Random Walk Fundamental Tensor and Graph Importance Measures

Daniel Boley[1*] and **Alejandro Buendia**[2†]
[1]Univ. of Minnesota
[2]Microsoft AI Development Acceleration Program
boley@umn.edu, albuendi@microsoft.com

Abstract

Random walks over directed graphs are used to model activities in many domains, such as social networks, influence/trust propagation, and Bayesian graphical models. They are often used to compute the importance or centrality of individual nodes according to a variety of different criteria.

Here we show how a fundamental tensor, yielding a unified treatment of a wide variety of importance measures for graphs, can be obtained by precomputation of just one slice of the tensor. This leads to quick ways to compute measures such as the average number of visits to a given node and various centrality and betweenness measures for individual nodes, both for the network in general and in the case a subset of nodes is to be avoided. This could lead to fast answers to queries about major bottlenecks in the network.

Keywords: Networks, Big Data, Scalability

1 Introduction

Background. We consider random walks over a strongly connected directed graph, corresponding to a recurring Markov chain. A graph with n vertices can be fully described by its $n \times n$ adjacency matrix A whose ij-th entry a_{ij} is the weight on the directed edge from i to j, or zero if there is no such edge. Let $\mathbf{1}$ be the vector of all ones. We let $\boldsymbol{d} = A \cdot \mathbf{1}$ be the vector of out-degrees, where $d_i = \sum_j a_{ij}$, and $D = \textsc{Diag}(d)$ be the diagonal matrix with the entries of $\boldsymbol{d}$ on the diagonal. Then the corresponding Markov chain has transition probability matrix $P = D^{-1}A$. It is well known that if the graph is strongly connected, then P has a simple eigenvalue $\lambda = 1$ with corresponding eigenvector with all positive entries $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)^T$, which can be scaled to unit length in the 1-norm, i.e. the vector of stationary probabilities for the random walk. We let $\boldsymbol{\Pi} = \textsc{Diag}(\boldsymbol{\pi})$ be the corresponding diagonal matrix. There are several Laplacians that can be formed from this matrix as noted in [3]:

$$\begin{array}{ll} L = I - P & \text{normalized Laplacian} \\ \mathbf{L} = \boldsymbol{\Pi}(I - P) & \text{random walk Laplacian} \\ \mathcal{L} = D - A & \text{combinatorial Laplacian} \end{array} \tag{1}$$

It is well known that these Laplacians play critical roles with respect to undirected graphs. For undirected graphs, we have $\boldsymbol{d} = \boldsymbol{\pi} \cdot (\text{volume})$, where volume is twice the number of edges in the graph, and hence the combinatorial Laplacian is just a scalar multiple of the random walk Laplacian. For a directed graph, these "Laplacians" are no longer symmetric, but [3] showed how they can still be used to compute interesting properties of the graph.

Golnari et al. (see e.g. [11]) introduced the third-order *random walk fundamental tensor* $\boldsymbol{N} = \{\boldsymbol{N}(i,j,k)\}_{i,j,k=1}^{n}$ where each entry $\boldsymbol{N}(i,j,k)$ is the expected number of visits to intermediate node j on all walks starting from node i before reaching target node k. This is an extension of the so-called fundamental matrix N of an absorbing Markov chain [13], which is a matrix whose ij-th entry gives the expected number of random walk passages through node j when starting a random walk from node i before being absorbed. Each slice $\boldsymbol{N}(:,:,k)$ of the tensor is the fundamental matrix for the modified Markov chain where node k is turned into an absorbing node. This tensor can be used to quickly compute several centrality measures such as random walk closeness $\sum_i H_i^{[k]} = \sum_{i,j} \boldsymbol{N}(i,j,k)$ for a node k [20], and random walk betweenness $\sum_{i \neq j, k \neq j} \Pr(i \to j \to k)$ (i.e., the sum of probabilities of a random walk passing an intermediate node j averaged over all starting nodes i and ending nodes k) [19; 16].

Contribution and Outline. In this paper we revisit the random walk Laplacian [3] and show that we can compute its pseudoinverse using one matrix inversion plus other computations of $\mathcal{O}(n^2)$ complexity. The complexity of the matrix inversion is $O(n^3)$ using the usual off-the-shelf algorithms, but we give an indication that one can take advantage of the graph structure to reduce this complexity. We then show how this pseudoinverse can be used to efficiently calculate many useful properties of the graph through the use of a *fundamental tensor* [11], which leads to convenient calculation of hitting times, absorption probabilities, and centrality measures. Finally, we show how to use this tensor to quickly update these measures when certain nodes are to be removed or avoided, leading to possible ways to quickly identify local bottlenecks in a network.

After the preliminaries, we sketch some properties of the

*Contact Author
†Research conducted while at Columbia University.

random walk Laplacian, formally define the random walk fundamental tensor, and show how the tensor can be used to compute properties of the underlying graph based on its correspondence to the random walk.

Related Work. This work was motivated by a long literature of centrality and betweenness measures [4; 2; 19; 9; 16; 18], which have traditionally been difficult to compute. Random walk-based centrality measures have improved on other notions of centrality by accounting for propagation through all possible paths between a source and target. This is in contrast to previous metrics that have ranked agents exclusively by geodesics or by maximum flow through a particular choice of idealized paths [21; 10]. Random walk probabilities of passages through individual nodes are also used in influence propagation, such as in trust mechanisms [15; 14; 17].

2 Random Walk Laplacian and its Pseudoinverse

We consider a strongly connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $|\mathcal{V}| = n$. This digraph can be modeled by an irreducible Markov chain with stationary probability vector $\boldsymbol{\pi}$. By *strongly connected* we mean that there is a path from any node to any other node within the directed graph. For simplicity we assume the graph is unweighted, though the development can easily be extended to the weighted case. The probability transition matrix for the Markov chain is calculated as $P = D^{-1}A$, in which A is the adjacency matrix and $D = \textsc{Diag}(A \cdot \mathbf{1})$ is the diagonal matrix of vertex out-degrees.

