Improving Information Centrality of a Node in Complex Networks by Adding Edges

Liren Shan[1,2], **Yuhao Yi**[1,2] and **Zhongzhi Zhang**[1,2,*]
[1]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, China
[2]School of Computer Science, Fudan University, China
{13307130150, yhyi15, zhangzz}@fudan.edu.cn

Abstract

The problem of increasing the centrality of a network node arises in many practical applications. In this paper, we study the optimization problem of maximizing the information centrality I_v of a given node v in a network with n nodes and m edges, by creating k new edges incident to v. Since I_v is the reciprocal of the sum of resistance distance $\mathcal{R}_v$ between v and all nodes, we alternatively consider the problem of minimizing $\mathcal{R}_v$ by adding k new edges linked to v. We show that the objective function is monotone and supermodular. We provide a simple greedy algorithm with an approximation factor $\left(1-\frac{1}{e}\right)$ and $O(n^3)$ running time. To speed up the computation, we also present an algorithm to compute $\left(1-\frac{1}{e}-\epsilon\right)$-approximate resistance distance $\mathcal{R}_v$ after iteratively adding k edges, the running time of which is $\widetilde{O}(mk\epsilon^{-2})$ for any $\epsilon > 0$, where the $\widetilde{O}(\cdot)$ notation suppresses the $\mathrm{poly}(\log n)$ factors. We experimentally demonstrate the effectiveness and efficiency of our proposed algorithms.

1 Introduction

Centrality metrics refer to indicators identifying the varying importance of nodes in complex networks [Lü *et al.*, 2016], which have become a powerful tool in network analysis and found wide applications in network science [Newman, 2010]. Over the past years, a great number of centrality indices and corresponding algorithms have been proposed to analyze and understand the roles of nodes in networks [White and Smyth, 2003; Boldi and Vigna, 2014]. Among various centrality indices, betweennees centrality and closeness centrality are probably the two most frequently used ones, especially in social network analysis. However, both indicators only consider the shortest paths, excluding the contributions from other longer paths. In order to overcome the drawback of these two measures, current flow closeness centrality [Brandes and Fleischer, 2005; Newman, 2005] was introduced and proved to be exactly the information centrality [Stephenson and Zelen, 1989], which counts all possible paths between nodes and has a better discriminating power than betweennees centrality [Newman, 2005] and closeness centrality [Bergamini *et al.*, 2016].

It is recognized that centrality measures have proved of great significance in complex networks. Having high centrality can have positive consequences on the node itself. In this paper, we consider the problem of adding a given number of edges incident to a designated node v so as to maximize the centrality of v. Our main motivation or justification for studying this problem is that it has several application scenarios, including airport networks [Ishakian *et al.*, 2012], recommendation systems [Parotsidis *et al.*, 2016], among others. For example, in airport networks, a node (airport) has the incentive to improve as much as possible its centrality (transportation capacity) by adding edges (directing flights) connecting itself and other nodes (airports) [Ishakian *et al.*, 2012]. Another example is the link recommendation problem of recommending to a user v a given number of links from a set of candidate inexistent links incident to v in order to minimize the shortest distance from v to other nodes [Parotsidis *et al.*, 2016].

The problem of maximizing the centrality of a specific target node through adding edges incident to it has been widely studied. For examples, some authors have studied the problem of creating k edges linked to a node v so that the centrality value for v with respect to concerned centrality measures is maximized, e.g., betweenness centrality [Crescenzi *et al.*, 2015; D'Angelo *et al.*, 2016; Crescenzi *et al.*, 2016; Hoffmann *et al.*, 2018] and closeness centrality [Crescenzi *et al.*, 2015; Hoffmann *et al.*, 2018]. Similar optimization problems for a predefined node v were also addressed for other node centrality metrics, including average shortest distance between v and remaining nodes [Meyerson and Tagiku, 2009; Parotsidis *et al.*, 2016], largest distance from v to other nodes [Demaine and Zadimoghaddam, 2010], PageRank [Avrachenkov and Litvak, 2006; Olsen, 2010], and the number of different paths containing v [Ishakian *et al.*, 2012]. However, previous works do not consider improving information centrality of a node by adding new edges linked to it, despite the fact that it can better distinguish different nodes, compared with betweennees [Newman, 2005] and closeness centrality [Bergamini *et al.*, 2016].

[*]Corresponding author. This work was supported by the National Natural Science Foundation of China under Grant No. 11275049. Liren Shan was also supported by Fudan's Undergraduate Research Opportunities Program, FDUROP.

In this paper, we study the following problem: Given a graph with n nodes and m edges, how to create k new edges incident to a designated node v, so that the information centrality I_v of v is maximized. Since I_v equals the reciprocal of the sum of resistance distance $\mathcal{R}_v$ between v and all nodes, we reduce the problem to minimizing $\mathcal{R}_v$ by introducing k edges connecting v. We demonstrate that the optimization function is monotone and supermodular. To minimize resistance distance $\mathcal{R}_v$, we present two greedy approximation algorithms by iteratively introducing k edges one by one. The former is a $\left(1-\frac{1}{e}\right)$-approximation algorithm with $O(n^3)$ time complexity, while the latter is a $\left(1-\frac{1}{e}-\epsilon\right)$-approximation algorithm with $\widetilde{O}(mk\epsilon^{-2})$ time complexity, where the $\widetilde{O}(\cdot)$ notation hides $\mathrm{poly}(\log n)$ factors. We test the performance of our algorithms on several model and real networks, which substantially increase information centrality score of a given node and outperform several other adding edge strategies.

2 Preliminary

Consider a connected undirected weighted network $G = (V, E, w)$ where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $w : E \to \mathbb{R}_+$ is the edge weight function. We use w_{max} to denote the maximum edge weight. Let $n = |V|$ denote the number of nodes and $m = |E|$ denote the number of edges. For a pair of adjacent nodes u and v, we write $u \sim v$ to denote $(u, v) \in E$. The Laplacian matrix of G is the symmetric matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{A}$ is the weighted adjacency matrix of the graph and $\boldsymbol{D}$ is the degree diagonal matrix.

