MANC: A Node Group Centrality Based on Absorbing Random Walk

Haisong Xia and Zhongzhi Zhang *
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China
School of Computer Science, Fudan University, Shanghai 200433, China
{hsxia18,zhangzz}@fudan.edu.cn

ABSTRACT

In the field of complex networks, the usage of random walk and its relevant quantities arises in many academic researches. In this paper, we consider the connection between random detour time and Kemeny constant, and subsequently extend it to the case of multiple nodes. Inspired by this newly discovered connection, we propose a new centrality that happens to be Multiple Absorbing Node Centrality(MANC). For undirected graphs, MANC for a node set S is defined as the weighted average of the hitting times of absorbing random walks from all nodes in the graph to S. Based on the new centrality, we investigate MANC minimization problem: return the optimal set S^* of absorbing nodes with capacity k, which minimize its MANC $M(S^*)$. In this paper, we prove that this problem is NP-hard. However, thanks to the monotonicity and supermodularity of MANC, we give greedy algorithms with a $1-\frac{k}{(k-1)e}$ approximate factor and cubic running time. To further accelerate the computation of MANC, we give a faster algorithm with a $1-\frac{k}{(k-1)e}-\epsilon$ approximate factor and nearly linear running time for any $\epsilon \in (0,1)$. Numerical experiments on large-scale real networks demonstrate that our second algorithm is still well scalable while maintaining the approximation error.

KEYWORDS

social network, random walk, graph algorithm, Laplacian solver

1 INTRODUCTION

Among all the topics in the field of network science and graph mining, the problem of identifying important nodes has been heavily studied. Due to the small-world and scale-free properties of complex networks, changes in a very few nodes may have a tremendous impact on the entire network. This feature can also be verified in real-life situations such as the spread of virus, community opinion impact, and power grid collapse. If we are able to identify the most critical or vulnerable nodes in a network, then we can take effective preventive measures in advance to avoid unnecessary huge cost. Therefore, the problem of identifying important nodes has high research value.

While a large number of studies related to node centralities have been conducted previously, most of the existing studies focus only on the centrality of single node in the network, which are inadequate when applied to multiple important nodes discovery problems. For example, removing an important node in a network may result in significant changes in the importance of the remaining nodes. Therefore, the traditional important node ranking algorithm has good recognition accuracy only for nodes with the highest centrality, and has limitations in solving the problem of identifying important nodes in practice. The centrality defined for node groups, on the other hand, considers the impact on the network after removing multiple nodes at the same time. It can better adapt to different scenarios in practical applications while improving the accuracy of identification.

Meanwhile, random walk on graphs is widely applicated in many fields due to its recursiveness and robustness: from link prediction [15] and improving search results [10], which are closely related to complex networks, to object detection [3], image background extraction [4] and image annotation refinement [13] in image processing, to DDoS attack detection [14] and clone attack detection [16] in the field of information security.

In this paper, we establish a metric relevant to node group centrality: Multiple Absorbing Node Centrality (MANC) through an absorbing random walk model on graphs. For a connected undirected graph, the MANC $M(S)$ of a node set S is defined as the weighted average of the hitting times of absorbing random walks from all nodes in the graph to S.

Due to the definition of MANC, this metric can be directly applied to the search engine ranking systems. If we denote web pages that satisfy the search keywords as nodes, and denote hyperlinks between web pages as edges, the selected set of web pages should be close enough to all candidate pages, i.e. cover all candidate pages as much as possible. The traditional method of ranking important nodes cannot meet the above requirements, while MANC can theoretically achieve better application results because it is defined directly according to the hitting time of node groups. Consequently, we further construct the MANC minimization problem: finding the node set S^* with capacity k that minimizes MANC $M(S^*)$.

In this paper, we prove that the MANC minimization problem is NP-hard. Subsequently, we use the monotonicity and supermodularity of MANC to design greedy algorithms with approximation guarantees to solve this problem. However, the inevitable matrix inversion in the greedy algorithm leads to a complexity of $O(kn^3)$ for networks containing n nodes, which cannot be employed in large networks. Then, we attempt to reformulate MANC as the quadratic form of either the pseudoinverse of Laplacian matrix or the inverse of a SDDM matrix so that we can approximate it by using SDD solver [2, 12] and Johnson-Lindenstrauss Lemma [5]. For network with n nodes and m edges, our proposed fast greedy algorithm Approx has a $\tilde{O}(km)$ time complexity. In order to verify the accuracy and efficiency of Approx, we perform extensive numerical experiments on networks of different sizes, the result demonstrates that Approx is still well scalable while maintaining

*Corresponding author. Zhongzhi Zhang is also with Shanghai Blockchain Engineering Research Center, as well as Research Institute of Intelligent Complex Systems, Fudan University, Shanghai 200433.

the approximation error and can be applicated in large networks with millions of nodes.

2 PRELIMINARY

In this section, we briefly indroduce some notations as well as some basic concepts on graphs, random walk and special functions.

2.1 Notations

In the rest of the paper, we distinguish scalars, vectors and matrices by using regular lowercase letters a, b, c, bold lowercase letters $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ and bold uppercase letters $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ respectively. For the convenience of representing some specific elements in vectors and matrices, we use $\boldsymbol{a}_i$ to denote the i^{th} entry of vector $\boldsymbol{a}$ and $\boldsymbol{A}_{[i,j]}$ to denote entry (i, j) of matrix $\boldsymbol{A}$. We also use $\boldsymbol{A}_{[i,:]}$ and $\boldsymbol{A}_{[:,j]}$ to respectively denote the i^{th} row and the j^{th} column of matrix $\boldsymbol{A}$.

Moreover, to facilitate the representation of subvectors and submatrices, we introduce sets in the subscripts of vectors and matrices. For example, let $S = \{S_1, S_2, \ldots, S_t\}$, we use $\boldsymbol{a}_{-S}$ to denote the subvector of $\boldsymbol{a}$ obtained by removing the ${S_1}^{\text{th}}, {S_2}^{\text{th}}, \ldots, {S_t}^{\text{th}}$ component. We also use $\boldsymbol{A}_{-S}$ to denote the submatrix of $\boldsymbol{A}$ obtained by removing elements that either in the ${S_1}^{\text{th}}, {S_2}^{\text{th}}, \ldots, {S_t}^{\text{th}}$ row or in the ${S_1}^{\text{th}}, {S_2}^{\text{th}}, \ldots, {S_t}^{\text{th}}$ column.

Note that when a vector or matrix has both superscripts and subscripts, the subscript takes precedence over the superscript. Therefore, $\boldsymbol{A}_{-S}^{-1}$ denotes the inverse of $\boldsymbol{A}_{-S}$ rather than the submatrix of $\boldsymbol{A}^{-1}$. Finally, we use $\boldsymbol{e}_i$ to denote the i^{th} standard basis vector of particular dimensions, and $\mathbf{1}_n \in \mathbb{R}^n$ to denote a vector of n dimensions with all elements of 1. Sometimes, we omit subscripts where there is no ambiguity.

Since we prove the approximation guarantee of our algorithms in Section 5, it is necessary to give the definition of approximate factor.

Definition 2.1 (ϵ-approximation). *Let $x, \tilde{x}$ be positive scalars, ϵ be an error parameter satisfying $\epsilon \in (0, 1)$. Then $\tilde{x}$ is called an ϵ-approximation of x if $(1-\epsilon)\tilde{x} \le x \le (1+\epsilon)\tilde{x}$ holds, which we denote as $x \approx_\epsilon \tilde{x}$ for the convenience of writing.*

2.2 Graphs and Laplacian Matrices

In the rest of the paper, we use $\mathcal{G} = (V, E, w)$ to denote connected weighted undirected graph with $n = |V|$ nodes and $m = |E|$ edges, where V, E denote the node set and edge set of $\mathcal{G}$ respectively, and $w : E \to \mathbb{R}^+$ denote the edge weight function. We use $e = \langle u, v\rangle$ to represent an edge e connecting node u and node v, and also use $w_{\min}$ and $w_{\max}$ to represent the minimum and maximum edge weight of $\mathcal{G}$ respectively, thus, $w_{\min} = \min_{e\in E}\{w_e\}$,$w_{\max} = \max_{e\in E}\{w_e\}$.

After giving the denotation of $\mathcal{G}$, then the adjacency matrix of $\mathcal{G}$ can be denoted as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$: for node $i, j \in V$, $\boldsymbol{A}_{[i,j]} = \boldsymbol{A}_{[j,i]} = w_{i,j}$ if i and j are adjacent, and $\boldsymbol{A}_{[i,j]} = \boldsymbol{A}_{[j,i]} = 0$ otherwise. Moreover, degree $\boldsymbol{d}_i$ of node i can be defined as $\boldsymbol{d}_i = \sum_{j=1}^n \boldsymbol{A}_{[i,j]}$. If we denote the relevant degree diagnoal matrix as $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{d})$, then the Laplacian matrix $\boldsymbol{L}$ of $\mathcal{G}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

The Laplacian matrix of a graph has other definitions. For an undirected graph $\mathcal{G}$, we first assign an arbitary direction to each edge in $\mathcal{G}$, then for edge $e = \langle i, j\rangle \in E$, the row corresponding to edge e in the incidence matrix $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ of $\mathcal{G}$ can be denoted as $\boldsymbol{B}_{[e,:]} = \boldsymbol{e}_i - \boldsymbol{e}_j$. Furthermore, let $\boldsymbol{W} \in \mathbb{R}^{m\times m}$ be a diagnoal matrix whose diagnoal entry (e, e) is denoted as w_e, then the Laplacian matrix $\boldsymbol{L}$ of $\mathcal{G}$ can be defined as $\boldsymbol{L} = \boldsymbol{B}^\top\boldsymbol{W}\boldsymbol{B} = \sum_{e\in E} w_e \boldsymbol{b}_e \boldsymbol{b}_e^\top$.

From the definition above, it is easy to prove that $\boldsymbol{L}$ is positive semi-definite. In addition, all its eigenvalues are positive except for one unique zero eigenvalue. If we denote eigenvalues and corresponding eigenvectors of $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ as $0 = \lambda_1 < \lambda_2 \le \lambda_3 \le \cdots \le \lambda_n$ and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ respectively, then $\boldsymbol{L}$ can be rewritten as $\boldsymbol{L} = \sum_{i=1}^n \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$. As $\boldsymbol{L}$ is not invertible due to its null space $\mathbf{1}$, we turn to use its pseudoinverse form, which is defined as $\boldsymbol{L}^\dagger = \sum_{i=2}^n \lambda_i^{-1} \boldsymbol{v}_i \boldsymbol{v}_i^\top$. $\boldsymbol{L}^\dagger$ appears in many quantities related to random walk, such as Kemeny constant, Kirchhoff index and betweenness measures.