The probability transition matrix for the random walk, treated as a Markov chain, is

$$P = \begin{pmatrix} P_{\alpha,\alpha} & \mathbf{r} \\ \mathbf{s}^T & \tau \end{pmatrix} = \begin{pmatrix} P_{\alpha,\alpha} & P_{\alpha,n} \\ P_{n,\alpha} & P_{nn} \end{pmatrix}, \tag{2}$$

where $\alpha = \{1, \ldots, n-1\}$ is the index set corresponding to all but the last vertex (in their arbitrary order), and n is a chosen absorbing state. Later on, we split $\alpha = \{\beta, \gamma\}$ into two parts β & γ, with γ temporarily treated as absorbing states. This matrix is row-stochastic (i.e., $P \cdot \mathbf{1} = \mathbf{1}$), and the vector of recurring probabilities $\boldsymbol{\pi} = (\pi_1; \ldots; \pi_n)$ satisfies $\boldsymbol{\pi}^T P = \boldsymbol{\pi}^T$, $\pi_1 > 0, \ldots, \pi_n > 0$, $\|\boldsymbol{\pi}\|_1 = 1$. Here we use ";" to denote vertical concatenation *à la* Matlab. If we were to assume no self-loops, τ would be zero.

The matrix P has a simple eigenvalue equal to 1 with right eigenvector $\mathbf{1}$ and left eigenvector $\boldsymbol{\pi}^T$:

$$P \cdot \mathbf{1} = \mathbf{1} \qquad \boldsymbol{\pi}^T P = \boldsymbol{\pi}^T$$

The left eigenvector is scaled by $\|\boldsymbol{\pi}\|_1 = \sum_j \pi_j = 1$ so that it is the vector of stationary probabilities for the recurring Markov chain. The fact that this eigenvalue is simple is implied by the assumption that the underlying graph is strongly connected.

The normalized Laplacian is $L = I - P$ and the random walk Laplacian is $\mathbf{L} = \boldsymbol{\Pi}(I - P)$. Both of these have rank $n-1$ and nullity 1. The left and right annihilating vectors for L are exactly the left and right eigenvectors of P corresponding to eigenvalue 1, namely π and $\mathbf{1}$, respectively. The annihilating vectors for $\mathbf{L}$ are both $\mathbf{1}$ (on both sides):

$$\mathbf{L} \cdot \mathbf{1} = \mathbf{0} \qquad \mathbf{1}^T \mathbf{L} = \mathbf{0}^T.$$

This makes it particularly easy to write down its pseudoinverse. These properties are summarized in [3].

We use the following lemma based on [3, Lemma 1] regarding the Moore-Penrose inverse of matrices with nullity 1.

Lemma 1. Suppose we have an invertible $(n-1) \times (n-1)$ matrix $L_{\alpha,\alpha}$ and two $(n-1)$-vectors $\mathbf{u}, \mathbf{v}$. Then there is a unique $n \times n$ matrix L with nullity 1 (rank $n-1$) of the form

$$\begin{aligned} L &= \begin{pmatrix} L_{\alpha,\alpha} & \boldsymbol{l}_{\alpha,n} \\ \boldsymbol{l}_{n,\alpha}^T & l_{nn} \end{pmatrix} \\ &= \begin{pmatrix} L_{\alpha,\alpha} & -L_{\alpha,\alpha}\mathbf{v} \\ -\mathbf{u}^T L_{\alpha,\alpha} & \mathbf{u}^T L_{\alpha,\alpha}\mathbf{v} \end{pmatrix} \\ &= \begin{pmatrix} I_{n-1} \\ -\mathbf{u}^T \end{pmatrix} L_{\alpha,\alpha} \begin{pmatrix} I_{n-1}, & -\mathbf{v} \end{pmatrix} \end{aligned} \tag{3}$$

having left and right annihilating vectors satisfying

$$(\mathbf{u}^T, 1)L = 0, \quad L(\mathbf{v}; 1) = 0. \tag{4}$$

Furthermore the Moore-Penrose pseudoinverse $M = L^+$ can be written in terms of $L_{\alpha,\alpha}$, $\mathbf{u}$, $\mathbf{v}$ as follows:

$$\begin{aligned} M &= \begin{pmatrix} M_{\alpha,\alpha} & \boldsymbol{m}_{\alpha,n} \\ \boldsymbol{m}_{n,\alpha}^T & m_{nn} \end{pmatrix} \\ &= \begin{pmatrix} R_{\mathbf{v}} \\ \frac{-1}{1+\mathbf{v}^T\mathbf{v}}\mathbf{v}^T \end{pmatrix} L_{\alpha,\alpha}^{-1} \begin{pmatrix} R_{\mathbf{u}}, & \frac{-1}{1+\mathbf{u}^T\mathbf{u}}\mathbf{u} \end{pmatrix} \end{aligned} \tag{5}$$

(where $M_{\alpha,\alpha}$ is the upper-left $(n-1) \times (n-1)$ block of M) with

$$\begin{aligned} M_{\alpha,\alpha} &= R_{\mathbf{v}} L_{\alpha,\alpha}^{-1} R_{\mathbf{u}} & \boldsymbol{m}_{\alpha,n} &= -M_{\alpha,\alpha}\mathbf{u} \\ \boldsymbol{m}_{n,\alpha}^T &= -\mathbf{v}^T M_{\alpha,\alpha} & m_{nn} &= \mathbf{v}^T M_{\alpha,\alpha}\mathbf{u}, \end{aligned} \tag{6}$$

where $R_{\mathbf{u}} = (I_{n-1} + \mathbf{u}\mathbf{u}^T)^{-1} = (I_{n-1} - \frac{1}{1+\mathbf{u}^T\mathbf{u}}\mathbf{u}\mathbf{u}^T)$, and $R_{\mathbf{v}} = (I_{n-1} + \mathbf{v}\mathbf{v}^T)^{-1} = (I_{n-1} - \frac{1}{1+\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T)$.

Proof (Sketch).