Let $\boldsymbol{e}_i$ denote the ith standard basis vector, and $\boldsymbol{b}_{u,v} = \boldsymbol{e}_u - \boldsymbol{e}_v$. We fix an arbitrary orientation for all edges in G. For each edge $e \in E$, we define $\boldsymbol{b}_e = \boldsymbol{b}_{u,v}$, where u and v are head and tail of e, respectively. It is easy to verify that $\boldsymbol{L} = \sum_{e\in E} w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$, where $w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$ is the Laplacian of e. $\boldsymbol{L}$ is singular and positive semidefinite. Its pseudoinverse $\boldsymbol{L}^\dagger$ is $\left(\boldsymbol{L} + \frac{1}{n}\boldsymbol{J}\right)^{-1} - \frac{1}{n}\boldsymbol{J}$, where $\boldsymbol{J}$ is the matrix with all entries being ones.

For network $G = (V, E, w)$, the resistance distance [Klein and Randić, 1993] between two nodes u, v is $\mathcal{R}_{uv} = \boldsymbol{b}_{u,v}^\top \boldsymbol{L}^\dagger \boldsymbol{b}_{u,v}$. The resistance distance $\mathcal{R}_v$ of a node v is the sum of resistance distances between v and all nodes in V, that is, $\mathcal{R}_v = \sum_{u\in V} \mathcal{R}_{uv}$, which can be expressed in terms of the entries of $\boldsymbol{L}^\dagger$ as [Bozzo and Franceschet, 2013]

$$\mathcal{R}_v = n\boldsymbol{L}^\dagger_{vv} + \mathrm{Tr}\left(\boldsymbol{L}^\dagger\right). \tag{1}$$

Let $\boldsymbol{L}_v$ denote the submatrix of Laplacian $\boldsymbol{L}$, which is obtained from $\boldsymbol{L}$ by deleting the row and column corresponding to node v. For a connected graph G, $\boldsymbol{L}_v$ is invertible for any node v, and the resistance distance $\mathcal{R}_{uv}$ between v and another node u is equal to $\left(\boldsymbol{L}_v^{-1}\right)_{uu}$ [Izmailian *et al.*, 2013]. Thus, we have

$$\mathcal{R}_v = \mathrm{Tr}\left(\boldsymbol{L}_v^{-1}\right). \tag{2}$$

The resistance distance $\mathcal{R}_v$ can be used as a measure of the efficiency for node v in transmitting information to other nodes, and is closely related to information centrality introduced by Stephenson and Zelen to measure the importance of nodes in social networks [Stephenson and Zelen, 1989]. The information I_{uv} transmitted between u and v is defined as

$$I_{uv} = \frac{1}{\boldsymbol{B}^{-1}(u,u) + \boldsymbol{B}^{-1}(v,v) - 2\boldsymbol{B}^{-1}(u,v)},$$

where $\boldsymbol{B} = \boldsymbol{L} + \boldsymbol{J}$. The information centrality I_v of node v is the harmonic mean of I_{uv} over all nodes u [Stephenson and Zelen, 1989].

Definition 2.1 *For a connected graph $G = (V, E, w)$, the information centrality I_v of a node $v \in V$ is defined as*

$$I_v = \frac{n}{\sum\limits_{u\in V} 1/I_{uv}}.$$

It was shown [Brandes and Fleischer, 2005] that

$$I_v = \frac{n}{\mathcal{R}_v}. \tag{3}$$

We continue to introduce some useful notations and tools for the convenience of description for our algorithms, including ϵ-approximation and supermodular function.

Let $a, b \geq 0$ be two nonnegative scalars. We say a is an ϵ-approximation [Peng and Spielman, 2014] of b if $\exp(-\epsilon)\, a \leq b \leq \exp(\epsilon)\, a$. Hereafter, we use $a \approx_\epsilon b$ to represent that a is an ϵ-approximation of b.

Let X be a finite set, and 2^X be the set of all subsets of X. Let $f : 2^X \to \mathbb{R}$ be a set function on X. For any subsets $S \subset T \subset X$ and any element $a \in X \setminus T$, we say function $f(\cdot)$ is supermodular if it satisfies $f(S) - f(S \cup \{a\}) \geq f(T) - f(T \cup \{a\})$. A function $f(\cdot)$ is submodular if $-f(\cdot)$ is supermodular. A set function $f : 2^X \to \mathbb{R}$ is called monotone decreasing if for any subsets $S \subset T \subset X$, $f(S) > f(T)$ holds.

3 Problem Formulation

For a connected undirected weighted network $G(V, E, w)$, given a set S of weighted edges not in E, we use $G + S$ to denote the network augmented by adding the edges in S to G, i.e. $G + S = (V, E \cup S, w')$, where $w' : E \cup S \to \mathbb{R}_+$ is the new weight function. Let $\boldsymbol{L}(S)$ denote the Laplacian matrix for $G + S$. Note that the information centrality of a node depends on the graph topology. If we augment a graph by adding a set of edges S, the information centrality of a node will change. Moreover, adding edges incident to some node v can only increase its information centrality [Doyle and Snell, 1984].

Assume that there is a set of nonexistent edges incident to a particular node v, each with a given weight. We denote this candidate edge set as E_v. Consider choosing a subset S of k edges from the candidate set E_v to augment the network so that the information centrality of node v is maximized. Let $I_v(S)$ denote the information centrality of the node v in augmented network. We define the following set function optimization problem:

$$\underset{S\subset E_v,\, |S|=k}{\mathrm{maximize}} \quad I_v(S). \tag{4}$$

Since the information centrality I_v of a node v is proportional to the reciprocal of $\mathcal{R}_v$, the optimization problem (4) is equivalent to the following problem:

$$\underset{S \subset E_v,\, |S|=k}{\text{minimize}} \quad \mathcal{R}_v(S), \tag{5}$$

where $\mathcal{R}_v(S)$ is the resistance distance of v in the augmented network $G + S$.