Moreover, Laplacian matrix has some useful properties. According to [9], we have the following fact.

Fact 2.1. *For a connected weighted undirected graph $\mathcal{G} = (V, E, w)$ with n nodes, where $w_{\min}$ denotes the minimum edge weith of gr, let $\boldsymbol{L}$ denote the Laplacian matrix of $\mathcal{G}$. Then for any nonempty set $S \subseteq V$,*

$$\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right) \le n^2 w_{\min}^{-1}$$

2.3 Random Walk on Graphs

For a connected weighted graph $\mathcal{G}$ with n nodes, the classical random walk model of $\mathcal{G}$ can be described by the transition matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$: at any time step, the walker located at node i moves to node j with probability $\boldsymbol{P}_{[i,j]} = \boldsymbol{A}_{[i,j]}/\boldsymbol{d}_i$. It is easy to verify that

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A} \tag{1}$$

If $\mathcal{G}$ is non-bipartite and finite, there exists a unique stationary distribution of the random walk

$$\boldsymbol{\pi} = \begin{pmatrix}\pi_1 & \pi_2 & \ldots & \pi_n\end{pmatrix}^\top = \begin{pmatrix}\frac{\boldsymbol{d}_1}{d} & \frac{\boldsymbol{d}_2}{d} & \ldots & \frac{\boldsymbol{d}_n}{d}\end{pmatrix}^\top$$

where $d = \sum_{i=1}^n \boldsymbol{d}_i$.

Let $X_{u,t}^r$ denote the location of random walker r starting from node u at time step t, then we can give the definition of hitting time.

The node hitting time $T_{u,v}$ from node u to node v is the expected number of time steps for a walker starting from u to visit node v for the first time.

Definition 2.2 (Hitting time).

$$T_{u,v}^r \overset{\text{def}}{=} \min\left\{t : X_{u,t} = v, t \ge 0\right\}, T_{u,v} \overset{\text{def}}{=} \mathbb{E}\left[T_{u,v}^r\right]$$

where the r at the superscript denotes ${r}^{\text{th}}$ independent random walk.

Hitting time is a fundamental quantity in random walks, which is also known as absorbing length. Many interesting quantities can be further obtained from hitting time, including Kemeny constant $\mathcal{K}$, absorbing random-walk centrality H_u and random detour time $D_{i,j}(u)$.

For a connected graph $\mathcal{G} = (V, E)$, its Kemeny constant $kcal$ is defined as the expected hitting time for a random walker who starts from an arbitary node i to node j which is selected according to the stationary distribution $\boldsymbol{\pi}$.

Definition 2.3 (Kemeny constant).

$$\mathcal{K} \stackrel{\text{def}}{=} \sum_{j \in V} \boldsymbol{\pi}_j T_{u,v}, \forall u \in V$$

For node u in a connected graph $\mathcal{G} = (V, E)$, its absorbing random-walk centrality H_u is defined as the expected hitting time of a walker who starts from node i for node u, where node i is selected according to the stationary distribution $\boldsymbol{\pi}$. To distinguish from the newly defined centrality in Section 3.2, we denote H_u as *Single Absorbing Node Centrality*(SANC).

Definition 2.4 (SANC).

$$H_u = \sum_{i \in V} \boldsymbol{\pi}_i T_{i,u}$$

Subsequently, for graph $\mathcal{G}$, its random detour time $D_{i,j}(u)$ is defined as the expected time of a walker who starts from node i, must visit node u, then first reaches node j, where $i, u, j \in V$ differs from each other.

Definition 2.5 (Random detour time).

$$D_{i,j}(u) \stackrel{\text{def}}{=} T_{i,u} + T_{u,j}$$

It is intuitive that the larger $D_{i,j}(u)$ is, the harder it is for a walker to reach node u, then less important u is in the whole network. Actually, previous research has confirmed this suspicion, establishing connection among Kemeny constant $\mathcal{K}$, SANC H_u and random detour time $D_{i,j}(u)$.

2.4 Supermodular Functions

Subsequently, we give the definitions of monotone and supermodular set functions. For simplicity, we denote $S \cup \{u\}$ as $S + u$.

Definition 2.6 (Monotonicity). *The set function $f : 2^V \to \mathbb{R}^+$ is monotone if and only if $\forall X \subseteq Y \subseteq V$ satisfies $f(X) \geq f(Y)$.*

Definition 2.7 (Supermodularity). *The set function $f : 2^V \to \mathbb{R}^+$ is supermodular if and only if $\forall X \subseteq Y \subseteq V, u \in V \setminus Y$ satisfies $f(X) - f(X + u) \geq f(Y) - f(Y + u)$.*

3 PROBLEM FORMULATION

3.1 Connections among quantities related to random walk

After proposing definitions in Section 2.3, it is intuitive that the larger $D_{i,j}(u)$ is, the harder it is for a walker to reach node u, then less important u is in the whole network. Actually, it is easy to prove the following theorem, establishing connection among Kemeny constant $\mathcal{K}$, SANC H_u and random detour time $D_{i,j}(u)$.

Theorem 3.1. *For an arbitary node u in a connected graph $\mathcal{G} = (V, E)$ with n nodes,*

$$H_u + \mathcal{K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{\pi}_i \boldsymbol{\pi}_j D_{i,j}(u) \tag{2}$$

Proof. According to definitions in Section 2.3, the right side of (2) can be rewritten as

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{\pi}_i \boldsymbol{\pi}_j D_{i,j}(u) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{\pi}_i \boldsymbol{\pi}_j \left(T_{i,u} + T_{u,j}\right) \\ &= \sum_{i=1}^{n} \boldsymbol{\pi}_i T_{i,u} + \sum_{j=1}^{n} \boldsymbol{\pi}_j T_{u,j} \\ &= H_u + \mathcal{K} \end{aligned}$$

where the first equality is due to Definition 2.5, the last equality is due to Definition 2.4 and Definition 2.3. □

Inspired by this single-node version of theorem, we try to extend it to the case of multiple nodes. To begin with, we need to extend Definition 2.2 and Definition 2.5 to multi-node case.

Similarly, for node set S in a connected graph $\mathcal{G} = (V, E)$, the group hitting time $T_{u,S}$ from node u to node set S is the expected number of time steps for a walker starting from u to visit an arbitary node of S for the first time.

Definition 3.1 (Group hitting time).

$$T^r_{u,S} \stackrel{\text{def}}{=} \min\left\{t : X_{u,t} \in S, t \geq 0\right\}, T_{u,S} \stackrel{\text{def}}{=} \mathbb{E}\left[T^r_{u,S}\right]$$

For node set S in graph $\mathcal{G}$, the group random detour time is defined as the expected time of a walker who starts from node i, must visit an arbitary node in set S, then first reaches node j.

Definition 3.2 (Group random detour time). *If we denote the probability that a walker starting from node i first reaches node u in absorbing set S as the $(i,u)^{\text{th}}$ entry of matrix $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n \times |S|}$, then we have*

$$D_{i,j}(S) \stackrel{\text{def}}{=} T_{i,S} + \sum_{k=1}^{|S|} \boldsymbol{\mathcal{P}}[i,k] T_{k,j}$$

It is clear that when S contains only one node v, group hitting time $T_{u,S}$ and group random detour time $D_{i,j}(S)$ automatically reduces to hitting time $T_{u,v}$ and random detour time $D_{i,j}(v)$. It prompts us to similarly extend Theorem 3.1 to multi-node case.

Theorem 3.2. *For an arbitary node subset $S \subseteq V$ in a connected graph $\mathcal{G} = (V, E)$ with n nodes,*

$$\sum_{i=1}^{n} \boldsymbol{\pi}_i T_{i,S} + \mathcal{K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{\pi}_i \boldsymbol{\pi}_j D_{i,j}(S) \tag{3}$$

Proof. To prove (3), we utilize the fundamental matrix $\boldsymbol{F}^*$ in the non-absorbing random walk. According to [1], $\boldsymbol{F}^*$ can be represented as $\boldsymbol{F}^* = (\boldsymbol{I} - \boldsymbol{P} + \mathbf{1}\boldsymbol{\pi}^\top)^{-1}\boldsymbol{\Pi}^{-1}$. Subsequently, the hitting time $H_{i,j}$ can be represented by $\boldsymbol{F}^*$ as $H_{i,j} = \boldsymbol{F}^*_{[j,j]} - \boldsymbol{F}^*_{[i,j]}$. If we denote the probability that a walker starting from node i first reaches node u in absorbing set S as the $(i,u)^{\text{th}}$ entry of matrix $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n \times |S|}$, then we can use $\boldsymbol{F}^*$ to represent the weighted average of random

detour time as

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\pi}_i\boldsymbol{\pi}_j D_{i,j}(S) &= \sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,S} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{|S|}\boldsymbol{\pi}_i\boldsymbol{\pi}_j\boldsymbol{\mathcal{P}}_{[i,k]}T_{k,j}\\ &= \sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,S} + \sum_{j=1}^{n}\boldsymbol{\pi}_j\boldsymbol{\pi}^\top\boldsymbol{\mathcal{P}}(\boldsymbol{F}^*_{[j,j]}\mathbf{1}-\boldsymbol{F}^*_{[S,j]})\\ &= \sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,S} + \sum_{j=1}^{n}\boldsymbol{\pi}_j\boldsymbol{F}^*_{[j,j]}\boldsymbol{\pi}^\top\boldsymbol{\mathcal{P}}\mathbf{1} - \boldsymbol{\pi}^\top\boldsymbol{\mathcal{P}}\boldsymbol{F}^*_{[S,:]}\boldsymbol{\pi}\\ &= \sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,S} + \sum_{j=1}^{n}\boldsymbol{\pi}_j\boldsymbol{F}^*_{[j,j]} - 1\end{aligned} \tag{4}$$

where the last equality is due to the fact that $\boldsymbol{\mathcal{P}}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{F}^*\boldsymbol{\pi} = \mathbf{1}$.