Equation (3) is a simple consequence of (4). Regarding M, it can be verified by direct calculation and some simplification that ML and LM are symmetric, $LML = L$, and $MLM = M$, satisfying the conditions for the Moore-Penrose pseudoinverse. ∎

Corollary 2. [3] Suppose M is an $n \times n$ matrix partitioned as in (6) with the upper $(n-1) \times (n-1)$ block $M_{\alpha,\alpha}$ invertible, and satisfying $(\mathbf{v}^T, 1)M = 0$ and $M(\mathbf{u}; 1) = 0$. Then the upper-left $(n-1) \times (n-1)$ block of $L = M^+$ satisfies

$$\begin{aligned} L_{\alpha,\alpha}^{-1} &= R_{\mathbf{v}}^{-1} M_{\alpha,\alpha} R_{\mathbf{u}}^{-1} \\ &= (I_{n-1} + \mathbf{v}\mathbf{v}^T) M_{\alpha,\alpha} (I_{n-1} + \mathbf{u}\mathbf{u}^T) \\ &= (I_{n-1}, -\mathbf{v}) \begin{pmatrix} M_{\alpha,\alpha} & \boldsymbol{m}_{\alpha,n} \\ \boldsymbol{m}_{n,\alpha}^T & m_{nn} \end{pmatrix} \begin{pmatrix} I_{n-1} \\ -\mathbf{u}^T \end{pmatrix}. \end{aligned} \tag{7}$$

∎

This corollary leads directly to an algorithm to compute the random walk Laplacian and its pseudoinverse using only one matrix inverse.

Algorithm 3 Compute Moore-Penrose pseudoinverse of random walk Laplacian.
Input: Adjacency Matrix A for a strongly connected digraph.
Output: pseudo-inverse $\mathbf{M}$ of random walk Laplacian.

1. Compute probability transition matrix $P = D^{-1}A$, where $D = \text{DIAG}(A \cdot \mathbf{1})$ is the diagonal matrix of out-degrees.
2. Compute normalized Laplacian $L = I - P$ partitioned as in (3).
3. Compute some representation of the inverse of the upper $(n-1) \times (n-1)$ part $L_{\alpha,\alpha}$*.
4. Solve the linear system $(\pi_1, \ldots, \pi_{n-1}) = -L_{\alpha,\alpha}^{-1} \boldsymbol{l}_{\alpha,n} \pi_n$, where π_n is scaled after the fact to satisfy $\|\boldsymbol{\pi}\|_1 = 1$.
5. Form diagonal matrix $\mathbf{\Pi} = \text{DIAG}(\boldsymbol{\pi})$ and random walk Laplacian $\mathbf{L} = \mathbf{\Pi} \cdot L = \mathbf{\Pi}(I - P)$, itself partitioned as in (3).
6. Compute the inverse of the upper-left block of $\mathbf{L}$, namely $\mathbf{L}_{\alpha,\alpha}^{-1} = (I - P_{\alpha,\alpha})^{-1}\mathbf{\Pi}_1^{-1}$ using the previously computed inverse, where $\mathbf{\Pi}_1 = \text{DIAG}(\pi_1, \ldots, \pi_{n-1})$.
   - Elementwise: $[\mathbf{L}_{\alpha,\alpha}^{-1}]_{ij} = [L_{\alpha,\alpha}^{-1}]_{ij}/\pi_j$.
7. Compute desired pseudoinverse $\mathbf{M}$ according to Lemma 1:

$$\begin{aligned} \mathbf{M} &= \begin{pmatrix} \mathbf{M}_{\alpha,\alpha} & \mathbf{m}_{\alpha,n} \\ \mathbf{m}_{n,\alpha}^T & \mathrm{m}_{nn} \end{pmatrix} \\ &= \begin{pmatrix} R_{\mathbf{1}} \\ \frac{-1}{n}\mathbf{1}^T \end{pmatrix} \mathbf{L}_{\alpha,\alpha}^{-1} \left(R_{\mathbf{1}},\ \tfrac{-1}{n}\mathbf{1} \right), \end{aligned} \tag{8}$$

   where we have used the fact that the left and right annihilating vectors for $\mathbf{L}$ and $\mathbf{M}$ are both $\mathbf{1}_n$, so the vectors in (4) are $\mathbf{u} = \mathbf{v} = \mathbf{1}_{n-1}$. We also have the identity $R_{\mathbf{1}}\mathbf{1} = (I_{n-1} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{1} = \frac{1}{n}\mathbf{1}$.
   This can be computed step-by-step as follows:
   a. Compute $\mathbf{b} = {}^1\!/_n \mathbf{L}_{\alpha,\alpha}^{-1}\mathbf{1}$ and $\mathbf{c}^T = {}^1\!/_n \mathbf{1}^T \mathbf{L}_{\alpha,\alpha}^{-1}$.
   b. Compute $\mathbf{M}_{\alpha,\alpha} = \mathbf{L}_{\alpha,\alpha}^{-1} - \mathbf{b}\cdot\mathbf{1}^T - \mathbf{1}\cdot\mathbf{c}^T + \frac{\mathbf{c}^T\mathbf{1}}{n}\mathbf{1}\mathbf{1}^T$, or elementwise $[\mathbf{M}_{\alpha,\alpha}]_{ij} = [\mathbf{L}_{\alpha,\alpha}^{-1}]_{ij} - [\mathbf{b}]_i - [\mathbf{c}]_j + \frac{\mathbf{c}^T\mathbf{1}}{n}$.
   c. Compute $\mathbf{m}_{\alpha,n} = \mathbf{b} - \frac{\mathbf{1}^T\mathbf{b}}{n}\mathbf{1}$, $\mathbf{m}_{n,\alpha}^T = \mathbf{c}^T - \frac{\mathbf{c}^T\mathbf{1}}{n}\mathbf{1}^T$, $\mathrm{m}_{nn} = \frac{\mathbf{c}^T\mathbf{1}}{n} = \frac{\mathbf{1}^T\mathbf{b}}{n}$.

We remark that in the case of undirected graphs, $\boldsymbol{\pi}$ is a multiple of the vector of vertex degrees, hence step 2 would be free, but we would still need the inverse in step 5.

3 Fundamental Tensor

Fundamental Matrix. If we set $\mathbf{s}^T = 0, \tau = 1$ in (2), we turn the recurring Markov chain into an absorbing Markov chain with corresponding probability transition matrix $\widetilde{P}$, where the last node $k = n$ is a single absorbing node. In this case the *fundamental matrix* N [13] is the matrix

$$\boldsymbol{N}(\alpha, \alpha, n) = L_{\alpha,\alpha}^{-1} = (I - P_{\alpha,\alpha})^{-1}, \tag{9}$$

whose ij-th entry is the average or expected number of passages through node j in random walks starting from node i which are absorbed by node k.