4 Supermodularity of Objective Function

Let 2^{E_v} denote all subsets of E_v. Then the resistance distance of node v in the augmented network can be represented as a set function $\mathcal{R}_v : 2^{E_v} \to \mathbb{R}$. To provide effective algorithms for the above-defined problems, we next prove that the resistance distance of v is a supermodular function.

Rayleigh's monotonicity law [Doyle and Snell, 1984] shows that the resistance distance between any pair of nodes can only decrease when edges are added. Then, we have the following theorem.

Theorem 4.1 *$\mathcal{R}_v(S)$ is a monotonically decreasing function of the set of edges S. That is, for any subsets $S \subset T \subset E_v$,*

$$\mathcal{R}_v(T) < \mathcal{R}_v(S).$$

We then prove the supermodularity of the objective function $\mathcal{R}_v(S)$.

Theorem 4.2 *$\mathcal{R}_v(S)$ is supermodular. For any set $S \subset T \subset E_v$ and any edge $e \in E_v \setminus T$,*

$$\mathcal{R}_v(T) - \mathcal{R}_v(T \cup \{e\}) \leq \mathcal{R}_v(S) - \mathcal{R}_v(S \cup \{e\}).$$

Proof. Suppose that edge e connects two nodes u and v, then $\boldsymbol{L}(S \cup \{e\})_v = \boldsymbol{L}(S)_v + w(e)\boldsymbol{E}_{uu}$, where $\boldsymbol{E}_{uu}$ is a square matrix with the uth diagonal entry being one, and all other entries being zeros. By (2), it suffices to prove that

$$\begin{aligned}&\mathrm{Tr}\left(\boldsymbol{L}(T)_v^{-1}\right) - \mathrm{Tr}\left((\boldsymbol{L}(T)_v + w(e)\boldsymbol{E}_{uu})^{-1}\right)\\ \leq &\mathrm{Tr}\left(\boldsymbol{L}(S)_v^{-1}\right) - \mathrm{Tr}\left((\boldsymbol{L}(S)_v + w(e)\boldsymbol{E}_{uu})^{-1}\right).\end{aligned}$$

Since S is a subset of T, $\boldsymbol{L}(T)_v = \boldsymbol{L}(S)_v + \boldsymbol{P}$, where $\boldsymbol{P}$ is a nonnegative diagonal matrix. For simplicity, in the following proof, we use $\boldsymbol{M}$ to denote matrix $\boldsymbol{L}(S)_v$. Then, we only need to prove

$$\begin{aligned}&\mathrm{Tr}\left((\boldsymbol{M}+\boldsymbol{P})^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{M}^{-1}\right)\\ \leq &\mathrm{Tr}\left((\boldsymbol{M}+\boldsymbol{P}+w(e)\boldsymbol{E}_{uu})^{-1}\right) - \mathrm{Tr}\left((\boldsymbol{M}+w(e)\boldsymbol{E}_{uu})^{-1}\right).\end{aligned}$$

Define function $f(t)$, $t \in [0, \infty)$, as

$$f(t) = \mathrm{Tr}\left((\boldsymbol{M}+\boldsymbol{P}+t\boldsymbol{E}_{uu})^{-1}\right) - \mathrm{Tr}\left((\boldsymbol{M}+t\boldsymbol{E}_{uu})^{-1}\right).$$

Then, the above inequality holds if $f(t)$ takes the minimum value at $t = 0$. We next show that $f(t)$ is an increasing function by proving $\frac{df(t)}{dt} \geq 0$. Using the matrix derivative formula

$$\frac{d}{dt}\mathrm{Tr}\left(\boldsymbol{A}(t)^{-1}\right) = -\mathrm{Tr}\left(\boldsymbol{A}(t)^{-1}\frac{d}{dt}\boldsymbol{A}(t)\boldsymbol{A}(t)^{-1}\right),$$

we can differentiate function $f(t)$ as

$$\begin{aligned}\frac{df(t)}{dt} &= -\mathrm{Tr}\left((\boldsymbol{M}+\boldsymbol{P}+t\boldsymbol{E}_{uu})^{-1}\boldsymbol{E}_{uu}(\boldsymbol{M}+\boldsymbol{P}+t\boldsymbol{E}_{uu})^{-1}\right)\\ &\quad+\mathrm{Tr}\left((\boldsymbol{M}+t\boldsymbol{E}_{uu})^{-1}\boldsymbol{E}_{uu}(\boldsymbol{M}+t\boldsymbol{E}_{uu})^{-1}\right)\\ &= -\mathrm{Tr}\left(\boldsymbol{E}_{uu}(\boldsymbol{M}+\boldsymbol{P}+t\boldsymbol{E}_{uu})^{-2}\right)\\ &\quad+\mathrm{Tr}\left(\boldsymbol{E}_{uu}(\boldsymbol{M}+t\boldsymbol{E}_{uu})^{-2}\right)\\ &= -\left((\boldsymbol{M}+\boldsymbol{P}+t\boldsymbol{E}_{uu})^{-2}\right)_{uu} + \left((\boldsymbol{M}+t\boldsymbol{E}_{uu})^{-2}\right)_{uu}.\end{aligned}$$

Let $\boldsymbol{N} = \boldsymbol{M} + t\boldsymbol{E}_{uu}$, and let $\boldsymbol{Q}$ be a nonnegative diagonal matrix with exactly one positive diagonal entry $\boldsymbol{Q}_{hh} > 0$ and all other entries being zeros. We now prove that $\boldsymbol{N}^{-1}_{ij} \geq (\boldsymbol{N}+\boldsymbol{Q})^{-1}_{ij}$ for $1 \leq i, j \leq n-1$. Using Sherman-Morrison formula [Meyer, 1973], we have

$$\boldsymbol{N}^{-1} - (\boldsymbol{N}+\boldsymbol{Q})^{-1} = \frac{\boldsymbol{Q}_{hh}\boldsymbol{N}^{-1}\boldsymbol{e}_h\boldsymbol{e}_h^\top\boldsymbol{N}^{-1}}{1+\boldsymbol{Q}_{hh}\boldsymbol{e}_h^\top\boldsymbol{N}^{-1}\boldsymbol{e}_h}.$$