Afterwards, we are able to rewrite Kemeny constant $\mathcal{K}$ as

$$\begin{aligned}\mathcal{K} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\pi}_i\boldsymbol{\pi}_j T_{i,j}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\pi}_i\boldsymbol{\pi}_j\left(\boldsymbol{F}^*_{[j,j]} - \boldsymbol{F}^*_{[i,j]}\right)\\ &= \sum_{j=1}^{n}\boldsymbol{\pi}_j\boldsymbol{F}^*_{[j,j]} - 1\end{aligned} \tag{5}$$

where the last equality is due to the fact that $\boldsymbol{F}^*\boldsymbol{\pi} = \mathbf{1}$ and $\boldsymbol{\pi}^\top\mathbf{1} = 1$. Combining (4) and (5) completes our proof. □

Since $D_{i,j}(S)$ is a multi-node extension from $D_{i,j}(u)$, it is natrual to view the term $\sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,S}$ in (3) as the multi-node extension of SANC $H_u = \sum_{i=1}^{n}\boldsymbol{\pi}_i T_{i,u}$, which means that it can be defined as a new form of group node centrality.

3.2 Definition of MANC and Its Minimization Problem

In this section, we define Multiple Absorbing Node Centrality (MANC) $M(S)$ as the weighted average of the hitting times of absorbing random walks from all nodes in the graph to S.

It is obvious that when S contains only one node v, MANC $M(S)$ automatically reduces to Single Absorbing Node Centrality (SANC) H_v.

Definition 3.3 (MANC).

$$M(S) \overset{\text{def}}{=} \sum_{u\in V}\boldsymbol{\pi}_u T_{u,S} = \sum_{u\in V}\boldsymbol{\pi}_u\mathbb{E}\left[\min\left\{t : X^r_{u,t}\in S, t\geq 0\right\}\right]$$

The definition of MANC naturally raises the problem of minimizing MANC subject to a cardinality constraint.

Problem 1 (Multiple Absorbing Node Centrality Minimization, MANCM). *Given a connected undirected graph $\mathcal{G} = (V, E, w)$ with n nodes, m edges and an integer $k \in [1, n]$, the goal is to find a node set $S^* \subseteq V$ such that MANC $M(S^*)$ is minimized:*

$$S^* = \underset{S\subseteq V, |S|=k}{\arg\max}\ M(S)$$

According to [6], we are able to transfrom the definition of MANC into the inverse of SDDM matrix.

Fact 3.3. *Given the absorbing node set S and transition matrix $\boldsymbol{P}$, then the fundamental matrix $\boldsymbol{F}$ can be denoted as*

$$\boldsymbol{F} = \sum_{l=0}^{\infty}\boldsymbol{P}^l_{-S} = (\boldsymbol{I}-\boldsymbol{P}_{-S})^{-1} = (\boldsymbol{I}-\boldsymbol{P})^{-1}_{-S} \tag{6}$$

According to (6), the entry (i, j) of $\boldsymbol{F}$ can be represented as the expected number of passages through node j by the random walker starting from node i before being absorbed. From the linearity of mean, it follows that the hitting time of a random walker is equivalent to the sum of the expected number of passages through all nodes in the graph, i.e. $\boldsymbol{l} = \boldsymbol{F}\mathbf{1}$, where $\boldsymbol{l}_i$ denotes the hitting time of a random walker starting from node i. In particular, if the random walker starts from absorbing node, the hitting time can be regarded as 0. Considering the above case, MANC $M(S)$ can be written as

$$M(S) = \boldsymbol{\pi}^\top_{-S}\boldsymbol{l} = \boldsymbol{\pi}^\top_{-S}\boldsymbol{F}\mathbf{1} = \boldsymbol{\pi}^\top_{-S}(\boldsymbol{I}-\boldsymbol{P})^{-1}_{-S}\mathbf{1}$$

Furthermore, after simplification using (1), we get the formula of $M(S)$:

$$\begin{aligned}M(S) &= \boldsymbol{\pi}^\top_{-S}(\boldsymbol{I}-\boldsymbol{P})^{-1}_{-S}\mathbf{1}\\ &= \boldsymbol{\pi}^\top_{-S}\left(\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{-1}_{-S}\mathbf{1}\\ &= \boldsymbol{\pi}^\top_{-S}\left(\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{-1}_{-S}\boldsymbol{D}^{-1}_{-S}\boldsymbol{D}_{-S}\mathbf{1}\\ &= \boldsymbol{\pi}^\top_{-S}(\boldsymbol{D}-\boldsymbol{A})^{-1}_{-S}\boldsymbol{d}_{-S}\\ &= \boldsymbol{\pi}^\top_{-S}\boldsymbol{L}^{-1}_{-S}\boldsymbol{d}_{-S}\end{aligned} \tag{7}$$

3.3 Properties of MANC

After proposing physic explanations of MANC, we attempt to study the properties of them. First, we prove that MANCM is NP-hard. Subsequently, we prove that the set function $M(\cdot)$ is monotone and supermodular, which provides us with a theorecical basis for designing greedy algorithms with approximation guarantees.

3.3.1 NP-hard.

To prove the NP-hardness of MANCM, we first give the detailed definition of vertex cover and a decision version of MANCM, trying to construct a reduction from the former to the latter on c-regular graphs.

Problem 2 (Vertex Cover on c-regular graphs). *Given a connected c-regular graph $G = (V, E)$ and an integer k, the goal is to find out whether there exists a node set $S \subseteq V$ such that $|S| \leq k$ and every edge in E is incident with at least one node in S.*

Problem 3 (Multiple Absorbing Node Centrality Minimization, Decision Version, MANCMD). *Given a connected weighted graph $\mathcal{G} = (V, E, w)$, a real number b and an interger k. The goal is to find out whether there exists a node set $S \subseteq V$ such that $|S| \leq k$ and $M(S) \leq b$.*

Lemma 3.4. *Let $\mathcal{G} = (V, E, w)$ be a connected c-regular graph graph with all edge weights being 1. Let $S \subseteq V$ be a nonempty node set with capacity k, then $M(S) \geq (n-k)/n$ and the equality holds if and only if S is a vertex cover of $\mathcal{G}$.*

Proof. We first prove that if S is a vertex cover of $\mathcal{G}$, then $M(S) = (n-k)/n$. When S is a vertex cover of $\mathcal{G}$, because there

are no edges between nodes in $V \setminus S$, we can simplify (7) as

$$M(S) = \boldsymbol{\pi}_{-S}^{\top} \boldsymbol{L}_{-S}^{-1} \boldsymbol{d}_{-S} = \boldsymbol{\pi}_{-S}^{\top} \boldsymbol{D}_{-S}^{-1} \boldsymbol{d}_{-S} = \boldsymbol{\pi}_{-S}^{\top} \mathbf{1}$$

Since $\mathcal{G}$ is a c-regular graph, $\boldsymbol{\pi} = \begin{pmatrix} 1/n & \cdots & 1/n \end{pmatrix}^{\top}$. Thus, $M(S) = \boldsymbol{\pi}_{-S}^{\top} \mathbf{1} = (n-k)/n$.

We then prove that if S is not a vertex cover of $\mathcal{G}$, then $M(S) > (n-k)/n$. For node $u \in S$ and node $v \in V \setminus S$, it is obvious that $\Pr(T^r_{u,S} = 0) = \Pr(T^r_{v,S} \neq 0) = 1$. Meanwhile, because S is not a vertex cover of $\mathcal{G}$, edge set E has at least one edge (u', v') that is not covered by S. Let $d_{\max} = \max\{d_u | u \in V\}$,$v \in V \setminus (S + u' + v')$, then we get

$$\Pr(T^r_{v,S} = 1) - \Pr(T^r_{u',S} = 1) > \frac{1}{d_{\max}} > \frac{1}{n}$$

$$\Pr(T^r_{v,S} = 1) - \Pr(T^r_{v',S} = 1) > \frac{1}{d_{\max}} > \frac{1}{n}$$

Therefore, we are able to give a lower bound of hitting time for nodes u',v' and node $v \in V \setminus (S + u' + v')$.

$$\mathbb{E}\left[T^r_{u,S}\right] \geq 1$$

$$\mathbb{E}\left[T^r_{u',S}\right] = \mathbb{E}\left[T^r_{v',S}\right] \geq \left(1 - \frac{1}{n}\right) + 2 \times \frac{1}{n} = 1 + \frac{1}{n}$$

Therefore, we finally get the lower bound of MANC:

$$\begin{aligned} M(S) &= \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[T^r_{v,S}\right] \\ &= \sum_{v \in V \setminus (S+u'+v')} \boldsymbol{\pi}_v \mathbb{E}\left[T^r_{v,S}\right] + \boldsymbol{\pi}_{u'} \mathbb{E}\left[T^r_{u',S}\right] + \boldsymbol{\pi}_{v'} \mathbb{E}\left[T^r_{v',S}\right] \\ &\geq \boldsymbol{\pi}_{-(S+u'+v')} \mathbf{1} + \boldsymbol{\pi}_{u'} \left(1 + \frac{1}{n}\right) + \boldsymbol{\pi}_{v'} \left(1 + \frac{1}{n}\right) \\ &= \boldsymbol{\pi}_{-S}^{\top} \mathbf{1} + \frac{\boldsymbol{\pi}_{u'} + \boldsymbol{\pi}_{v'}}{n} > \boldsymbol{\pi}_{-S}^{\top} \mathbf{1} = (n-k)/n \end{aligned}$$

□

After accomplishing the proof of Lemma 3.4, we prove that MANCM is NP-hard based on it.

Theorem 3.5. *MANCM is NP-hard.*

Proof. we prove the NP-hardness of MANCM by constructing a polynomial reduction from Vertex Cover to MANCMD.

$$p : \{(G = (V, E), k)\} \rightarrow \{(\mathcal{G} = (V, E, w), k, b)\}$$

For connected graph $G = (V, E)$ with n nodes, we construct a new graph $G' = (V, E, w')$, where the node set and edge set are identical to those in G, and the weight function on the edge set is defined as $w' : E \rightarrow \{1\}$, i.e. the weights of all edges are equal to 1. Then we can construct a reduction as

$$p((G = (V, E), k)) = (G' = (V, E, w'), k, \boldsymbol{\pi}_{-S})$$

According to Lemma 3.4, p is a polynomial reduction from Vertex Cover to MANCMD, which completes our proof. □

3.3.2 *Monotonicity and Supermodularity.*

In this part, we prove that the objective function $M(\cdot)$ is monotone and supermodular.