Fundamental Tensor. We would like to compute the *fundamental tensor* $\boldsymbol{N} = \{\boldsymbol{N}(i,j,k)\}_{i,j,k=1,\ldots,n}$ where $\boldsymbol{N}(i,j,k)$ is the average number of passages through intermediate node j for random walks starting from i before reaching k. This would be the fundamental matrix in a Markov chain in which node k is made absorbing instead of node n. In Algorithm 3 we have shown how to compute the pseudoinverse of the random walk Laplacian from the fundamental matrix $N = \boldsymbol{N}(\alpha, \alpha, n)$ obtained when node n is made absorbing. We would now like to do the reverse: obtain the fundamental matrix N from the pseudoinverse $\mathbf{M}$ of the random walk Laplacian without any more matrix inversions. Since the choice of absorbing node n is arbitrary, this method will also yield a method to compute the fundamental matrix $\boldsymbol{N}(:,:,k)$ obtained when any other node $k \neq n$ is made absorbing. In this way we can fill in the entire fundamental tensor.

Generalization of Fundamental Tensor. In later sections below, we consider $N_{\beta,\gamma} = \boldsymbol{N}(\beta, \gamma, n)$, namely the submatrix of $N = \boldsymbol{N}(\alpha, \alpha, n)$ consisting of rows indexed by index set β and columns by index set γ. We also later discuss other tensors of visit counts such as (with $i, j \in \beta$)

(a) $\boldsymbol{N}(i, j, \{\gamma, n\})$ = expected number of visits to j before reaching any node in $\{\gamma, n\}$, starting from i
(b) $\boldsymbol{N}(i, j, n, \gamma)$ = expected number of visits to j before reaching n starting from i, counting only walks that avoid γ

Corollary 2 provides the formula to go from $\mathbf{M}$ to $N = \boldsymbol{N}(\alpha, \alpha, n)$. From (6), it follows that

$$N = (I - P_{\alpha,\alpha})^{-1} = L_{\alpha,\alpha}^{-1} = \mathbf{L}_{\alpha,\alpha}^{-1}\mathbf{\Pi}_1 = R_{\mathbf{1}}^{-1}\mathbf{M}_{\alpha,\alpha}R_{\mathbf{1}}^{-1}\mathbf{\Pi}_1$$

A little algebra yields

$$\begin{aligned} N &= (I + \mathbf{1}\mathbf{1}^T)\mathbf{M}_{\alpha,\alpha}(I + \mathbf{1}\mathbf{1}^T)\mathbf{\Pi}_1 \\ &= (I, -\mathbf{1}) \begin{pmatrix} \mathbf{M}_{\alpha,\alpha} & \mathbf{m}_{\alpha,n} \\ \mathbf{m}_{n,\alpha}^T & \boldsymbol{m}_{nn} \end{pmatrix} \begin{pmatrix} I \\ -\mathbf{1}^T \end{pmatrix} \mathbf{\Pi}_1 \\ &= (I, -\mathbf{1})\,\mathbf{M} \begin{pmatrix} I \\ -\mathbf{1}^T \end{pmatrix} \mathbf{\Pi}_1, \end{aligned}$$

or elementwise $N(i,j) = (\boldsymbol{m}_{ij} - \boldsymbol{m}_{nj} - \boldsymbol{m}_{in} + \boldsymbol{m}_{nn})\pi_j$ for $1 \leq i, j \leq n-1$. This yields the values for the ij-th entries in the n-th slice of the tensor, $\boldsymbol{N}(i,j,n)$, including the values when $i = n$ or $j = n$. We are free to reorder the nodes because the left and right annihilating vectors of $\mathbf{L}$ remain the same regardless of the order (namely, the vector of all ones). By reordering the nodes, we obtain the corresponding formula for every other slice in terms of the entries of $\mathbf{M}$:

$$\boldsymbol{N}(i,j,k) = (\boldsymbol{m}_{ij} - \boldsymbol{m}_{kj} - \boldsymbol{m}_{ik} + \boldsymbol{m}_{kk})\pi_j \tag{10}$$

Cost. Using Algorithm 3, we can compute the pseudo inverse of the Laplacian in $O(n^2)$ time plus the cost of one matrix inverse. The entire remaining part of the fundamental tensor $\boldsymbol{N}$ can then be computed by formula (10) in constant time per entry $\boldsymbol{N}(i,j,k)$. The cost of the matrix inverse is $O(n^3)$

*The inverse could be represented in terms of its LU factors, but later we need the entries of the inverse itself.

number of			time in csec	
vertices	edges	LU fill	LU	backsolve
1,024	4,059	20,620	5	2
2,048	8,140	66,851	2	< 1
4,096	16,314	205,826	4	< 1
8,192	32,671	763,440	12	1
16,384	65,402	2,804,208	56	5
32,768	130,884	10,740,194	250	19
65,536	261,882	43,504,911	1,363	82
131,072	523,920	168,455,437	7,989	328

Table 1: Cost of Gaussian elimination for a sample of synthetic scale-free graphs using Matlab R2018a.

using the usual off-the-shelf algorithm [12]. However, it has been shown in [6] that the random walk Laplacian L satisfies the property that $\frac{1}{2}(L + L^T)$ can be considered as the Laplacian for an undirected graph with the same link structure as the original directed graph. Cohen et al. [7] use this property to find an approximate sparse LU factorization for L whose fill (number of non-zero entries) is linear in the fill of the original L, with high probability. This leads to a stochastic preconditioner for an iterative method to approximately solve linear systems involving Laplacians for strongly connected graphs. The end result is a stochastic process to find an approximation to the inverse of $L_{\alpha,\alpha}$ with theoretical complexity a little over $O(n^2)$, depending on the the conditioning of the system and the desired accuracy. Certain graph structures naturally lead to fast deterministic algorithms. We illustrate this with so called scale-free graphs enjoying the small-world property, commonly found in many real-world social networks [1]. Small-world networks are characterized by a high clustering coefficient but have a small expected path length. Hence, the expected path length between pairs of nodes is small relative to the number of nodes in the network. We generate synthetic scale-free graphs using preferential attachment (Albert-Barabasi model [1]) of varying sizes, and construct a directed graph by randomly deleting some edges. Table 1 shows the results using a deterministic LU factorization with the approximate minimum degree ordering [8]. For these simple examples, the process takes only $O(n^2)$ time and space, leading to faster generation of the inverse matrix. Using iterative methods and approximate factorizations could further reduce this cost with high probability [7].

