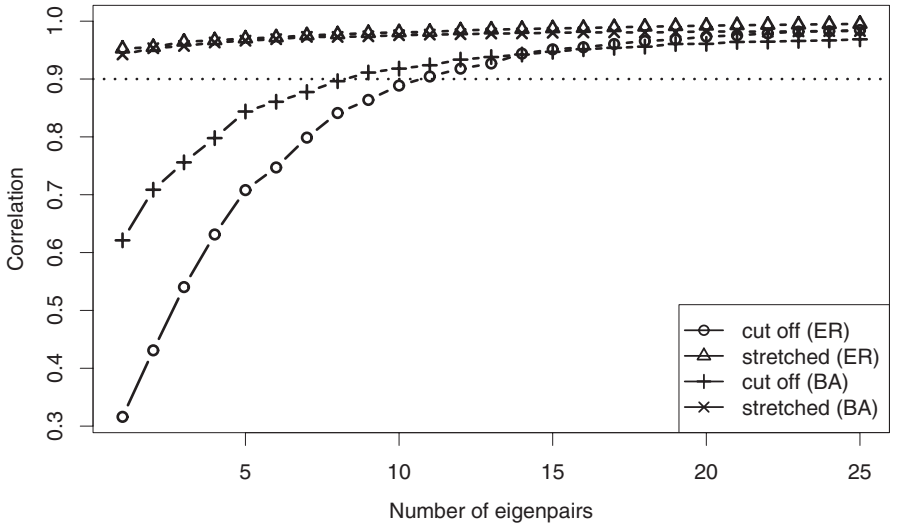

Figure 11. Comparison of cutoff and stretch approximations for betweenness increasing the number of eigenpairs from 1 to 25 on ER($n = 100, q = 4$) and BA($n = 100, r = 2$).

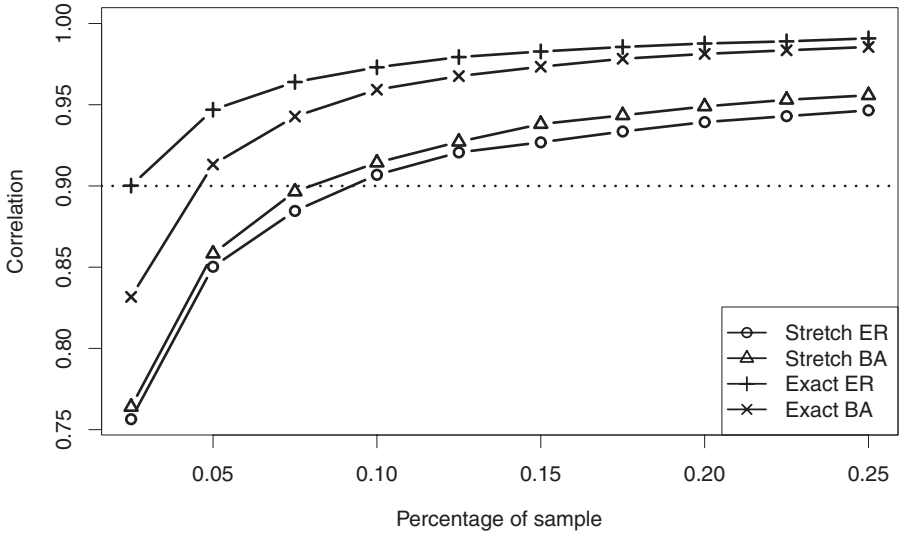

Figure 12. Comparison of the probabilistic approach on both exact and stretch betweenness (one eigenpair), increasing the size of the node pair sample from 2.5% to 25% on ER($n = 100, q = 4$) and BA($n = 100, r = 2$).

of the node pair space is sufficient to get an approximate ranking that correlates at 0.9 with the exact ranking.

6.4. Efficiency of Approximations

In this section we show the outcomes of experiments about the efficiency of the proposed approximations for the generalized inverse of the Laplacian matrix. We compare elapsed time of the computation of the full generalized inverse of the Laplacian with that of the computation of the approximation that uses only one eigenpair (the second-smallest eigenvalue and the corresponding eigenvector).