Theorem 3.6 (Monotonicity). *Let S be an arbitary nonempty node set of connected weighted graph $\mathcal{G} = (V, E, w)$, then for node $u \in V \setminus S$,*

$$M(S) \geq M(S + u)$$

Proof. According to Definition 3.3, $M(S) = \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[T^r_{v,S}\right]$. Since S is a subset of $S+u$, when a random walker reaches nodes in S, it must have reached nodes in $S + u$, i.e. $T^r_{v,S} \geq T^r_{v,S+u}$. Therefore $\mathbb{E}\left[T^r_{v,S}\right] \geq \mathbb{E}\left[T^r_{v,S+u}\right]$ holds for any node $v \in V$, which completes our proof. □

Theorem 3.7 (Supermodularity). *Let S, T be arbitary nonempty node sets of connected weighted graph $\mathcal{G} = (V, E, w)$ such that $S \subseteq T \subsetneq V$, then for node $u \in V \setminus T$,*

$$M(S) - M(S + u) \geq M(T) - M(T + u)$$

Proof. For a subset A of the probability space, we denote $\mathbf{1}_A$ as the indicator function of the event A. Then we rewrite $M(S) - M(S + u)$ by discussing the hitting node of random walker.

$$\begin{aligned} M(S) - M(S + u) &= \sum_{v \in V} \boldsymbol{\pi}_v \left(\mathbb{E}\left[T^r_{v,S}\right] - \mathbb{E}\left[\min\left\{T^r_{v,S}, T^r_{v,u}\right\}\right]\right) \\ &= \sum_{v \in V} \boldsymbol{\pi}_v (\mathbb{E}\left[T^r_{v,S}\right] - \mathbb{E}\left[T^r_{v,S} \mathbf{1}_{\left\{T^r_{v,S} \leq T^r_{v,u}\right\}}\right] \\ &\quad - \mathbb{E}\left[T^r_{v,S} \mathbf{1}_{\left\{T^r_{v,u} < T^r_{v,S}\right\}}\right]) \\ &= \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[(T^r_{v,S} - T^r_{v,u}) \mathbf{1}_{\left\{T^r_{v,u} < T^r_{v,S}\right\}}\right] \end{aligned}$$

Similarly, we also have

$$M(T) - M(T + u) = \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[(T^r_{v,T} - T^r_{v,u}) \mathbf{1}_{\left\{T^r_{v,u} < T^r_{v,T}\right\}}\right]$$

As is mentioned in proof of Theorem 3.6, $T^r_{v,S} \geq T^r_{v,T} \forall v \in V$. It is also obvious that $\mathbf{1}_{\{T_{v,u} - T_{v,S}\}} \geq \mathbf{1}_{\{T_{v,u} - T_{v,T}\}}$. Combining the above two inequalities, we can prove that

$$(T^r_{v,S} - T^r_{v,u}) \mathbf{1}_{\left\{T^r_{v,u} < T^r_{v,S}\right\}} \geq (T^r_{v,T} - T^r_{v,u}) \mathbf{1}_{\left\{T^r_{v,u} < T^r_{v,T}\right\}}$$

holds for any node $v \in V$, which completes our proof. □

4 SIMPLE GREEDY ALGORITHM

Due to the monotonicity and supermodularity of the set function $M(\cdot)$, there exists a simple greedy algorithm with a $1 - \frac{1}{e}$ approximate factor to MANCM [11]: At each iteration, a new absorbing node u^* is selected from the set of non-absorbing nodes $V \setminus S$, satisfying

$$\Delta(u, S) = M(S) - M(S + u)$$
$$u^* = \arg\max_{u \in V \setminus S} \{\Delta(u, S)\}$$

It is easy to find that the simple algorithm above needs to compute $\Delta(u, S)$ for each node u in the graph at each iteration, leading

to $O(n)$ matrix inverse operations. When using the naive matrix inverse algorithm with complexity $O(n^3)$, the overall complexity of the simple greedy algorithm is $O(kn^4)$. By simplifying $\Delta(u,S)$, we attempt to reduce the time complexity of greedy algorithm.

For node $u \in V \setminus S$, after adjusting the order of submatrix $\boldsymbol{L}_{-S}$, we rewrite $\boldsymbol{L}_{-S}$ as

$$\boldsymbol{L}_{-S} = \begin{pmatrix} l_u & -\boldsymbol{a}^\top \\ -\boldsymbol{b} & \boldsymbol{A} \end{pmatrix}$$

where $\boldsymbol{A}$ denotes $\boldsymbol{L}_{-(S+u)}$. According to the properties of block matrices, we can get

$$\boldsymbol{L}_{-S}^{-1} = \begin{pmatrix} t & t\boldsymbol{a}^\top \boldsymbol{A}^{-1} \\ t\boldsymbol{A}^{-1}\boldsymbol{b} & \boldsymbol{A}^{-1} + t\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{a}^\top \boldsymbol{A}^{-1} \end{pmatrix}$$

where $t = (l_u - \boldsymbol{a}^\top \boldsymbol{A}^{-1}\boldsymbol{a})^{-1}$. Therefore, when $S \neq \emptyset$, $\Delta(u,S)$ can be rewritten as

$$\begin{aligned}
&M(S) - M(S+u) \\
=&\boldsymbol{\pi}_{-S}^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S} - \boldsymbol{\pi}_{-(S+u)}^\top \boldsymbol{L}_{-(S+u)}^{-1}\boldsymbol{d}_{-(S+u)} \\
=&\begin{pmatrix} \boldsymbol{\pi}_u & \boldsymbol{\pi}_{-(S+u)}^\top \end{pmatrix} \begin{pmatrix} t & t\boldsymbol{a}^\top \boldsymbol{A}^{-1} \\ t\boldsymbol{A}^{-1}\boldsymbol{b} & \boldsymbol{A}^{-1} + t\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{a}^\top \boldsymbol{A}^{-1} \end{pmatrix} \begin{pmatrix} d_u \\ \boldsymbol{d}_{-(S+u)} \end{pmatrix} \\
&- \boldsymbol{\pi}_{-(S+u)}^\top \boldsymbol{L}_{-(S+u)}^{-1}\boldsymbol{d}_{-(S+u)} \\
=&t\boldsymbol{\pi}_u d_u + t d_u \boldsymbol{\pi}_{-(S+u)}^\top \boldsymbol{A}^{-1}\boldsymbol{b} + t\boldsymbol{\pi}_u \boldsymbol{a}^\top \boldsymbol{A}^{-1}\boldsymbol{d}_{-(S+u)} \\
&+ t\boldsymbol{\pi}_{-(S+u)}^\top \boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{a}^\top \boldsymbol{A}^{-1}\boldsymbol{d}_{-(S+u)} \\
=&\frac{1}{t}(t\boldsymbol{\pi}_u + t\boldsymbol{\pi}_{-(S+u)}^\top \boldsymbol{A}^{-1}\boldsymbol{b})(t d_u + t\boldsymbol{a}^\top \boldsymbol{A}^{-1}\boldsymbol{d}_{-(S+u)}) \\
=&\frac{(\boldsymbol{\pi}_{-S}^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_u)(\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S})}{\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_u} = \frac{(\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S})^2}{d(\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_u)}
\end{aligned} \tag{8}$$

However, when $S = \emptyset$, (8) is illegal due to the singularity of $\boldsymbol{L}$. Given that $M(\emptyset) = \infty$, we can define $\Delta(u,\emptyset)$ as $-M(S+u) = -M(\{u\})$, which is the negative of absorbing random-walk centrality for single node u. It is easy to verify that [17]

$$M(\{u\}) = d(\boldsymbol{e}_u - \boldsymbol{\pi})^\top \boldsymbol{L}^\dagger (\boldsymbol{e}_u - \boldsymbol{\pi}) \tag{9}$$

(8) and (9) illustrate that computing $\Delta(u,S) \forall u \in V \setminus S$ only requires the inverse of the same matrix $\boldsymbol{L}_{-S}$ or the pseudoinverse of the same matrix $\boldsymbol{L}$. Consequently, we are able to optimize the simple greedy algorithm as Algorithm 1, reducing time complexity from $O(kn^4)$ to $O(kn^3)$.

Afterwards, we prove that algorithm Exact has a $1 - \frac{k}{(k-1)e}$ approximate factor.

Theorem 4.1. *The algorithm $S_k = \textsc{Exact}(\mathcal{G}, k)$ takes a connected weighted undirected graph $\mathcal{G} = (V, E, w)$ and a positive integer k, then returns a node subset S_k with capacity k. When $k > 1$, the node set S satisfies*

$$M(\{u^*\}) - M(S_k) \geq \left(1 - \frac{k}{(k-1)e}\right)\left(M(\{u^*\}) - M(S^*)\right)$$

where

$$u^* \overset{\text{def}}{=} \arg\min_{u \in V} M(\{u\}), S^* \overset{\text{def}}{=} \arg\min_{|S|=k} M(S)$$

Algorithm 1: Exact$(\mathcal{G}, k)$

Input : A connected weighted undirected graph $\mathcal{G} = (V, E, w)$; an integer $k \in [1, |V|]$
Output : S_k: A subset of V with $|S_k| = k$

```
Compute L and d
d ← d1, π ← d^{-1}d
S_1 ← {arg min_{u∈V} {d(e_u − π)^⊤ L^†(e_u − π)}}
for i = 2, 3, ..., k do
    foreach u ∈ V \ S do
        Δ(u, S) ← (e_u^⊤ L_{-S}^{-1} d_{-S})^2 / (d(e_u^⊤ L_{-S}^{-1} e_u))
    u* ← arg max_{u∈V\S} {Δ(u, S)}
    S_i ← S_{i-1} ∪ {u*}
return S_k
```

Proof. According to the supermodularity of $M(\cdot)$,

$$M(S_i) - M(S_{i+1}) \geq \frac{1}{k}\left(M(S_i) - M(S^*)\right)$$

holds for any positive integer i, which indicates that

$$M(S_{i+1}) - M(S^*) \leq \left(1 - \frac{1}{k}\right)\left(M(S_i) - M(S^*)\right)$$

Subsequently, we can further obtain that

$$\begin{aligned}
&M(S_k) - M(S^*) \\
\leq&\left(1 - \frac{1}{k}\right)^{k-1}\left(M(S_1) - M(S^*)\right) \\
\leq&\frac{k}{(k-1)e}\left(M(S_1) - M(S^*)\right)
\end{aligned}$$

which completes our proof based on the fact that $S_1 = \{u^*\}$. □

5 FASTER GREEDY ALGORITHM

Despite the time complexity optimization made by proposing Algorithm 1, this simple greedy algorithm still requires matrix inversion so that it cannot be applicated in large networks. In order to accelerate the computation of (8) and (9), we establish efficient approximations of (9), numerator in (8) and denominator in (8) by using Lemma 5.1 and Lemma 5.2.