To obtain the generalizations $\boldsymbol{N}(\beta, \beta, n, \gamma)$, it will be seen later that we need to compute $L_{\beta\beta}^{-1} = (I - P_{\beta\beta})^{-1}$. The following holds for any invertible $(n-1) \times (n-1)$ matrix $L_{\alpha,\alpha} = N^{-1}$ whose principal submatrix $L_{\beta,\beta}$ is also invertible. Suppose the rows/columns are ordered WLOG[†] with $\beta = \{1, \ldots, n_1\}$, $\gamma = \{n_1 + 1, \ldots, n-1\}$. Then

$$L_{\alpha,\alpha} \cdot N = \begin{pmatrix} L_{\beta,\beta} & L_{\beta,\gamma} \\ L_{\gamma,\beta} & L_{\gamma,\gamma} \end{pmatrix} \cdot \begin{pmatrix} N_{\beta,\beta} & N_{\beta,\gamma} \\ N_{\gamma,\beta} & N_{\gamma,\gamma} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}.$$

[†] without loss of generality

From this relation we see that

$$\begin{array}{lrcl} \text{(a)} & L_{\beta,\beta}N_{\beta,\gamma} + L_{\beta,\gamma}N_{\gamma,\gamma} & = & 0, \\ \text{(b)} & L_{\beta,\beta}N_{\beta,\beta} + L_{\beta,\gamma}N_{\gamma,\beta} & = & I_{\beta,\beta}. \end{array} \tag{11}$$

Then we can verify the formula for the inverse of $L_{\beta,\beta}$ as follows, assuming the indicated inverses exist:

$$\begin{array}{rcl} L_{\beta,\beta}\left[N_{\beta,\beta} - N_{\beta,\gamma}N_{\gamma,\gamma}^{-1}N_{\gamma,\beta}\right] & & \\ = \; I_{\beta,\beta} - L_{\beta,\gamma}N_{\gamma,\beta} - L_{\beta,\beta}N_{\beta,\gamma}N_{\gamma,\gamma}^{-1}N_{\gamma,\beta} & & \\ = \; I_{\beta,\beta} - L_{\beta,\gamma}N_{\gamma,\beta} + L_{\beta,\gamma}N_{\gamma,\gamma}N_{\gamma,\gamma}^{-1}N_{\gamma,\beta} & & \\ = \; I_{\beta,\beta} & & \end{array} \tag{12}$$

We have (where (13b) follows from (11b)):

$$\begin{array}{lrcl} \text{(a)} & L_{\beta,\beta}^{-1} & = & N_{\beta,\beta} - N_{\beta,\gamma}N_{\gamma,\gamma}^{-1}N_{\gamma,\beta} \\ \text{(b)} & & = & N_{\beta,\beta} + L_{\beta,\beta}^{-1}L_{\beta,\gamma}N_{\gamma,\beta} \end{array} \tag{13}$$

4 Applications

Hitting Time. The expected time for a random walk starting at source i to reach target k is

$$H(i,k) = \sum_j \boldsymbol{N}(i,j,k) = \mathrm{m}_{kk} - \mathrm{m}_{ik} + \sum_j (\mathrm{m}_{ij} - \mathrm{m}_{kj})\pi_j.$$

The expected round-trip commute time between nodes i and k is

$$\begin{aligned} C(i,k) &= H(i,k) + H(k,i) = \textstyle\sum_j (\boldsymbol{N}(i,j,k) + \boldsymbol{N}(k,j,i)) \\ &= \mathrm{m}_{kk} + \mathrm{m}_{ii} - \mathrm{m}_{ik} - \mathrm{m}_{ki}. \end{aligned}$$

These can easily be computed directly from the Laplacian (see e.g. [3] and references therein).

Centrality Measures. It is easy to compute a variety of centrality measures such as the average round-trip commute time from a given node k to all other nodes

$$\begin{aligned} C(k) &= \textstyle\sum_i C(i,k)/n = \sum_{i,j} (\boldsymbol{N}(i,j,k) + \boldsymbol{N}(k,j,i))/n \\ &= (\mathrm{m}_{kk} + \textsc{Trace}(\mathbf{M}))/n, \end{aligned}$$

where we have used $\textsc{Trace}(\mathbf{M}) = \sum_i \mathrm{m}_{ii}$ and the fact that $\sum_i \mathrm{m}_{ki} = \sum_i \mathrm{m}_{ik} = 0$. Another measure of the importance of individual nodes in terms of bottleneck or influence is random walk closeness [Noh and Rieger, 2004]:

$$\text{closeness}(k) = \sum_i H(i,k) = \sum_{i,j} \boldsymbol{N}(i,j,k).$$

The fundamental tensor can also be used to compute betweenness measures such as random walk betweenness [19; 16]:

$$\text{betweenness}(j) = \sum_{i \neq j, k \neq j} \Pr(i \to j \to k),$$

where $\Pr(i \to j \to k)$ denotes the probability of passing j before reaching k in a random walk starting at i.

Probability of passage before another node. The fundamental tensor can be used to obtain the probability of passing through a node j before reaching a node k, denoted $\Pr(i \to j \to k)$. This is the key concept behind the personalized hitting time trust mechanism [5]. We have the following result.

Theorem 4. (Probability of ordered passage) The probability of passing through j on a random walk through the entire network starting from node i and before reaching k is

$$\Pr(i \to j \to k) = \boldsymbol{N}(i,j,k) \,/\, \boldsymbol{N}(j,j,k).$$

Proof. We show WLOG $\Pr(i \to (n-1) \to n) = \boldsymbol{N}(i, n-1, n) \,/\, \boldsymbol{N}(n-1, n-1, n)$. Consider a recurring Markov chain with transition probabilities P (an $n \times n$ matrix), partitioned as

$$P = \left[\begin{array}{cc|c} Q & \mathbf{r}_1 & \mathbf{r}_2 \\ \mathbf{s}_1^T & t_{11} & t_{12} \\ \hline \mathbf{s}_2^T & t_{21} & t_{22} \end{array}\right],$$

where Q is $(n-2)\times(n-2)$, $\mathbf{r}_1, \mathbf{r}_2$ are column $(n-2)$-vectors, and $\mathbf{s}_1^T, \mathbf{s}_2^T$ are row $(n-2)$-vectors.