Since $\boldsymbol{N}$ is an M-matrix, every entry of $\boldsymbol{N}^{-1}$ is positive [Plemmons, 1977], it is the same with every entry of $\boldsymbol{N}^{-1}\boldsymbol{e}_h\boldsymbol{e}_h^\top\boldsymbol{N}^{-1}$. In addition, the denominator $1 + \boldsymbol{Q}_{hh}\boldsymbol{e}_h^\top\boldsymbol{N}^{-1}\boldsymbol{e}_h$ is also positive, because $\boldsymbol{N}$ is positive definite. Therefore, $\boldsymbol{N}^{-1} - (\boldsymbol{N}+\boldsymbol{Q})^{-1}$ is a positive matrix, the entries of which are all greater than zero.

By repeatedly applying the above process, we conclude that $\boldsymbol{N}^{-1} \geq (\boldsymbol{N}+\boldsymbol{P})^{-1}$ is a positive matrix. Thus,

$$\frac{df(t)}{dt} = -\left((\boldsymbol{N}+\boldsymbol{P})^{-2}\right)_{uu} + \left(\boldsymbol{N}^{-2}\right)_{uu} \geq 0,$$

which completes the proof. □

5 Simple Greedy Algorithm

Theorems 4.1 and 4.2 indicate that the objective function (5) is monotone and supermodular. Thus, a simple greedy algorithm is sufficient to approximate problem (5) with provable optimality bounds. In the greedy algorithm, the augmented edge set S is initially empty. Then k edges are iteratively added to the augmented edge set from the set E_v of candidate edges. At each iteration, an edge e_i in the candidate edge set is selected to maximize $\mathcal{R}_v(S) - \mathcal{R}_v(S \cup \{e_i\})$. The algorithm terminates when $|S| = k$.

According to (1), the effective resistance $\mathcal{R}_v$ is equal to $n\boldsymbol{L}^\dagger_{vv} + \mathrm{Tr}(\boldsymbol{L}^\dagger)$. A naive algorithm requires $O(k|E_v|n^3)$ time complexity, which is prohibitively expense. Below we show that the computation cost can be reduced to $O(n^3)$ by using Sherman-Morrison formula [Meyer, 1973].

Lemma 5.1 *For a connected weighted graph $G = (V, E, w)$ with weighted Laplacian matrix $\boldsymbol{L}$, let e be a nonexistent edge with given weight $w(e)$ connecting node v. Then,*

$$(\boldsymbol{L}(\{e\}))^\dagger = \left(\boldsymbol{L} + w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top\right)^\dagger = \boldsymbol{L}^\dagger - \frac{w(e)\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^\top\boldsymbol{L}^\dagger}{1 + w(e)\boldsymbol{b}_e^\top\boldsymbol{L}^\dagger\boldsymbol{b}_e}.$$

For a candidate edge not added to S, let $\mathcal{R}_v^\Delta(e) = \mathcal{R}_v(S) - \mathcal{R}_v(S \cup \{e\})$. Lemma 5.1 and (1) lead to the following result.

Lemma 5.2 *Let $G = (V, E, w)$ be a connected weighted graph with weighted Laplacian matrix $\boldsymbol{L}$. Let $e \notin E$ be a candidate edge with given weight $w(e)$ incident to node v. Then,*

$$\mathcal{R}_v^{\Delta}(e) = \frac{w(e)\left(n\left(\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\right)_{vv} + \mathrm{Tr}\left(\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\right)\right)}{1 + w(e)\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\boldsymbol{b}_e}. \tag{6}$$

Lemma 5.2 yields a simple greedy algorithm EXACTSM(G, v, E_v, k), as outlined in Algorithm 1. The first step of this algorithm is to compute the pseudoinverse of $\boldsymbol{L}$, the time complexity of which is $O(n^3)$ time. Then this algorithm works in k rounds, each involving operations of computations and updates with time complexity $O(n^2)$. Thus, the total running time of Algorithm 1 is $O(n^3)$.

```
Algorithm 1: EXACTSM($G, v, E_v, k$)
  Input  : A connected graph $G$; a node $v \in V$; a
           candidate edge set $E_v$; an integer $k \le |E_v|$
  Output : A subset of $S \subset E_v$ and $|S| = k$
1 Initialize solution $S = \emptyset$
2 Compute $\boldsymbol{L}^{\dagger}$
3 for $i = 1$ to $k$ do
4     Compute $\mathcal{R}_v^{\Delta}(e)$ for each $e \in E_v \setminus S$
5     Select $e_i$ s.t. $e_i \leftarrow \arg\max_{e \in E_v \setminus S} \mathcal{R}_v^{\Delta}(e)$
6     Update solution $S \leftarrow S \cup \{e_i\}$
7     Update the graph $G \leftarrow G(V, E \cup \{e_i\})$
8     Update $\boldsymbol{L}^{\dagger} \leftarrow \boldsymbol{L}^{\dagger} - \frac{w(e_i)\boldsymbol{L}^{\dagger}\boldsymbol{b}_{e_i}\boldsymbol{b}_{e_i}^{\top}\boldsymbol{L}^{\dagger}}{1+w(e_i)\boldsymbol{b}_{e_i}^{\top}\boldsymbol{L}^{\dagger}\boldsymbol{b}_{e_i}}$
9 return S
```

Moreover, due to the result in [Nemhauser *et al.*, 1978], Algorithm 1 is able to achieve a $\left(1 - \frac{1}{e}\right)$ approximation factor, as given in the following theorem.