Lemma 5.1 (Johnson-Lindenstrauss Lemma, JL Lemma [5]). *Given fixed vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n \in \mathbb{R}^d$ and $\epsilon > 0$, let $\boldsymbol{Q} \in \mathbb{R}^{k \times d}$ denote a matrix where $k \geq \lceil 24\epsilon^{-2}\log n \rceil$, each entry in $\boldsymbol{Q}$ is equal to $1/\sqrt{k}$ or $-1/\sqrt{k}$ with the same probability $1/2$. Then $\forall i, j \leq n$,*

$$\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2 \approx_\epsilon \left\|\boldsymbol{Q}\boldsymbol{v}_i - \boldsymbol{Q}\boldsymbol{v}_j\right\|^2 \tag{10}$$

Lemma 5.1 indicates that if we project n vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ into a lower dimensional space spanned by $O(\log n)$ random vectors, the distances between projected vectors will be preserved with high probability.

Lemma 5.2 (SDD Solver [2, 12]). *There exists an algorithm $\boldsymbol{x} = \textsc{Solve}(\boldsymbol{S}, \boldsymbol{b}, \delta)$ which takes a SDDM matrix or a Laplacian $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ with nnz$(\boldsymbol{S}) = m$, a vector $\boldsymbol{b} \in \mathbb{R}^n$, and an error parameter $\delta > 0$ as input, and returns a vector $\boldsymbol{x} \in \mathbb{R}^n$ which satisies*

$$\left\|\boldsymbol{x} - \boldsymbol{S}^{-1}\boldsymbol{b}\right\|_{\boldsymbol{S}} \leq \delta \left\|\boldsymbol{S}^{-1}\boldsymbol{b}\right\|_{\boldsymbol{S}}$$

with high probability.

In Lemma 5.2, $\|\boldsymbol{x}\|_{\boldsymbol{S}}$ is denoted as $\sqrt{\boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{x}}$, and $\boldsymbol{S}^{-1}$ is denoted as the pseudoinverse of $\boldsymbol{S}$ when $\boldsymbol{S}$ is a Laplacian. The expected time complexity of algorithm Solver is $\tilde{O}(m)$, where $\tilde{O}(\cdot)$ omits the poly$(\log n)$ factors.

Based on Lemma 5.1 and Lemma 5.2, we are able to efficiently approximate the quantities in (8) and (9) that are related to $\boldsymbol{L}^\dagger$ and $\boldsymbol{L}_{-S}^{-1}$.

5.1 Approximations of (9)

Given that (9) can be regarded as the Single Absorbing Node Centrality (SANC), we use an efficient algorithm SANCEst proposed by [17].

Lemma 5.3. *there exists an algorithm called SANCEst with time complexity $\tilde{O}(m)$, which takes a connected weighted undirected graph $\mathcal{G} = (V, E, w)$ and an error parameter $\epsilon > 0$ as input, and returns the approximation $\tilde{M}(\{u\})$ of SANC $M(\{u\})$ for any node u in graph $\mathcal{G}$ such that*

$$M(\{u\}) \approx_\epsilon \tilde{M}(\{u\})$$

Algorithm 2: SANCEst$(\mathcal{G}, \epsilon)$

Input : A connected weighted undirected graph $\mathcal{G} = (V, E, w)$; an error parameter $\epsilon > 0$

Output : The approximation $\tilde{M}(\{u\})$ of SANC $M(\{u\})$ for any node u in graph $\mathcal{G}$

Compute $\boldsymbol{L}$ and $\boldsymbol{d}$

$d \leftarrow \boldsymbol{d}\boldsymbol{1}, \boldsymbol{\pi} \leftarrow d^{-1}\boldsymbol{d}, n \leftarrow |V|$

$w_{\min} \leftarrow \min\{w_e | e \in E\}, w_{\max} \leftarrow \max\{w_e | e \in E\}$

$k \leftarrow \lceil 24\epsilon^{-2}\log n \rceil, \delta \leftarrow \frac{\epsilon}{6n^2}\sqrt{\frac{(1-\epsilon)w_{\min}}{(1+\epsilon)w_{\max}}}$

Construct a matrix $\boldsymbol{Q} \in \mathbb{R}^{k\times m}$, each entry of which is $\pm 1/\sqrt{k}$ with identical probability

Decompose $\boldsymbol{L}$ into $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{W} \in \mathbb{R}^{m\times m}$, where $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{W}\boldsymbol{B}$

$\overline{\boldsymbol{X}} \leftarrow \boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}$

for $i = 1, 2, \ldots, k$ **do**

  $\boldsymbol{X}'_{[i,:]} \leftarrow \textsc{Solve}(\boldsymbol{L}, \overline{\boldsymbol{X}}_{[i,:]}, \delta)$

foreach $u \in V$ **do**

  $\tilde{M}(\{u\}) \leftarrow d\|\boldsymbol{X}'(\boldsymbol{e}_u - \boldsymbol{\pi})\|^2$

return $\{\tilde{M}(\{u\}) | u \in V\}$

5.2 Approximations of (8)

Before proposing the approximations of (8), the proof of the following lemma is necessary.

Lemma 5.4. *Given a connected weighted undirected graph $\mathcal{G} = (V, E, w)$, let $\boldsymbol{L}$ and $w_{\min}$ denote the Laplacian and the minimal edge weight of the graph respectively. Then for an arbitary node set $S \subseteq V$ and an arbitary vector $\boldsymbol{v} \in \mathbb{R}^{n-|S|}$, $\boldsymbol{v}_i^2 \le n w_{\min}^{-1}\|\boldsymbol{v}\|_{\boldsymbol{L}_{-S}}^2$ holds.*

Proof. As is mentioned in Section 2.2, we can decompose Laplacian matrix as $\boldsymbol{L} = \boldsymbol{B}^\top\boldsymbol{W}\boldsymbol{B}$. Similarly, we can decompose SDDM matrix $\boldsymbol{L}_{-S}$ as $\boldsymbol{L}_{-S} = \boldsymbol{B}'^\top\boldsymbol{W}'\boldsymbol{B}' + \boldsymbol{Z}$. If $\boldsymbol{Z}_{[i,i]} \neq 0$, then we have $\boldsymbol{Z}_{[i,i]} \ge w_{\min}$, it is apparent that $\boldsymbol{v}_i^2 \le n w_{\min}^{-1}\|\boldsymbol{v}\|_{\boldsymbol{L}_{-S}}^2$. If $\boldsymbol{Z}_{[i,i]} = 0$, there must exist a node j that satisfies $\boldsymbol{Z}_{[j,j]} \ge w_{\min}$ in the component of graph that contains node i after removing nodes in S. Let $\mathscr{P}_{ij}$ denotes the simple path connecting node i and node j, then we have

$$\begin{aligned}\|\boldsymbol{v}\|_{\boldsymbol{L}_{-S}}^2 &\ge w_{\min}\sum_{\langle a,b\rangle\in\mathscr{P}_{ij}}(\boldsymbol{v}_a - \boldsymbol{v}_b)^2 + w_{\min}\boldsymbol{v}_j^2 \\ &\ge w_{\min}n^{-1}\left(\sum_{\langle a,b\rangle\in\mathscr{P}_{ij}}(\boldsymbol{v}_a - \boldsymbol{v}_b) + \boldsymbol{v}_j\right)^2 \ge w_{\min}n^{-1}\boldsymbol{v}_i^2\end{aligned}$$

which completes our proof. □

We first approximate the numerator of (8), which requires the approximation of computing $\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S}$. Since $\boldsymbol{L}_{-S}$ is a SDDM matrix, we can use Lemma 5.2 to avoid computing matrix inverse $\boldsymbol{L}_{-S}^{-1}$.

Lemma 5.5. *Given a connected weighted undirected graph $\mathcal{G} = (V, E, w)$,the Laplacian submatrix $\boldsymbol{L}_{-S}$ and an error parameter $\epsilon \in (0, 1)$, let $\boldsymbol{x}' = \textsc{Solve}(\boldsymbol{L}_{-S}, \boldsymbol{d}_{-S}, \delta_1)$ where $\delta_1 = \frac{w_{\min}\epsilon}{7n^3 w_{\max}}$. Then for any node $i \in V \setminus S$, we have*

$$\boldsymbol{x}_i = \boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S} \approx_{\epsilon/7} \boldsymbol{x}'_i \tag{11}$$

Proof. Combining Lemma 5.2 and Lemma 5.4, we can bound $(\boldsymbol{x}'_i - \boldsymbol{x}_i)^2$ as

$$\begin{aligned}(\boldsymbol{x}'_i - \boldsymbol{x}_i)^2 &\le n w_{\min}^{-1}\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|_{\boldsymbol{L}_{-S}}^2 \\ &\le n w_{\min}^{-1}\delta_1^2\|\boldsymbol{x}\|_{\boldsymbol{L}_{-S}}^2 \\ &\le n^4 w_{\min}^{-1}w_{\max}^2\delta_1^2\mathrm{Tr}(\boldsymbol{L}_{-S}) \\ &\le n^6 w_{\min}^{-2}w_{\max}^2\delta_1^2\end{aligned}$$

where the last inequality is due to Fact 2.1. On the other hand, since $\boldsymbol{x}_i = \boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S}$ denotes the expected hitting time of a random walker starting from node i, we have $\boldsymbol{x}_i \ge 1$. Finally, we are able to give the approximation of $\boldsymbol{x}_i$ as

$$\frac{\left|\boldsymbol{x}'_i - \boldsymbol{x}_i\right|}{\boldsymbol{x}_i} \le n^3 w_{\max}w_{\min}^{-1}\delta_1 = \epsilon/7$$

□

Subsequently, we attempt to approximate the denominator of (8), which can be recast in an euclidian norm by using Lemma 5.1:

$$\begin{aligned}&\boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i \\ =&\boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}(\boldsymbol{B}'^\top\boldsymbol{W}'\boldsymbol{B}' + \boldsymbol{Z})\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i \\ =&\boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{B}'^\top\boldsymbol{W}'\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i + \boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{Z}\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i \\ =&\left\|\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2 + \left\|\boldsymbol{Z}^{1/2}\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2\end{aligned} \tag{12}$$

Although (12) indicates that we can avoid matrix inversion by using Lemma 5.2, due to the dimension of the matrices $\boldsymbol{W}'$ and $\boldsymbol{Z}$ being $|E| = m$ and $|V| = n$ respectively, the number of calls to Solve is still unacceptable. Therefore, we need to use Lemma 5.1 to reduce the dimension of the corresponding matrices.