To count how many times we pass through node $n-1$ before reaching n, turn node n into an absorbing node and follow the prescription in [13]. Set $\mathbf{s}_2 = \mathbf{0}$, $t_{21} = 0$, $t_{22} = 1$, and follow [13] to find the average number of visits to any node j starting from i via the fundamental matrix:

$$N = \begin{bmatrix} (I-Q) & -\mathbf{r}_1 \\ -\mathbf{s}_1^T & 1 - t_{11} \end{bmatrix}^{-1} = \begin{bmatrix} W & \mathbf{x} \\ \mathbf{y}^T & z \end{bmatrix}, \tag{14}$$

partitioned conformally (so z is a scalar). In particular, x_i is the expected number of passages through node $n-1$ before reaching node n when starting from node i, and z is the expected number of passages through node $n-1$ when starting from node $n-1$. So we have $x_i = \mathbf{N}(i, n-1, n)$ and $z = \mathbf{N}(n-1, n-1, n)$.

From

$$\begin{bmatrix} (I-Q) & -\mathbf{r}_1 \\ -\mathbf{s}_1^T & 1 - t_{11} \end{bmatrix} \cdot \begin{bmatrix} W & \mathbf{x} \\ \mathbf{y}^T & z \end{bmatrix} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

we have $(I-Q)\mathbf{x} - z\mathbf{r}_1 = \mathbf{0}$, or

$$\mathbf{x} = z(I-Q)^{-1}\mathbf{r}_1. \tag{15}$$

Now set $\mathbf{s}_1 = \mathbf{0}$, $t_{11} = 1$, $t_{12} = 0$, turning both $n-1, n$ into absorbing states, and follow [13] to find the vector of probabilities of reaching $n-1$ before n (starting from any node $i = 1, \ldots, n-2$) as $\mathbf{b}_1 = (I-Q)^{-1}\mathbf{r}_1$, which from (15) is just $\mathbf{x}/z$. ∎

Avoiding nodes. We now consider the average hitting time to a given node assuming we avoid a certain subset of nodes. Specifically, we divide the set of nodes into three subsets,

$$\beta = \{1, \ldots, n_1\}, \quad \gamma = \{n_1 + 1, \ldots, n-1\}, \quad \{n\},$$

such that $\alpha = \{\beta, \gamma\} = \{1, \ldots, n-1\}$ in (2). We consider the case where, starting from some node i in β, we would like to obtain the average hitting time to node n, *conditioned* on avoiding any node in γ. We denote the expected number of passages through an individual node j, starting at i, before reaching n, and avoiding γ as $\boldsymbol{N}(i,j,n,\gamma)$. Then this desired conditional average hitting time is $H(i,n,\gamma) = \sum_{j\in\beta} \boldsymbol{N}(i,j,n,\gamma)$. Letting avoiding nodes γ vary over the major nodes in the network could be a way to satisfy a query seeking major bottlenecks. Assume WLOG that the nodes are ordered so that the probability transition matrix is partitioned as

$$P = \left(\begin{array}{c|c} P_{\alpha,\alpha} & P_{\alpha,n} \\ \hline P_{n,\alpha} & P_{nn} \end{array}\right) = \left(\begin{array}{cc|c} P_{\beta,\beta} & P_{\beta,\gamma} & P_{\beta,n} \\ P_{\gamma,\beta} & P_{\gamma,\gamma} & P_{\gamma,n} \\ \hline P_{n,\beta} & P_{n,\gamma} & P_{nn} \end{array}\right).$$

If node n is temporarily treated as an absorbing node, we obtain the $(n-1)\times(n-1)$ matrix of visit counts to any individual node in α before reaching node n shown in eq. (9). In general, we use the following result from [13, chap. 11]:

Proposition 5. For $i, j \in \beta$, $\ell \in \{\gamma, n\}$:

$$\begin{aligned} [(I - P_{\beta,\beta})^{-1}]_{ij} &= \boldsymbol{N}(i,j,\{\gamma,n\}) \\ &= \text{expected number of passages through } j \text{ starting from } i \\ &\quad \text{before leaving } \beta \\ [(I - P_{\beta,\beta})^{-1} \cdot P_{\beta,:}]_{i\ell} &= \Pr(i \to \ell \to [\{\gamma,n\} - \{\ell\}]) \\ &= \Pr(\ell \text{ is first node reached outside of } \beta, \text{ starting from } i) \end{aligned}$$

Proposition 6. For any $j \in \beta$ such that n is reachable from j without passing through γ, the average number of return visits to the starting node j before reaching n and avoiding γ is

$$\boldsymbol{N}(j,j,n,\gamma) = \boldsymbol{N}(j,j,\{\gamma,n\}) = [(I - P_{\beta,\beta})^{-1}]_{jj}. \tag{16}$$

That is, when counting return visits, the condition to avoid certain nodes makes no difference, compared to making them absorbing.

Proof. Starting from j, consider the three mutually exclusive events with corresponding probabilities

- $p = \Pr(\text{pass } j \text{ before reaching } \gamma, n)$,
- $q = \Pr(\text{reach } \gamma \text{ before reaching } j, n)$,
- $r = \Pr(\text{pass } n \text{ before reaching } j, \gamma)$,

with $p + q + r = 1$. We have the following probabilities, starting from j and wandering among nodes in β only:

- $a := \Pr(\text{pass } j \text{ exactly } k \text{ times, then reach } n \text{ before } \gamma)$
  $= p^k r$
- $b := \Pr(\text{eventually reach } n \text{ before } \gamma)$
  $= r/(q+r) = r/(1-p) = r\sum_{k=0}^{\infty} p^k$
- $c := \Pr(\text{pass } j \text{ exactly } k \text{ times, then reach } n \text{ or } \gamma)$
  $= p^k(1-p)$

Hence the probability of passing j exactly k times *conditioned* on eventually reaching n is

$$\Pr(\text{pass } j \text{ exactly } k \text{ times, given reach } n \text{ before } \gamma) = a/b = c$$

Since the probability distributions over k match, their expected values match. ∎

Proposition 7. For any $i, j \in \beta$ such that n is reachable from i without passing through γ,

$$\boldsymbol{N}(i,j,n,\gamma) = \Pr(i \to j \to n \text{ avoiding } \gamma) \cdot \boldsymbol{N}(j,j,\{\gamma,n\}). \tag{17}$$