Theorem 5.3 *The set S returned by Algorithm 1 satisfies*

$$\mathcal{R}_v(\emptyset) - \mathcal{R}_v(S) \ge \left(1 - \frac{1}{e}\right)(\mathcal{R}_v(\emptyset) - \mathcal{R}_v(S^*)),$$

where S^* is the optimal solution to (5), i.e.,

$$S^* \stackrel{\mathrm{def}}{=} \underset{S \subset V,\, |S|=k}{\arg\min}\ \mathcal{R}_v(S).$$

6 Fast Greedy Algorithm

Although Algorithm 1 is faster than the naive algorithm, it is still computationally infeasible for large networks, since it involves the computation of the pseudoinverse for $\boldsymbol{L}$. In this section, in order to avoid inverting the matrix $\boldsymbol{L}$, we give an efficient approximation algorithm, which achieves a $\left(1 - \frac{1}{e} - \epsilon\right)$ approximation factor of optimal solution to problem (5) in time $\widetilde{O}(km\epsilon^{-2})$.

6.1 Approximating $\mathcal{R}_v^{\Delta}(e)$

In order to solve problem (5), one need to compute the key quantity $\mathcal{R}_v^{\Delta}(e)$ in (6). Here, we provide an efficient algorithm to approximate $\mathcal{R}_v^{\Delta}(e)$ properly.

We first consider the denominator in (6). Assume that the new added edge e connects nodes u and v. Note that the term $r_e = \boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\boldsymbol{b}_e$ in the denominator is in fact the resistance distance $\mathcal{R}_{uv}$ between u and v in the network excluding e. It can be computed by the following approximation algorithm [Spielman and Srivastava, 2011].

Lemma 6.1 *Let $G = (V, E, w)$ be a weighted connected graph. There is an algorithm* APPROXIER(G, E_v, ϵ) *that returns an estimate $\hat{r}_e$ of r_e for all $e \in E_v$ in $\widetilde{O}(m\epsilon^{-2})$ time. With probability at least $1 - 1/n$, $\hat{r}_e \approx_{\epsilon} r_e$ holds for all $e \in E_v$.*

For the numerator of (6), it includes two terms, $\left(\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\right)_{vv}$ and $\mathrm{Tr}\left(\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\right)$. The first term can be calculated by $\left(\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\right)_{vv} = \boldsymbol{e}_v^{\top}\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}\boldsymbol{e}_v$. The second term is the trace of an implicit matrix which can be approximated by Hutchinson's Monte-Carlo method [Hutchinson, 1989]. By generating M independent random ± 1 vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M \in \mathbb{R}^n$ (i.e., independent Bernoulli entries), $\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i^{\top}\boldsymbol{A}\boldsymbol{x}_i$ can be used to estimate the trace of matrix $\boldsymbol{A}$. Since $\mathbb{E}\left[\boldsymbol{x}_i^{\top}\boldsymbol{A}\boldsymbol{x}_i\right] = \mathrm{Tr}(\boldsymbol{A})$, by the law of large numbers, $\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i^{\top}\boldsymbol{A}\boldsymbol{x}_i$ should be close to $\mathrm{Tr}(\boldsymbol{A})$ when M is large. The following lemma [Avron and Toledo, 2011] provides a good estimation of $\mathrm{Tr}(\boldsymbol{A})$.

Lemma 6.2 *Let $\boldsymbol{A}$ be a positive semidefinite matrix with rank $\mathrm{rank}(\boldsymbol{A})$. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ be independent random ± 1 vectors. Let ϵ, δ be scalars such that $0 < \epsilon \le 1/2$ and $0 < \delta < 1$. For any $M \ge 24\epsilon^{-2}\ln(2\mathrm{rank}(\boldsymbol{A})/\delta)$, the following statement holds with probability at least $1 - \delta$:*

$$\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i^{\top}\boldsymbol{A}\boldsymbol{x}_i \approx_{\epsilon} \mathrm{Tr}(\boldsymbol{A}).$$

Thus, we have reduced the estimation of the numerator of (6) to the calculation of the quadratic form of $\boldsymbol{L}^{\dagger}\boldsymbol{b}_e\boldsymbol{b}_e^{\top}\boldsymbol{L}^{\dagger}$. If we directly compute the quadratic form, we must first evaluate $\boldsymbol{L}^{\dagger}$, the time complexity is high. To avoid inverting $\boldsymbol{L}$, we will utilize the nearly-linear time solver for Laplacian systems from [Kyng and Sachdeva, 2016], whose performance can be characterized in the following lemma.

Lemma 6.3 *The algorithm $\boldsymbol{y}$ = LAPLSOLVE$(\boldsymbol{L}, \boldsymbol{z}, \epsilon)$ takes a Laplacian matrix $\boldsymbol{L}$ of a graph G with n nodes and m edges, a vector $\boldsymbol{z} \in \mathbb{R}^n$ and a scalar $\epsilon > 0$ as input, and returns a vector $\boldsymbol{y} \in \mathbb{R}^n$ such that with probability $1 - 1/\mathrm{poly}(n)$ the following statement holds:*

$$\left\|\boldsymbol{y} - \boldsymbol{L}^{\dagger}\boldsymbol{z}\right\|_{\boldsymbol{L}} \le \epsilon\left\|\boldsymbol{L}^{\dagger}\boldsymbol{z}\right\|_{\boldsymbol{L}},$$

where $\|\boldsymbol{x}\|_{\boldsymbol{L}} = \sqrt{\boldsymbol{x}^{\top}\boldsymbol{L}\boldsymbol{x}}$. The algorithm runs in expected time $\widetilde{O}(m)$.