From Lemma 5.1, let $\boldsymbol{Q} \in \mathbb{R}^{q\times m}, \boldsymbol{R} \in \mathbb{R}^{r\times n}$ denote the projection matrix constructed according to the definition of Lemma 5.1, where $q = r = \lceil 24\epsilon^{-2}\log n\rceil$, then we have

$$\begin{aligned}&\left\|\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2 + \left\|\boldsymbol{Z}^{1/2}\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2\\ \approx_\epsilon &\left\|\boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2 + \left\|\boldsymbol{R}\boldsymbol{Z}^{1/2}\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i\right\|^2\end{aligned} \tag{13}$$

For the convenience of writing, we denote $\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}$ and $\boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S}^{-1}$ as $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ respectively, denote $\boldsymbol{Z}^{1/2}\boldsymbol{L}_{-S}^{-1}$ and $\boldsymbol{R}\boldsymbol{Z}^{1/2}\boldsymbol{L}_{-S}^{-1}$ as $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$ respectively. Then (12) and (13) can be rewritten as

$$\boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i = \|\boldsymbol{X}\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}\boldsymbol{e}_i\|^2 \tag{14}$$

$$\|\boldsymbol{X}\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}\boldsymbol{e}_i\|^2 \approx_\epsilon \left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 + \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2 \tag{15}$$

Combining (14) and (15), we can efficiently approximate the denominator of (8).

Lemma 5.6. *Given a connected weighted undirected graph $\mathcal{G} = (V, E, w)$,the Laplacian submatrix $\boldsymbol{L}_{-S}$ and an error parameter $\epsilon \in (0,1)$. Let $\overline{\boldsymbol{X}}$ and $\overline{\boldsymbol{Y}}$ denote $\boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}$ and $\boldsymbol{R}\boldsymbol{Z}^{1/2}$ respectively. Let $\boldsymbol{X}' \in \mathbb{R}^{q\times n}$ and $\boldsymbol{Y}' \in \mathbb{R}^{r\times n}$ be matrices such that $\boldsymbol{X}'_{[i,:]} = \textsc{Solve}(\boldsymbol{L}_{-S}, \overline{\boldsymbol{X}}_{[i,:]}, \delta_2)$, $\boldsymbol{Y}'_{[i,:]} = \textsc{Solve}(\boldsymbol{L}_{-S}, \overline{\boldsymbol{Y}}_{[i,:]}, \delta_2)$, where $q = r = \lceil 24(\epsilon/7)^{-1}\rceil \log n$,$\delta_2 = \frac{w_{\min}\epsilon}{31n^2}\sqrt{\frac{1-\epsilon/7}{2w_{\max}(1+\epsilon/7)}}$. Then for any node $i \in V \setminus S$, we have*

$$\|\boldsymbol{X}\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}\boldsymbol{e}_i\|^2 = \boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i \approx_{\epsilon/3} \left\|\boldsymbol{X}'\boldsymbol{e}_i\right\|^2 + \left\|\boldsymbol{Y}'\boldsymbol{e}_i\right\|^2 \tag{16}$$

Proof. According to triangle inequality, we have

$$\begin{aligned}&\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\| - \left\|\boldsymbol{X}'\boldsymbol{e}_i\right\|\right| \le \left\|(\tilde{\boldsymbol{X}} - \boldsymbol{X})\boldsymbol{e}_i\right\| \le \left\|\tilde{\boldsymbol{X}} - \boldsymbol{X}'\right\|_F\\ =&\sqrt{\sum_{j=1}^{q}\left\|\tilde{\boldsymbol{X}}_{[j,:]} - \boldsymbol{X}'_{[j,:]}\right\|^2} \le \sqrt{\sum_{j=1}^{q} n w_{\min}^{-1}\left\|\tilde{\boldsymbol{X}}_{[j,:]} - \boldsymbol{X}'_{[j,:]}\right\|_{\boldsymbol{L}_{-S}}^2}\\ \le&\sqrt{\sum_{j=1}^{q} n w_{\min}^{-1}\delta_2^2\left\|\tilde{\boldsymbol{X}}_{[j,:]}\right\|_{\boldsymbol{L}_{-S}}^2} \le n\delta_2\sqrt{w_{\min}^{-1}}\left\|\tilde{\boldsymbol{X}}\right\|_F\end{aligned}$$

where the third inequality and the fourth inequality are due to Lemma 5.4 and Lemma 5.2 respectively.

As $q = r = \lceil 24(\epsilon/7)^{-1}\rceil \log n$, by using Lemma 5.1, we are able to further obtain

$$\begin{aligned}&n\delta_2\sqrt{w_{\min}^{-1}}\left\|\tilde{\boldsymbol{X}}\right\|_F \le n\delta_2\sqrt{w_{\min}^{-1}(1+\epsilon/7)\sum_{i\in V\setminus S}\|\boldsymbol{X}\boldsymbol{e}_i\|^2}\\ \le&n\delta_2\sqrt{w_{\min}^{-1}(1+\epsilon/7)\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)} \le n^2\delta_2 w_{\min}^{-1}\sqrt{1+\epsilon/7}\end{aligned}$$

where the last inequality is due to Fact 2.1. Similarly, for $\left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\| - \|\boldsymbol{Y}'\boldsymbol{e}_i\|\right|$, we have

$$\left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\| - \left\|\boldsymbol{Y}'\boldsymbol{e}_i\right\|\right| \le n^2\delta_2 w_{\min}^{-1}\sqrt{1+\epsilon/7} \tag{17}$$

On the other hand, according to Lemma 5.1,

$$\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 + \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2 \ge (1-\epsilon/7)\boldsymbol{e}_i^\top\boldsymbol{L}_{-S}^{-1}\boldsymbol{e}_i \ge (1-\epsilon/7)n^{-2}w_{\max}^{-1}$$

where the last inequality is due to the scaling of $\boldsymbol{e}_i^\top\boldsymbol{L}_{-S}^{-1}\boldsymbol{d}_{-S} \ge 1$. According to the Pigeonhole Principle, at least one element of $\left\{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|, \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|\right\}$ is no less than $\frac{1}{n}\sqrt{\frac{1-\epsilon/7}{2w_{\max}}}$. Without loss of generality, we let $\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\| \ge \frac{1}{n}\sqrt{\frac{1-\epsilon/7}{2w_{\max}}}$, then we have

$$\frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\| - \|\boldsymbol{X}'\boldsymbol{e}_i\|\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|} \le n\delta_2 w_{\min}^{-1}\sqrt{\frac{2w_{\max}(1+\epsilon/7)}{1-\epsilon/7}} = \frac{\epsilon}{31}$$

According to the inequality above, we can further obtain

$$\begin{aligned}\frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{X}'\boldsymbol{e}_i\|^2\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2} &= \frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\| - \|\boldsymbol{X}'\boldsymbol{e}_i\|\right|\left(\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\| + \|\boldsymbol{X}'\boldsymbol{e}_i\|\right)}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2}\\ &\le \frac{\epsilon}{31}\left(2+\frac{\epsilon}{31}\right) \le \frac{\epsilon}{15}\end{aligned}$$

which means

$$\frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{X}'\boldsymbol{e}_i\|^2\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 + \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2} \le \frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{X}'\boldsymbol{e}_i\|^2\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2} \le \frac{\epsilon}{15} \tag{18}$$

For the case of $\|\boldsymbol{Y}'\boldsymbol{e}_i\|^2$, as (17) still holds, we can obtain

$$\frac{\left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\| - \|\boldsymbol{Y}'\boldsymbol{e}_i\|\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|} \le n\delta_2 w_{\min}^{-1}\sqrt{\frac{2w_{\max}(1+\epsilon/7)}{1-\epsilon/7}} = \frac{\epsilon}{31}$$

thus indicating that

$$\begin{aligned}\frac{\left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2\right|}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 + \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2} &\le \frac{\left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\| - \|\boldsymbol{Y}'\boldsymbol{e}_i\|\right|\left(\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\| + \|\boldsymbol{Y}'\boldsymbol{e}_i\|\right)}{\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2}\\ &\le \frac{\epsilon}{31}\left(2+\frac{\epsilon}{31}\right) \le \frac{\epsilon}{15}\end{aligned} \tag{19}$$

Combining (18) and (19), we are finally able to get

$$\begin{aligned}&\frac{\left|\left(\|\boldsymbol{X}'\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2\right) - \left(\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 + \left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2\right)\right|}{\|\boldsymbol{X}'\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2}\\ \le&\frac{\left|\left\|\tilde{\boldsymbol{X}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{X}'\boldsymbol{e}_i\|^2\right| + \left|\left\|\tilde{\boldsymbol{Y}}\boldsymbol{e}_i\right\|^2 - \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2\right|}{\|\boldsymbol{X}'\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2} \le \frac{\epsilon}{7}\end{aligned}$$

which completes our proof in conjunction with Lemma 5.1. □

Combining Lemmas 5.5 and 5.6, we finally propose the approximate algorithm MarginEst of computing $\Delta(u,S)$ when $S \neq \emptyset$. Furthermore, we are able to provide its approximation guarantee.

Theorem 5.7. *For any real number $\epsilon \in (0,1)$, $\Delta'(u,S)$ satisfies*

$$\Delta(u,S) \approx_\epsilon \Delta'(u,S)$$

with high probability.