The quantity $\Pr(i \to j \to n \text{ avoiding } \gamma)$ is the probability that a random walk starting from i will visit intermediate node j before reaching n, conditioned on the random walk never visiting any node in γ. This proposition says that the expected number of passages through j is equal to the probability of reaching j initially times the expected number

of return visits to j once it is reached. We have the following for the probability of reaching j initially:

Proposition 8. For any $i, j \in \beta$ such that n is reachable from i without passing through γ,

$$\begin{aligned}&\Pr(i \to j \to n \text{ avoiding } \gamma)\\ &\quad = \frac{\Pr(i \to j \to \{\gamma, n\}) \cdot \Pr(j \to n \to \{\gamma\})}{\Pr(i \to n \to \{\gamma\})}. \end{aligned} \tag{18}$$

Proof (Sketch). To derive (18), we temporarily treat all the nodes in $\gamma, \{n\}$ as absorbing states. The numerator of the fraction in (18) is the joint probability that starting from i, the random walker passes j before absorption into any of $\gamma, \{n\}$, and once at j the walker is absorbed by n as opposed to any node in γ. The denominator is the probability that starting from i, the walker is absorbed by n as opposed to any node in γ. ∎

We plug in the formulas from Proposition 5 into (18) to get the probability of passing j in terms of $(I - P_{\beta,\beta})^{-1}$:

Proposition 9. For any $i, j \in \beta$,

$$\begin{aligned}&\Pr(i \to j \to n \text{ avoiding } \gamma) =\\ &\quad = \Pr(i \to j \to \{\gamma, n\}) \cdot \frac{\Pr(j \to n \to \{\gamma\})}{\Pr(i \to n \to \{\gamma\})}\\ &\quad = \frac{[(I - P_{\beta,\beta})^{-1}]_{ij}}{[(I - P_{\beta,\beta})^{-1}]_{jj}} \cdot \frac{[(I - P_{\beta,\beta})^{-1} P_{\beta,:}]_{jn}}{[(I - P_{\beta,\beta})^{-1} P_{\beta,:}]_{in}} \end{aligned} \tag{19}$$

Proposition 7 gives a formula for the average counts conditioned on avoiding certain set of nodes. If we plug in formulas from Propositions 6 and 9, we see that all the terms can be written in terms of $(I - P_{\beta,\beta})^{-1}$ and closely related quantities. If we wish to compute these quantities for a variety of β's and γ's representing various subsets of vertices in the network, it is useful to know how to quickly obtain $(I - P_{\beta,\beta})^{-1}$ from the original Fundamental Tensor $\boldsymbol{N}(i, j, k)$ (10). In the following, we assume WLOG that the nodes are ordered so that $k = n$.

Proposition 10.

$$\begin{aligned}&(I - P_{\beta,\beta})^{-1} = \boldsymbol{N}(\beta, \beta, \{\gamma, n\})\\ &= [(I - P_{\alpha,\alpha})^{-1}]_{\beta,\beta} && \textbf{[A]}\\ &\quad - [(I - P_{\alpha,\alpha})^{-1}]_{\beta,\gamma} \cdot [[(I - P_{\alpha,\alpha})^{-1}]_{\gamma,\gamma}]^{-1} && \textbf{[B]}\\ &\quad \cdot [(I - P_{\alpha,\alpha})^{-1}]_{\gamma,\beta} && \textbf{[C]}\\ &= \underbrace{\boldsymbol{N}(\beta, \beta, n)}_{\textbf{[A]}} - \underbrace{\boldsymbol{N}(\beta, \gamma, n)\boldsymbol{N}(\gamma, \gamma, n)^{-1}}_{\textbf{[B]}} \underbrace{\boldsymbol{N}(\gamma, \beta, n)}_{\textbf{[C]}}\\ &= [(I - P_{\alpha,\alpha})^{-1}]_{\beta,\beta} && \textbf{[A]}\\ &\quad - (I - P_{\beta,\beta})^{-1}(P_{\beta,\gamma}) && \textbf{[B]}\\ &\quad \cdot [(I - P_{\alpha,\alpha})^{-1}]_{\gamma,\beta} && \textbf{[C]}\end{aligned}$$

where the individual entries in the last expression represent

- $\textbf{[A]}_{ij}$ = expected number of passages through node $j \in \beta$ starting at $i \in \beta$
- $\textbf{[B]}_{i\ell}$ = $\Pr(\ell \in \gamma$ is the first node visited outside of β, starting at $i \in \beta)$
- $\textbf{[C]}_{\ell j}$ = expected number of passages through node $j \in \beta$ starting at $l \in \gamma$

In all three cases the walks stop when they reach n. Here the notation $\boldsymbol{N}(\beta, \gamma, n)$ denotes the sub-matrix of $\boldsymbol{N}(:, :, n)$ consisting of the rows indexed by β and columns indexed by γ.

Proof (Sketch). The first equality is just a rewrite of (13a) and the last equality is a rewrite of (13b). The interpretation of $\textbf{[B]}_{i\ell}$ as the indicated probability comes from Proposition 5 and (13b). ∎

In the above formulas, if the avoidance set γ is relatively small, then the cost of the above formulas will be modest, at most quadratic in n once $(I - P_{\alpha,\alpha})^{-1}$ has been obtained.

5 Examples

Illustrative Example. We give a small example with four nodes.