Lemmas 6.1, 6.2 and 6.3 result in the following algorithm VREFFCOMP(G, v, E_v, ϵ) for computing $\mathcal{R}_v^{\Delta}(e)$ for all $e \in E_v$, as depicted in Algorithm 2. The algorithm has a total running time $\widetilde{O}(m\epsilon^{-2})$, and returns a set of pairs $\{(e, \hat{\mathcal{R}}_v^{\Delta}(e)) | e \in E_v\}$, satisfying that $\mathcal{R}_v^{\Delta}(e) \approx_{\epsilon} \hat{\mathcal{R}}_v^{\Delta}(e)$ for all $e \in E_v$.

```
Algorithm 2: VREFFCOMP$(G, v, E_v, \epsilon)$
Input  : A graph $G$; a node $v \in V$; a candidate edge set $E_v$;
         a real number $0 \leq \epsilon \leq 1/2$
Output : $\{(e, \hat{\mathcal{R}}_v^{\Delta}(e)) | e \in E_v\}$
1  Let $\boldsymbol{z}_1, \dots, \boldsymbol{z}_M$ be independent random $\pm 1$ vectors, where
     $M = \lceil 432\epsilon^{-2} \ln(2n) \rceil$.
2  for $i = 1$ to $M$ do
3      $\boldsymbol{y}_i \leftarrow$ LAPLSOLVE$(\boldsymbol{L}, \boldsymbol{z}_i, \frac{1}{72}\epsilon n^{-8} w_{max}^{-4})$
4      for each $e \in E_v$ do
5          Compute $t_i(e) \overset{\text{def}}{=} \boldsymbol{y}_i^\top \boldsymbol{b}_e \boldsymbol{b}_e^\top \boldsymbol{y}_i$
6  $\boldsymbol{x} \leftarrow$ LAPLSOLVE$(\boldsymbol{L}, \boldsymbol{e}_v, \frac{1}{72}\epsilon n^{-9} w_{max}^{-4})$
7  for each $e \in E_v$ do
8      Compute $\alpha(e) \overset{\text{def}}{=} \boldsymbol{x}^\top \boldsymbol{b}_e \boldsymbol{b}_e^\top \boldsymbol{x}$
9  $\hat{r}_e \leftarrow$ APPROXIER$(G, \epsilon/3)$
10 Compute $\hat{\mathcal{R}}_v^{\Delta}(e) = w(e) \frac{n\alpha(e) + \frac{1}{M}\sum_i^M t_i(e)}{1 + w(e)\hat{r}_e}$ for each $e$
11 return $\{(e, \hat{\mathcal{R}}_v^{\Delta}(e)) | e \in E_v\}$
```

6.2 Fast Algorithm for Objective Function

By using Algorithm 2 to approximate $\mathcal{R}_v^{\Delta}(e)$, we give a fast greedy algorithm APPROXISM(G, v, E_v, k, ϵ) for solving problem (5), as outlined in Algorithm 3.

```
Algorithm 3: APPROXISM$(G, v, E_v, k, \epsilon)$
Input  : A graph $G$; a node $v \in V$; a candidate edge set $E_v$;
         an integer $k \leq |E_v|$; a real number $0 \leq \epsilon \leq 1/2$
Output : $S$: a subset of $E_v$ and $|S| = k$
1  Initialize solution $S = \emptyset$
2  for $i = 1$ to $k$ do
3      $\{e, \hat{\mathcal{R}}_v^{\Delta}(e) | e \in E_v \backslash S\} \leftarrow$ VREFFCOMP$(G, v, E_v \backslash S, 3\epsilon)$.
4      Select $e_i$ s.t. $e_i \leftarrow \arg\max_{e \in E_v \backslash S} \hat{\mathcal{R}}_v^{\Delta}(e)$
5      Update solution $S \leftarrow S \cup \{e_i\}$
6      Update the graph $G \leftarrow G(V, E \cup \{e_i\})$
7  return $S$
```

Algorithm 3 works in k rounds (Lines 2-6). In every round, the call of VREFFCOMP and updates take time $\widetilde{O}(m\epsilon^{-2})$. Then, the total running time of Algorithm 3 is $\widetilde{O}(km\epsilon^{-2})$. The following theorem shows that the output $\hat{S}$ of Algorithm 3 gives a $\left(1 - \frac{1}{e} - \epsilon\right)$ approximate solution to problem (5).

Theorem 6.4 *For any $0 < \epsilon \leq 1/2$, the set $\hat{S}$ returned by the greedy algorithm above satisfies*

$$\mathcal{R}_v(\emptyset) - \mathcal{R}_v(\hat{S}) \geq \left(1 - \frac{1}{e} - \epsilon\right)(\mathcal{R}_v(\emptyset) - \mathcal{R}_v(S^*)),$$

where S^* is the optimal solution to problem (5), i.e.,

$$S^* \overset{\text{def}}{=} \underset{S \subset V, |S|=k}{\arg\min} \quad \mathcal{R}_v(S).$$

We omit the proof, since it is similar to that in [Badanidiyuru and Vondrák, 2014].

Network	n	m	n'	m'
BA network	50	94	50	94
WS network	50	100	50	100
Zachary karate club	34	78	34	78
Windsufers	43	336	43	336
Jazz musicians	198	2742	195	1814
Virgili	1,133	5,451	1,133	5,451
Euroroad	1,174	1,417	1,039	1,305
Hamster full	2,426	16,631	2,000	16,098
Facebook	2,888	2,981	2,888	2,981
Powergrid	4,941	6,594	4,941	6,594
ca-GrQc	5,242	14,496	4,158	13,422
ca-HepPh	12,008	118,521	11,204	117,619
com-DBLP	317,080	1,049,866	317,080	1,049,866
roadNet-TX	1,379,917	1,921,660	1,351,137	1,879,201

Table 1: Statistics of datasets. For a network with n nodes and m edges, we denote the number of nodes and edges in its largest connected component by n' and m', respectively.

7 Experiments

In this section, we experimentally evaluate the effectiveness and efficiency of our two greedy algorithms on some model and real networks. All algorithms in our experiments are implemented in Julia. In our algorithms, we use the LAPLSOLVE [Kyng and Sachdeva, 2016], the implementation (in Julia) of which is available on website[1]. All experiments were conducted on a machine with 4.2 GHz Intel i7-7700 CPU and 32G RAM.