Figure 13 shows the elapsed time necessary for the computation of the generalized Laplacian inverse for BA and ER networks. The time growth is $O(n^3)$ and the memory requirement is $O(n^2)$ on both models, where n is the number of nodes of the graph. These costs make the computation infeasible on large networks. For instance, for a large network with 10^6 nodes, the necessary time would be 642 days with a storage of about one terabyte. On the other hand, Figure 14 gives the elapsed time necessary for the computation of the approximation that uses only one eigenpair for BA and ER networks. For both models, the time growth is $O(n^{1.5})$, and the memory requirement is $O(n)$. Note that the computation on ER graphs is one order of magnitude more efficient than that of BA graphs. These complexities make it possible to approximate the generalized inverse of the Laplacian, and in particular current-flow betweenness scores, on relatively large networks: on a standard machine, the computation on a random network with 10^6 nodes would last less than one hour, and it would take about

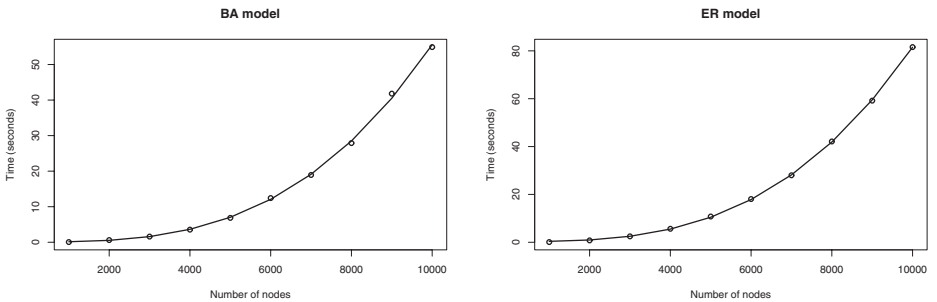

Figure 13. Efficiency of the computation of the generalized inverse of the Laplacian matrix of BA graphs ($r = 2$, plot on the left) and ER graphs ($q = 4$, plot on the right) with increasing number of nodes. The data fit a cubic curve $ax^3 + b$ with $b = 8.906 \cdot 10^{-2}$ and $a = 5.545 \cdot 10^{-11}$ for the BA model (R-Square = 0.9992), and a cubic curve $ax^3 + b$ with $b = 3.038 \cdot 10^{-1}$ and $a = 8.120 \cdot 10^{-11}$ for the ER model (R-Square = 0.9999).

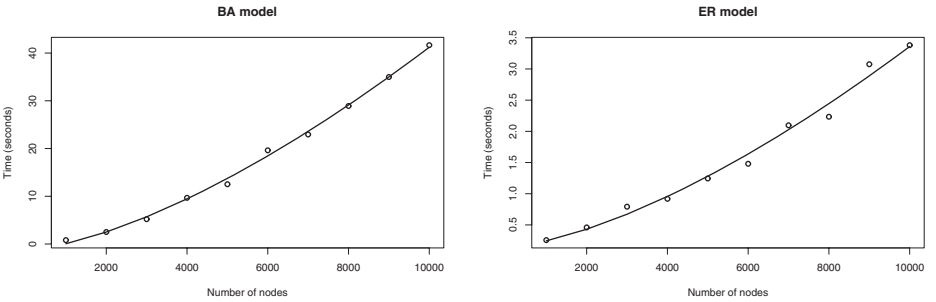

Figure 14. Efficiency of the computation of the second smallest eigenpair of the Laplacian matrix of BA graphs ($r = 2$, plot on the left) and ER graphs ($q = 4$, plot on the right) with increasing number of nodes. The data fits a curve $ax^{1.5} + b$ with $b = -1.292$ and $a = 4.25 \cdot 10^{-5}$ for the BA model (R-Square = 0.9976), and a curve $ax^{1.5} + b$ with $b = 1.426 \cdot 10^{-1}$ and $a = 3.217 \cdot 10^{-6}$ for the ER model (R-Square = 0.9877).

12 hours (a reasonable time) on a scale-free network with the same number of nodes. In both cases, the required storage is only one megabyte.

7. Conclusion

We have proposed effective and efficient methods for approximating the generalized inverse Laplacian matrix of a graph, a matrix that arises in many graph-theoretic applications. A notable such application investigated in the present contribution is current-flow betweenness centrality, a measure for finding central nodes in a graph that lie between many other nodes. Our approximation methods turn out to be suitable when the network at hand is large, so that computing the full generalized inverse is not realistic.

Many other applications can be sought, such as, for instance, the approximation of the resistance distance matrix of a graph. Resistance distance is a metric on the graph, alternative to the classical shortest-path distance, that was defined, independently, in [Stephenson and Zelen 89], following an information-theoretic approach, and in [Klein and Randić 93], following an electrical-theoretic approach. The resistance distance notion has a number of interesting interpretations and many applications, even outside of network science [Ghosh et al. 08]. It turns out that the resistance distance matrix can be immediately defined in terms of the generalized inverse Laplacian matrix [Ghosh et al. 08], allowing us to extend our approximation methods to resistance distance matrices as well.

References

[Aho et al. 74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[Anderson et al 99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J., Demmel, J. Dongarra, et al. *LAPACK Users Guide*, 3rd edition. SIAM, 1999.

[Barabási and Albert 99] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks." *Science* 286 (1999), 509–512.

[Barabási et al. 99] A.-L. Barabási, R. Albert, and H. Jeong. "Mean-Field Theory for Scale-Free Random Networks." *Physica A* 272:1-2 (1999), 173–187.