We define a visit to a node j as one departure from that node. Hence the visit counts for target nodes are zero: $\boldsymbol{N}(i, j, k) = 0$ for $i = k$ or $j = k$. In this case the probability transition matrix and normalized Laplacian are

$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & -1/2 & -1/2 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{pmatrix},$$

$$L^+ = \frac{1}{28} \begin{pmatrix} 8 & -3 & -3 & -10 \\ 0 & 21 & -7 & -14 \\ -8 & -11 & 17 & 10 \\ 0 & -7 & -7 & 14 \end{pmatrix},$$

where the recurring probabilities are $\boldsymbol{\pi} = (0.4;\ 0.2;\ 0.2;\ 0.2)$. The computed tensor is then

$$\boldsymbol{N}_{::1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{N}_{::2} = \begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 2 \\ 2 & 0 & 1 & 2Algorithm \end{pmatrix}$$

$$\boldsymbol{N}_{::3} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{N}_{::4} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Trust Mechanism. We consider the personalized hitting time (PHT) trust mechanism, which is one way to model the propagation of trust through a network of actors [17; 5]. Consider the network of Fig. 1 with probability transition matrix

$$P = \begin{pmatrix} 0 & 0.340 & 0 & 0.510 & 0 & 0.150 \\ 0.170 & 0 & 0.425 & 0.255 & 0 & 0.150 \\ 0 & 0 & 0 & 0 & 0.850 & 0.150 \\ 0.425 & 0 & 0 & 0 & 0.425 & 0.150 \\ 0.170 & 0 & 0.680 & 0 & 0 & 0.150 \\ 0.200 & 0.200 & 0.200 & 0.200 & 0.200 & 0 \end{pmatrix} \tag{20}$$

Here we seek to infer trust values for every node from the viewpoint of every other node, where v_6 is an outside evaporation node to enforce proximity in the trust values (See [17; 5] for details). The personalized hitting time trust mechanism value is defined as

$$\mathbf{PHT}(i, j) = \Pr(i \to j \to 6).$$

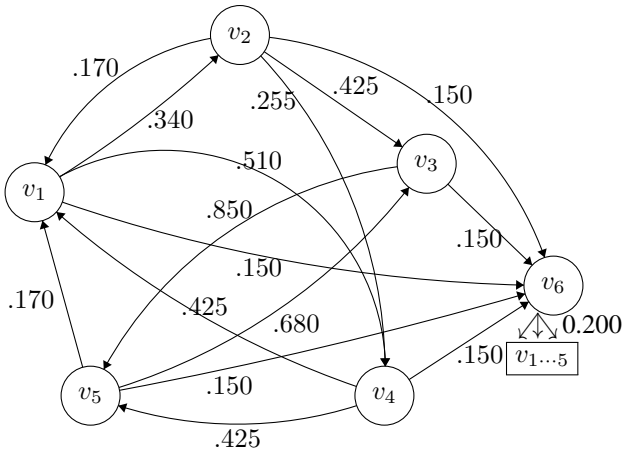

Figure 1: Sample network for trust mechanism model. Node v_6 is used to model the global evaporation node.

According to Theorem 4, this can be computed as $\mathbf{PHT}(i,j) = \boldsymbol{N}(i,j,6) \,/\, \boldsymbol{N}(j,j,6)$. For example, from the viewpoint of node 4, the trust values for nodes $1,\ldots,5$ are

$$\mathbf{PHT}(4,1:5) = (0.5962,\ 0.2913,\ 0.5332,\ 1.0,\ 0.6573),$$

where node 5 enjoys the highest trust value from the point of view of node 4. If we instead consider only paths that avoid node 2 (for instance, node 2 is known to be a bad actor), we can compute the PHT values based on the restricted matrix of counts $\boldsymbol{N}(:,:,6,2)$ instead of $\boldsymbol{N}(:,:,6)$, to obtain the modified trust values

$$\mathbf{PHT}_2(4,1:5) = (0.5962,\ 0.0000,\ 0.3872,\ 1.0,\ 0.5426),$$

where it is seen that the node with the highest trust value has changed to node 1.

6 Conclusions

We have sketched a streamlined algorithm to compute a fundamental tensor for a strongly connected directed graph. This tensor encodes the number of visits to each node given a particular starting and ending node in a directed graph, based on the random walk model. We have illustrated how this tensor can be used to quickly compute many importance measures and related queries for the graph and for small modifications of the graph if a few nodes are corrupted or removed. The algorithm has concentrated the parts of high complexity into a single matrix inverse of a principal submatrix of the graph Laplacian. This means a fast algorithm for this one piece of higher complexity would yield a low-complexity algorithm for all the subsequent applications. Though a fast exact algorithm for the general matrix inverse does not exist, we have indicated that an approximate algorithm which takes advantage of the special structure of the Laplacian is possible.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[2] Marc Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B*, 38:163–168, 2004.

[3] Daniel Boley, Gyan Ranjan, and Zhi-Li Zhang. Commute times for a directed graph using an asymmetric Laplacian. *Lin. Alg. & Appl.*, 435:224–242, 2011.

[4] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25:163–177, 2001.

[5] Alejandro Buendia and Daniel Boley. Optimized graph-based trust mechanisms using hitting times. In *AAMAS Intl Workshop on Trust in Agent Societies*, May 2017.

[6] Michael B. Cohen, Jon Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *IEEE 57th Annual Symp. on Found. Comput. Sci. (FOCS)*, pages 583–592, Oct 2016.

[7] Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. arxiv.org/abs/1811.10722, 2018.

[8] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, September 2004.

[9] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[10] Linton Freeman, Stephen Borgatti, and Douglas White. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13:141–154, 1991.

[11] Golshan Golnari, Zhi-Li Zhang, and Daniel Boley. Markov fundamental tensor and its applications to network analysis. *Linear Algebra and Appl.*, 564:126–158, 2019.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 4th edition, 2013.

[13] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 2012.

[14] Chung-Wei Hang and Munindar P. Singh. Trust-based recommendation based on graph similarity. In *Proceedings of the 13th AAMAS Workshop on Trust in Agent Societies (Trust)*. AAMAS, 2010.

[15] John Hopcroft and Daniel Sheldon. Manipulation-resistant reputations using hitting time. In *Algorithms and Models for the Web-Graph*. WAW, 2006.

[16] U. Kang, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong. Centralities in large networks: Algorithms and observations. In *SIAM Intl Conf Data Mining*, 2011.

[17] Brandon Liu, David Parkes, and Sven Seuken. Personalized hitting time for informative trust mechanisms despite sybils. In *Int'l Conf. Auto. Agents & Multiagent Sys. (AAMAS)*, 2016.

[18] Charalampos Mavroforakis, Michael Mathioudakis, and Aristides Gionis. Absorbing random-walk centrality. arxiv.org/abs/1509.02533, 2015.

[19] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social Net.*, 27(1):39–54, 2005.

[20] Jae Dong Noh and Heiko Rieger. Random walks on complex networks. *Physical Review Letters*, 92(11), 2004.

[21] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1–37), 1989.