Combining Algorithm 2 and Algorithm 3, we can give the faster greedy algorithm Approx with a $1-\frac{k}{(k-1)e}-\epsilon$ approximate factor and nearly linear running time.

Theorem 5.8. *The algorithm $S_k = \textsc{Approx}(\mathcal{G}, k, \epsilon)$ takes a connected weighted undirected graph $\mathcal{G} = (V, E, w)$,a positive integer k and an error parameter $\epsilon \in (0,1)$, then returns a node subset S_k with capacity k. When $k > 1$, the node set S satisfies*

$$\begin{aligned}&(1+\epsilon)M\left(\{u^*\}\right) - M\left(S_k\right) \ge\\ &\left(1 - \frac{k}{(k-1)e} - \epsilon\right)\left((1+\epsilon)\,M\left(\{u^*\}\right) - M\left(S^*\right)\right)\end{aligned}$$

Algorithm 3: MarginEst$(\mathcal{G}, S, \epsilon,)$

Input : A connected weighted undirected graph $\mathcal{G} = (V, E, w)$; The absorbing node set $S \subseteq V$; an error parameter $\epsilon \in (0, 1)$

Output : The margin $\Delta(u, S)$ of MANC created by adding node u to S for any node $u \in V \setminus S$

Compute $\boldsymbol{L}$ and $\boldsymbol{d}$

$n \leftarrow |V|, m \leftarrow |E|$

$w_{\min} \leftarrow \min\{w_e | e \in E\}, w_{\max} \leftarrow \max\{w_e | e \in E\}$

$d \leftarrow \boldsymbol{d}\mathbf{1}, q, r \leftarrow \lceil 24(\epsilon/7)^{-2} \log n \rceil$

$\delta_1 \leftarrow \frac{w_{\min}\epsilon}{7n^2 w_{\max}}, \delta_2 \leftarrow \frac{w_{\min}\epsilon}{31n^2}\sqrt{\frac{1-\epsilon/7}{2w_{\max}(1+\epsilon/7)}}$

$\boldsymbol{x}' \leftarrow \text{Solve}(\boldsymbol{L}_{-S}, \boldsymbol{d}_{-S}, \delta_1)$

$\boldsymbol{Q} \leftarrow \text{GenerateRandomMatrix}(q, m)$

$\boldsymbol{R} \leftarrow \text{GenerateRandomMatrix}(r, n)$

Decompose $\boldsymbol{L}_{-S}$ into $\boldsymbol{B}' \in \mathbb{R}^{m\times n}, \boldsymbol{W}' \in \mathbb{R}^{m\times m}$ and $\boldsymbol{Z} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{L}_{-S} = \boldsymbol{B}'^{\top}\boldsymbol{W}'\boldsymbol{B}' + \boldsymbol{Z}$

$\overline{\boldsymbol{X}} \leftarrow \boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}, \overline{\boldsymbol{Y}} \leftarrow \boldsymbol{R}\boldsymbol{Z}^{1/2}$

for $i = 1, 2, \ldots, q$ **do**

  $\boldsymbol{X}'_{[i,:]} \leftarrow \text{Solve}(\boldsymbol{L}_{-S}, \overline{\boldsymbol{X}}_{[i,:]}, \delta_2)$

  $\boldsymbol{Y}'_{[i,:]} \leftarrow \text{Solve}(\boldsymbol{L}_{-S}, \overline{\boldsymbol{Y}}_{[i,:]}, \delta_2)$

foreach $u \in V \setminus S$ **do**

  $\Delta'(u, S) \leftarrow \frac{\boldsymbol{x}'^2}{\|\boldsymbol{X}'\boldsymbol{e}_i\|^2 + \|\boldsymbol{Y}'\boldsymbol{e}_i\|^2}$

return $\{\Delta'(u, S) | u \in V \setminus S\}$

Algorithm 4: Approx$(\mathcal{G}, k, \epsilon)$

Input : A connected weighted undirected graph $\mathcal{G} = (V, E, w)$; an integer $k \in [1, |V|]$; an error parameter $\epsilon \in (0, 1)$

Output : S_k: A subset of V with $|S_k| = k$

$d \leftarrow \boldsymbol{d}\mathbf{1}, \boldsymbol{\pi} \leftarrow d^{-1}\boldsymbol{d}$

$\left\{\tilde{M}(\{u\}) | u \in V\right\} \leftarrow \text{SANCEst}(\mathcal{G}, \epsilon)$

$S_1 \leftarrow \left\{\arg\min_{u\in V}\left\{\tilde{M}(\{u\})\right\}\right\}$

for $i = 2, 3, \ldots, k$ **do**

  $\{\Delta'(u, S) | u \in V \setminus S\} \leftarrow \text{MarginEst}(\mathcal{G}, S, \epsilon)$

  $u^* \leftarrow \arg\max_{u\in V\setminus S}\{\Delta'(u, S)\}$

  $S_i \leftarrow S_{i-1} \cup \{u^*\}$

return S_k

where

$$u^* \overset{\text{def}}{=} \arg\min_{u\in V} M(\{u\}), S^* \overset{\text{def}}{=} \arg\min_{|S|=k} M(S)$$

Proof. Since algorithm Approx use $\Delta'(u, S)$ computed by algorithm MarginEst instead of $\Delta(u, S)$, combining Theorem 5.7 and supermodularity, we are able to obtain that

$$M(S_i) - M(S_{i+1}) \geq \frac{1-\epsilon}{k}\left(M(S_i) - M(S^*)\right)$$

holds for any positive integer i, which indicates that

$$M(S_{i+1}) - M(S^*) \leq \left(1 - \frac{1-\epsilon}{k}\right)\left(M(S_i) - M(S^*)\right)$$

Table 1: Information of datasets, where n, m and n', m' denote the number of nodes and edges of a network and its largest connected component respectively.

Network	n	m	n'	m'
Zachary karate club	34	78	34	78
Contiguous USA	49	107	49	107
Les Misérables	77	254	77	254
Jazz musicians	198	2742	198	2742
Euroroads	1174	1417	1039	1305
Hamsterster friends	2952	12534	1788	12476
ego-Facebook	4039	88234	4039	88234
CA-GrQc	5242	14496	4158	13428
US power grid	4941	6594	4941	6594
Reactome	6327	147547	5973	146992
CA-HepTh	9877	25998	8638	24827
Sister cities	14274	20573	10320	17988
CA-HepPh	12008	118521	11204	117649
CAIDA	26475	53381	26475	53381
loc-Gowalla	196591	950327	196591	950327
com-Amazon	334863	925872	334863	925872
Dogster friends	451710	8543549	426485	8543321
roadNet-PA	1088092	1541898	1087562	1541514
roadNet-CA	1965206	2766607	1957027	2760388

Subsequently, we can further obtain that

$$\begin{aligned} & M(S_k) - M(S^*) \\ \leq & \left(1 - \frac{1-\epsilon}{k}\right)^{k-1}\left(M(S_1) - M(S^*)\right) \\ \leq & \left(\frac{k}{(k-1)e} + \epsilon\right)\left(M(S_1) - M(S^*)\right) \end{aligned}$$

which completes our proof based on the fact that $M(S_1) \leq (1+\epsilon)M(u^*)$. □

6 EXPERIMENTS

After theorecical analyses of the two approximate algorithm Exact and Approx, we attempt to verify their performance on real-world network datasets. We choose some datasets from KONECT [7] and SNAP [8]. The information of these datasets is shown in Table 1. Note that our algorithm works only for connected graphs, therefore for the originally disconnected networks, we perform numerical experiments on their largest connected components.

To facilitate calling SDD solver in Laplacians.jl*, our numerical experiment programs are implemented by Julia. We run our programs on a linux box equipped with 128GiB RAM and 3.5GHz AMD EPYC Milan CPU, using 32 threads.

6.1 Accuracy of MarginEst

Since we give the approximate guarantee for algorithm MarginEst and SANCEst above, it is necessary to calculate

*https://github.com/danspielman/Laplacians.jl

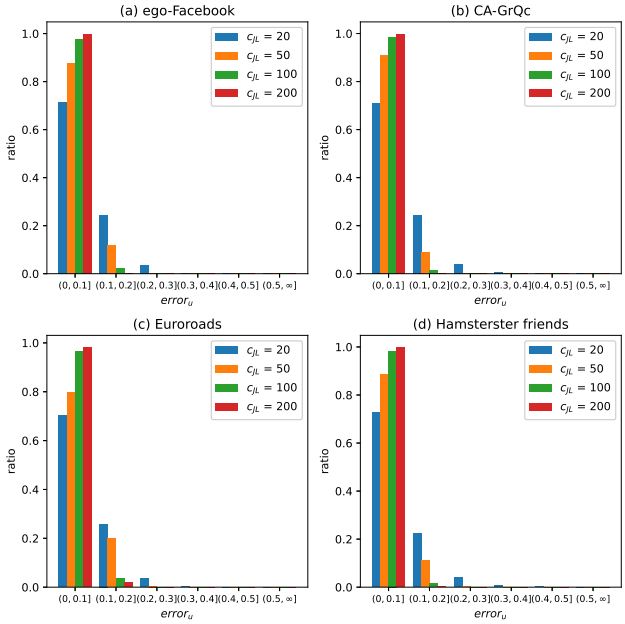

Figure 1: Relative error $error_u$ distribution of algorithm MarginEst with different c_{JL} on four networks: ego-Facebook (a), CA-GrQc (b), Euroroads (c) and Hamsterster friends (d).

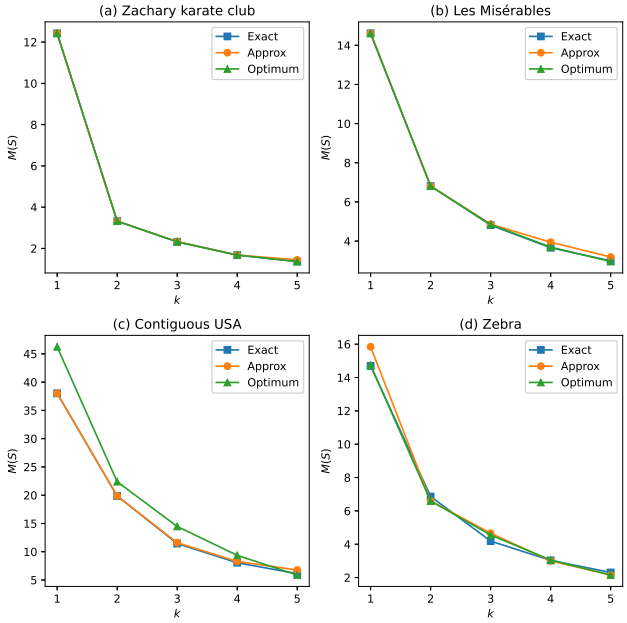

Figure 2: MANC $M(S)$ of node set S computed by three different algorithms(Exact,Approx and Optimum) on four networks: Zachary karate club (a), Les Misérables (b), Contiguous USA (c) and Zebra (d).

the approximation errors of both algorithms in real networks by numerical experiments.