[Ben-Israel and Greville 03] A. Ben-Israel and T. N. Greville. *Generalized Inverses. Theory and Applications*, 2nd edition. Springer, 2003.

[Bergamaschi and Putti 02] L. Bergamaschi and M. Putti. "Numerical Comparison of Iterative Eigensolvers for Large Sparse Symmetric Positive Definite Matrices." *Computer Methods in Applied Mechanics and Engineering* 191:45 (2002), 5233–5247.

[Bollobás et al. 01] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády. "The Degree Sequence of a Scale-Free Random Graph Process." *Random Structures and Algorithms* 18:3 (2001), 279–290.

[Borgatti 05] S. P. Borgatti. "Centrality and Network Flow." *Social Networks* 27:1 (2005) 55–71.

[Brandes and Erlebach 05] U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations*, Lecture Notes in Computer Science 3418. Springer, 2005.

[Brandes and Fleischer 05] U. Brandes and D. Fleischer. "Centrality Measures Based on Current Flow." In *Proc. 22nd Symp. Theoretical Aspects of Computer Science (STACS '05)*, pp. 533–544. Springer, 2005.

[de Solla Price 76] D. de Solla Price. "A General Theory of Bibliometric and Other Cumulative Advantage Processes." *Journal of the American Society for Information Science* 27 (1976), 292–306.

[Erdős and Rényi 60] P. Erdős and A. Rényi. "On the Evolution of Random Graphs." *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5 (1960), 17–61.

[Freeman 79] L. C. Freeman. "Centrality in Networks: I. Conceptual Clarification." *Social Networks* 1 (1979), 215–239.

[Ghosh et al. 08] A. Ghosh, S. Boyd, and A. Saberi. "Minimizing Effective Resistance of a Graph." *SIAM Review* 50:1 (2008), 37–66.

[Golub and Meurant 10] G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with Applications*, Princeton Series in Applied Mathematics. Princeton University Press, 2010.

[Golub and Van Loan 96] G. H. Golub and C. F. Van Loan. *Matrix Computations*, 3rd edition. The Johns Hopkins University Press, 1996.

[Hu and Scott 03] Y. Hu and J. A. Scott. "HSL_MC73: A fast Multilevel Fiedler and Profile Reduction Code." Tech. Rep. RAL-TR-2003-036, Rutherford Appleton Laboratory, 2003.

[Klein and Randić 93] D. J. Klein and M. Randić. "Resistance Distance." *Journal of Mathematical Chemistry* 12:1 (1993), 81–95.

[Lehoucq et al. 98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, 1998.

[Lusseau and Newman 04] D. Lusseau and M. E. J. Newman. "Identifying the Role That Animals Play in Their Social Networks." *Proceedings of the Royal Society B: Biological Sciences* 271 (2004), S477–S481.

[Newman 05] M. Newman. "A Measure of Betweenness Centrality Based on Random Walks." *Social Networks* 27:1 (2005), 39–54.

[Newman 10] M. E. J. Newman. *Networks: An Introduction.* Oxford University Press, 2010.

[Newman et al. 06] M. E. J. Newman, A.-L. Barabási, and D. J. Watts. *The Structure and Dynamics of Networks.* Princeton University Press, 2006.

[Raz 03] R. Raz. "On the Complexity of Matrix Product." *SIAM Journal on Computing* 32:5 (2003), 1356–1369.

[Saad 03] Y. Saad. *Iterative Methods for Sparse Linear Systems*, 2nd edition. Society for Industrial and Applied Mathematics, 2003.

[Sabidussi 66] G. Sabidussi. "The Centrality Index of a Graph." *Psychometrika* 31 (1966), 581–603.

[Simon 55] H. A. Simon. "On a Class of Skew Distribution Functions." *Biometrika* 42 (1955), 425–440.

[Solomonoff and Rapoport 51] R. Solomonoff and A. Rapoport. "Connectivity of Random Nets." *Bulletin of Mathematical Biophysics* 13 (1951), 107–117.

[Stephenson and Zelen 89] K. Stephenson and M. Zelen. "Rethinking Centrality: Methods and Examples." *Social Networks* 11:1 (1989), 1–37.

[Zhan et al. 10] C. Zhan, G. Chen, and L. F. Yeung. "On the Distributions of Laplacian Eigenvalues versus Node Degrees in Complex Networks." *Physica A: Statistical Mechanics and Its Applications* 389:8 (2010), 1779–1788.

Enrico Bozzo, Department of Mathematics and Computer Science, University of Udine, Via delle Scienze 206, 33100 Udine, Italy (enrico.bozzo@uniud.it)

Massimo Franceschet, Department of Mathematics and Computer Science, University of Udine, Via delle Scienze 206, 33100 Udine, Italy (massimo.franceschet@uniud.it)