We execute our experiments on two popular model networks, Barabási-Albert (BA) network and Watts–Strogatz (WS) network, and a large connection of realistic networks from KONECT [Kunegis, 2013] and SNAP[2]. Table 1 provides the information of these networks, where real-world networks are shown in increasing size of the number of nodes in original networks.

7.1 Effectiveness of Greedy Algorithms

To show the effectiveness of our algorithms, we compare the results of our algorithms with the optimum solutions on two small model networks, BA network and WS network, and two small real-world networks, Zachary karate club network and Windsufers contact network. Since these networks are small, we are able to compute the optimal edge set.

For each network, we randomly choose 20 target nodes. For each target node v, the candidate edge set is composed of all nonexistent edges incident to it with unit weight $w = 1$. And for each designated $k = 1, 2, \cdots, 6$, we add k edges linked to v and other k non-neighboring nodes of v. We then compute the average information centrality of the 20 target nodes for each k. Also, we compute the solutions for the random scheme, by adding k edges from randomly selected k non-neighboring nodes. The results are reported in Figure 1. We observe that there is little difference between the solutions of our greedy algorithms and the optimal solutions, since their approximation ratio is always greater than 0.98, which is far better than the theoretical guarantees. Moreover, our greedy schemes outperform the random scheme in these four networks.

[1]https://github.com/danspielman/Laplacians.jl

[2]https://snap.stanford.edu

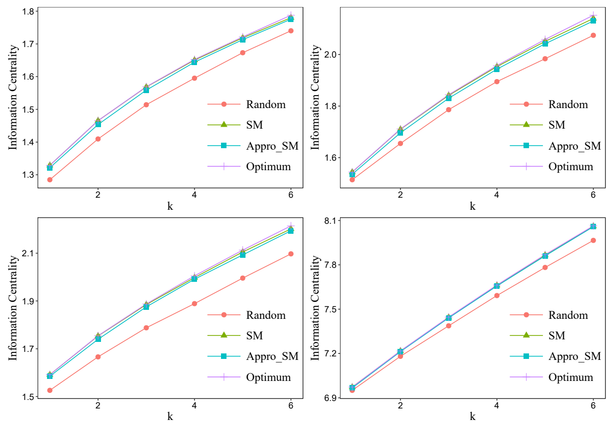

Figure 1: Average information centrality of target nodes as a function of the number k of inserted edges for EXACTSM, APPROXISM, random and the optimum solution on four networks: BA (a), WS (b), Karate club (c), and Windsufers (d).

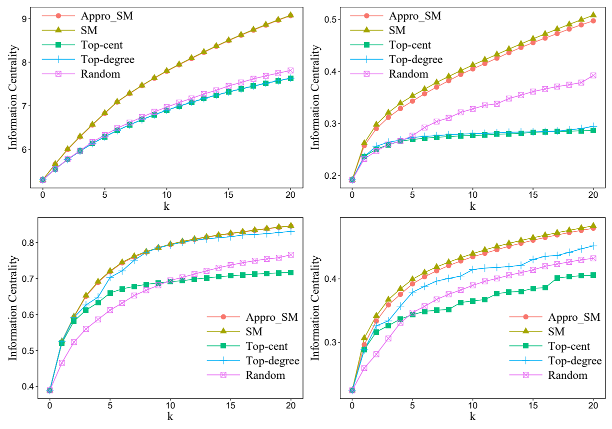

Figure 2: Average information centrality of target nodes as a function of the number k of inserted edges for the five heuristics on Jazz musicians (a), Euroroad (b), Facebook (c), Powergrid (d).

To further demonstrate the effectiveness of our algorithms, we compare the results of our methods with the random scheme and other two baseline schemes, Top-degree and Top-cent, on four other real-world networks. In Top-degree scheme, the added edges are simply the k edges connecting target node v and its nonadjacent nodes with the highest degree in the original network; while in Top-cent scheme, the added edges are simply those k edges connecting target node v and its nonadjacent nodes with the largest information centrality in the original network.

Since the results may vary depending on the initial information centrality of the target node v, for each of the four real networks, we select 10 different target nodes at random. For each target node, we first compute its original information centrality and increase it by adding up to $k = 20$ new edges, using our two greedy algorithms and the three baselines. Then, we compute and record the information centrality of the target node after insertion of every edge. Finally, we compute the average information centrality of all the 10 target nodes for each $k = 1, 2, \ldots, 20$, which is plotted in Figure 2. We observe that for all the four real-world networks our greedy algorithms outperform the three baselines.

7.2 Efficiency Comparison of Greedy Algorithms

Although both of our greedy algorithms are effective, we will show that their efficiency greatly differs. To this end, we compare the efficiency of the greedy algorithms on several real-world networks. For each network, we choose stochastically 20 target nodes, for each of which, we create $k = 10$ new edges incident to it to maximize its information centrality according to Algorithms 1 and 3. We compute the average information centrality of 10 target nodes for each network and record the average running times. In Table 2 we provide the results of average information centrality and average running time of our greedy algorithms. We observe that APPROXISM algorithm are faster than EXACTSM algorithm, especially for large networks, while their final information centrality score are close. More interestingly, APPROXISM applies to massive networks. For example, for com-DBLP and roadNet-TX

Network	Time (seconds)			Information centrality		
	ASM	ESM	Ratio	ASM	ESM	Ratio
Virgili	1.3996	0.9172	1.5259	2.5005	2.5037	0.9987
Euroroad	0.6563	0.7593	0.8643	0.4003	0.4069	0.9838
Hamster full	3.0785	4.8528	0.6344	2.9904	2.9944	0.9987
Facebook	1.7151	12.9203	0.1327	0.7937	0.7947	0.9987
Powergrid	5.8727	58.3359	0.1006	0.4327	0.4369	0.9904
ca-GrQc	5.3023	34.0228	0.1558	1.2118	1.2136	0.9985
ca-HepPh	28.7462	620.4557	0.0463	2.2569	2.2592	0.9990
com-DBLP	697.1835	-	-	1.1327	-	-
roadNet-TX	1569.5059	-	-	0.0556	-	-

Table 2: The average running times and results of APPROXISM (ASM) and EXACTSM (ESM) algorithms on several real-world networks, as well as the ratios for times and results of APPROXISM to those of EXACTSM.

networks, APPROXISM computes their information centrality in half an hour, while APPROXISM fails due to its high time complexity.