The introduced relative error of approximate algorithm can be represented as

$$error_u = \frac{|\Delta'(u,S) - \Delta(u,S)|}{\Delta(u,S)}$$

which is caused by using Lemma 5.1 and Lemma 5.2. To intuitively demonstrate the relationship between the error and the dimension of the projection matrix in Lemma 5.1, we use quantity c_{JL} to denote $q/\log n$ in MarginEst. We run MarginEst with various c_{JL} on different real networks (Euroroads, Hamsterster friends, ego-Facebook and CA-GrQc), setting absorbing set $S = \{u^*\}$ where

$$u^* = \arg\min_{u \in V} M(\{u\})$$

For each c_{JL}, we compute $error_u$ for each node $u \neq u^*$, then draw the distribution of them in Figure 1. We observe that almost all of the relative errors are in the interval $(0, 0.2]$ when $c_{JL} \geq 50$, so we set parameter $c_{JL} = 50$ in the following experiments, whose results will show that this parameter is sufficient to get approximate solution.

6.2 Performance of Exact and Approx

Subsequently, we show the performance of proposed algorithms by comparing them with optimum solution and two other algorithms: Top-SANC and Top-Degree when solving MANC minimization problem.

We first compare the performance of Exact and Approx with optimum solution on four small networks. For each network $\mathcal{G} = (V, E)$, we find the k-element set with minimum MANC value by enumerating all the k-element subsets of V, then compare the minimum MANC value with solution given by Exact and Approx, whose results is shown in Figure 2. Figure 2 demonstrates that the solutions given by our greedy algorithms are almost the same with each other, both of them are also quite close to the optimum solution, which indicates that the approximation ratio of our proposed algorithms are far better than the theorecical guarantees.

Top-SANC simply chooses k absorbing nodes with minimum SANC value which is defined in Section 5.1, while Top-Degree chooses them with biggest degrees. We run these four algorithms on four different real networks, the results is shown in Figure 3. Figure 3 demonstrates that both of our algorithms also get similar approximate solutions in larger networks, which outperform the solutions of other algorithms.

6.3 Running time of Exact and Approx

Finally, we prove that algorithm Approx is much more efficient than algorithm Exact, especially when applicated on large networks. We test both algorithms on a larger set of real networks,. For each network, we use Exact and Approx separately to solve MANC minimization problem, setting $k = 10$. The running time of both algorithms is listed in Table 2. From Table 2, we can observe that the running time of Approx is proportional to the number of edges in the network, thus leading to an increase in the speedup ratio of Approx compared to Exactas the network size grows. In

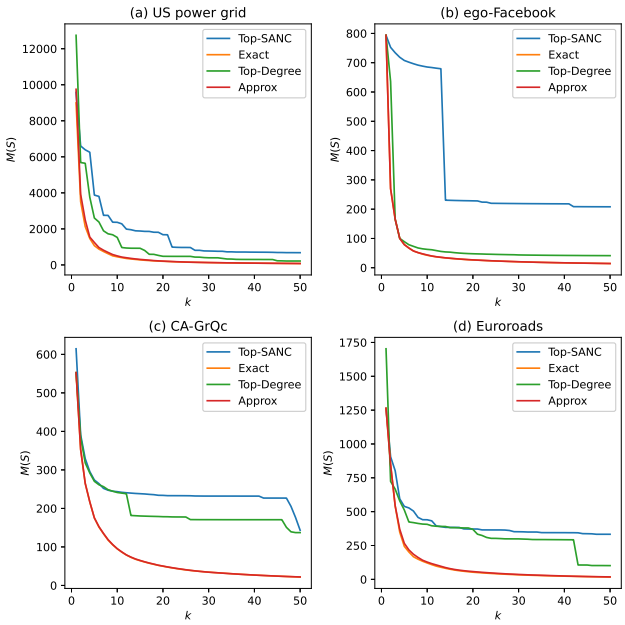

Figure 3: MANC $M(S)$ of node set S computed by four different algorithms(Exact,Approx,Top-SANC and Top-Degree) on four networks: US power grid (a), ego-Facebook (b), CA-GrQc (c) and Euroroads (d).

Table 2: The average running times and corresponding speedup ratio of Exact and Approx on a larger set of real networks.

Network	Time (seconds)		Speedup ratio
	Exact	Approx	
Euroroads	0.864	0.509	1.697
Hamsterster friends	2.679	1.133	2.365
ego-Facebook	39.499	6.150	6.423
CA-GrQc	42.611	1.515	28.126
US power grid	74.966	1.530	48.997
Reactome	139.377	10.426	13.368
CA-HepTh	392.903	4.050	97.013
Sister cities	667.902	8.362	79.873
CA-HepPh	-	23.377	-
CAIDA	-	23.705	-
loc-Gowalla	-	543.582	-
com-Amazon	-	1643.860	-
Dogster friends	-	3702.121	-
roadNet-PA	-	8556.227	-
roadNet-CA	-	17876.751	-

addition, Table 2 indicates that Approx is still usable when dealing with networks with millions of nodes, while Exact fails because of its high time complexity.

7 CONCLUSIONS

In this paper, we took the definition of hitting time of absorbing random walk as a basis, and extended it to the case to multiple nodes, i.e, Multiple Absorbing Node Centrality (MANC). For a connected weighted undirected graph with n nodes and m edges, MANC $M(S)$ of the node set S is defined as the weighted average of the hitting times of absorbing random walks from all nodes in the graph to S. Furthermore, we constructed the problem of finding the node set S^* with capacity k that minimizes $M(S^*)$. We proved that this problem is NP-hard, and the objective function is monotone and supermodular. Due to these properties of MANC, we designed two approximate greedy algorithm, the former algorithm has a $1-\frac{1}{e}$ approximate factor and $O(kn^3)$ time complexity, while the latter algorithm obtains a $1-\frac{1}{e}-\epsilon$ approximate factor and runs in time $\tilde{O}(km)$. Numerical experiments on real-world networks illustrate that both of the algorithms can provide solutions that are quite close to the optimal. Specifically, numerical experiments on large networks incidate that the latter algorithm Approx is still well scalable while maintaining the approximation error and can be applicated in large networks with millions of nodes.

REFERENCES

[1] Daniel Boley, Gyan Ranjan, and Zhi-Li Zhang. 2011. Commute times for a directed graph using an asymmetric Laplacian. *Linear Algebra Appl.* 435, 2 (7 2011), 224–242. https://doi.org/10.1016/j.laa.2011.01.030

[2] Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. 2014. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. ACM, 343–352.

[3] Viswanath Gopalakrishnan, Yiqun Hu, and Deepu Rajan. 2010. Random Walks on Graphs for Salient Object Detection in Images. *IEEE Transactions on Image Processing* 19, 12 (2010), 3232–3242. https://doi.org/10.1109/tip.2010.2053940

[4] Kai-Lung Hua, Hong-Cyuan Wang, Chih-Hsiang Yeh, Wen-Huang Cheng, and Yu-Chi Lai. 2016. Background Extraction Using Random Walk Image Fusion. *IEEE Transactions on Cybernetics* 48, 1 (2016), 423–435. https://doi.org/10.1109/tcyb.2016.2640288

[5] William B Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.* 26 (1984), 189–206.

[6] John G Kemeny and James Laurie Snell. 1976. *Finite Markov Chains*. Springer, New York.

[7] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International World Wide Web Conference*. ACM, 1343–1350.

[8] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[9] Huan Li, Richard Peng, Liren Shan, Yuhao Yi, and Zhongzhi Zhang. 2019. Current Flow Group Closeness Centrality for Complex Networks? *The World Wide Web Conference on - WWW '19* (2019), 961–971. https://doi.org/10.1145/3308558.3313490 arXiv:1802.02556

[10] Gu-Li Lin, Hong Peng, Qian-Li Ma, Jia Wei, and Jiang-Wei Qin. 2010. Improving diversity in Web search results re-ranking using absorbing random walks. *2010 International Conference on Machine Learning and Cybernetics* 5 (2010), 2416–2421. https://doi.org/10.1109/icmlc.2010.5580733

[11] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 14, 1 (1978), 265–294.

[12] Daniel A Spielman and Shang-Hua Teng. 2014. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Anal. Appl.* 35, 3 (2014), 835–885.

[13] Changhu Wang, Feng Jing, Lei Zhang, and Hong-Jiang Zhang. 2006. Image annotation refinement using random walk with restarts. *Proceedings of the 14th annual ACM international conference on Multimedia - MULTIMEDIA '06* (2006), 647–650. https://doi.org/10.1145/1180639.1180774

[14] Chuan Xu, Guofeng Zhao, Gaogang Xie, and Shui Yu. 2014. Detection on application layer DDoS using random walk model. *2014 IEEE International Conference on Communications (ICC)* (2014), 707–712. https://doi.org/10.1109/icc.2014.6883402

[15] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. 2010. A Unified Framework for Link Recommendation Using Random Walks. *2010 International Conference on Advances in Social Networks Analysis and Mining* (2010), 152–159.

https://doi.org/10.1109/asonam.2010.27

[16] Yingpei Zeng, Jiannong Cao, Shigeng Zhang, Shanqing Guo, and Li Xie. 2010. Random-walk based approach to detect clone attacks in wireless sensor networks. *IEEE Journal on Selected Areas in Communications* 28, 5 (2010), 677–691. https://doi.org/10.1109/jsac.2010.100606

[17] Zuobai Zhang, Wanyue Xu, and Zhongzhi Zhang. 2020. Nearly Linear Time Algorithm for Mean Hitting Times of Random Walks on a Graph. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020), 726–734. https://doi.org/10.1145/3336191.3371777