8 Conclusions

In this paper, we considered the problem of maximizing the information centrality of a designated node v by adding k new edges incident to it. This problem is equivalent to minimizing the resistance distance $\mathcal{R}_v$ of node v. We proposed two approximation algorithms for computing $\mathcal{R}_v$ when k edges are repeatedly inserted in a greedy way. The first one gives a $\left(1 - \frac{1}{e}\right)$ approximation of the optimum in time $O(n^3)$. While the second one returns a $\left(1 - \frac{1}{e} - \epsilon\right)$ approximation in time $\widetilde{O}(mk\epsilon^{-2})$. Since the considered problem has never been addressed before, we have no other algorithms to compare with, but compare our algorithms with potential alternative algorithms. Extensive experimental results on model and realistic networks show that our algorithms can often compute an approximate optimal solution. Particularly, our second algorithm can achieve a good approximate solution very quickly, making it applicable to massive networks.

References

[Avrachenkov and Litvak, 2006] K. Avrachenkov and N. Litvak. The effect of new links on Google PageRank. *Stochastic Models*, 22(2):319–331, 2006.

[Avron and Toledo, 2011] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of ACM*, 58(2):8:1–8:34, 2011.

[Badanidiyuru and Vondrák, 2014] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514. SIAM, 2014.

[Bergamini *et al.*, 2016] E. Bergamini, M. Wegner, D. Lukarski, and H. Meyerhenke. Estimating current-flow closeness centrality with a multigrid Laplacian solver. In CSC, pages 1–12, 2016.

[Boldi and Vigna, 2014] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[Bozzo and Franceschet, 2013] E. Bozzo and M. Franceschet. Resistance distance, closeness, and betweenness. *Social Networks*, 35(3):460–469, 2013.

[Brandes and Fleischer, 2005] U. Brandes and D. Fleischer. Centrality measures based on current flow. In *STACS*, volume 3404, pages 533–544. Springer-Verlag, 2005.

[Crescenzi *et al.*, 2015] P. Crescenzi, G. D'Angelo, L. Severini, and Y. Velaj. Greedily improving our own centrality in a network. In *SEA*, pages 43–55. Springer, 2015.

[Crescenzi *et al.*, 2016] P. Crescenzi, G. D'Angelo, L. Severini, and Y. Velaj. Greedily improving our own closeness centrality in a network. *ACM Transactions on Knowledge Discovery from Data*, 11(1):9, 2016.

[D'Angelo *et al.*, 2016] G. D'Angelo, L. Severini, and Y. Velaj. On the maximum betweenness improvement problem. *Electronic Notes in Theoretical Computer Science*, 322:153–168, 2016.

[Demaine and Zadimoghaddam, 2010] E. D. Demaine and M. Zadimoghaddam. Minimizing the diameter of a network using shortcut edges. In *SWAT*, pages 420–431. Springer-Verlag, 2010.

[Doyle and Snell, 1984] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.

[Hoffmann *et al.*, 2018] C. Hoffmann, H. Molter, and M. Sorge. The parameterized complexity of centrality improvement in networks. In *SOFSEM*, pages 111–124. Springer, 2018.

[Hutchinson, 1989] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[Ishakian *et al.*, 2012] V. Ishakian, D. Erdös, E. Terzi, and A. Bestavros. A framework for the evaluation and management of network centrality. In *SDM*, pages 427–438. SIAM, 2012.

[Izmailian *et al.*, 2013] N. Sh Izmailian, R. Kenna, and FY Wu. The two-point resistance of a resistor network: a new formulation and application to the cobweb network. *Journal of Physics A: Mathematical and Theoretical*, 47(3):035003, 2013.

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

[Kunegis, 2013] J. Kunegis. Konect: the koblenz network collection. In *WWW*, pages 1343–1350. ACM, 2013.

[Kyng and Sachdeva, 2016] R. Kyng and S. Sachdeva. Approximate Gaussian elimination for Laplacians-fast, sparse, and simple. In *FOCS*, pages 573–582. IEEE, 2016.

[Lü *et al.*, 2016] L Lü, D Chen, X Ren, Q Zhang, Y Zhang, and T Zhou. Vital nodes identification in complex networks. *Physics Reports*, 650:1–63, 2016.

[Meyer, 1973] C. D. Meyer, Jr. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24(3):315–323, 1973.

[Meyerson and Tagiku, 2009] A. Meyerson and B. Tagiku. Minimizing average shortest path distances via shortcut edge addition. *APPROX*, pages 272–285. Springer-Verlag 2009.

[Nemhauser *et al.*, 1978] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

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

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

[Olsen, 2010] M. Olsen. Maximizing PageRank with new backlinks. In *ICAC*, pages 37–48. Springer-Verlag, 2010.

[Parotsidis *et al.*, 2016] N. Parotsidis, E. Pitoura, and P. Tsaparas. Centrality-aware link recommendations. In *WSDM*, pages 503–512. ACM, 2016.

[Peng and Spielman, 2014] R. Peng and D. A Spielman. An efficient parallel solver for SDD linear systems. In *STOC*, pages 333–342. ACM, 2014.

[Plemmons, 1977] R. J. Plemmons. M-matrix characterizations. I nonsingular M-matrices. *Linear Algebra and its Applications*, 18(2):175–188, 1977.

[Spielman and Srivastava, 2011] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal of Computing*, 40(6):1913–1926, 2011.

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

[White and Smyth, 2003] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD*, pages 266–275. ACM, 2003.