Absorbing Time of Random Walks as a Node Group Centrality

Haisong Xia
Fudan University
Shanghai 200433, China
hsxia22@m.fudan.edu.cn

Xiaotian Zhou
Fudan University
Shanghai 200433, China
20210240043@fudan.edu.cn

Zhuoqing Song
Fudan University
Shanghai 200433, China
zqsong19@fudan.edu.cn

Zhongzhi Zhang*
Fudan University
Shanghai 200433, China
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 among Kemeny constant, absorbing random-walk centrality and random detour time, then 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 sum 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 $H(S^*)$. 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 $1 - \frac{k}{k-1} \cdot \frac{1}{e}$ approximate factor and cubic running time. To further accelerate the computation of MANC, we give a faster algorithm with a $1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon$ 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

As a useful method or model, random walk on graphs is widely applicated in many fields: from link prediction [?] and improving search results [?], which are closely related to complex networks, to object detection [?], image background extraction [?] and image annotation refinement [?] in the field of image processing, to DDoS attack detection [?] and clone attack detection [?] in the field of information security. A fundamental quantity relevant to random walks is hitting time. The hitting time $H_{u,v}$ is the expected number of time steps for a walker starting from node u to first reach node v. Hitting time can be used to define other interesting quantities like Kemeny constant, absorbing random-walk centrality and random detour time [?].

In this paper, we first establish a connection among Kemeny constant, absorbing random-walk centrality and random detour time, which prompts us to further extend it to the case of multiple nodes. Inspired by the newly discovered connection, we propose a centrality relevant to node groups: Multiple Absorbing Node Centrality (MANC) through an absorbing random walk model on graphs. For a connected undirected graph, the MANC $H(S)$ of a node set S is defined as the weighted sum of the hitting times of absorbing random walks from all nodes in the graph to S.

Due to the definition of MANC, this centrality can be directly applied to the search engine ranking systems. If we denote web pages that satisfy the search keywords as nodes, then 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 $H(S^*)$.

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 [? ? ?] and Johnson-Lindenstrauss Lemma [?]. For a 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 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 special functions, graphs and random walks.

[*] 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.

2.1 Notations

We use normal regular lowercase letters like a, b, c to denote scalars in $\mathbb{R}$, use bold lowercase letters like $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ to denote vectors, and use bold uppercase letters like $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ to denote matrices.

For the convenience of representing specific element in vectors and matrices, we use $\boldsymbol{a}_i$ to denote the i^{th} element of vector $\boldsymbol{a}$ and use $\boldsymbol{A}_{[i,j]}$ 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, we write sets in subscripts to denote subvectors and submatrices. For example, $\boldsymbol{a}_{-S}$ represents the subvector of $\boldsymbol{a}$ obtained by removing elements with indices in set S, $\boldsymbol{A}_{-S}$ represents the submatrix of $\boldsymbol{A}$ obtained by removing elements with row indices or column indices in set S. Note that the subscript takes precedence over the superscript, thus $\boldsymbol{A}_{-S}^{-1}$ 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 being 1. Sometimes we omit subscripts where there is no ambiguity.

For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, its Frobenius form is $\|\boldsymbol{A}\|_F = \sqrt{\mathrm{Tr}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}$.

Since we prove the approximation guarantee of our algorithms in Section **??**, 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 $(1-\epsilon)\tilde{x} \leq x \leq (1+\epsilon)\tilde{x}$ holds, which we denote as $x \approx_\epsilon \tilde{x}$ for the convenience of writing.*

2.2 Supermodular Functions

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

Definition 2.2 (Monotonicity). *The set function $f : 2^V \to \mathbb{R}^+$ is monotone if and only if for any nonempty set X, Y that satisfies $X \subseteq Y \subseteq V$, the inequality $f(X) \geq f(Y)$ holds.*

Definition 2.3 (Supermodularity). *The set function $f : 2^V \to \mathbb{R}^+$ is supermodular if and only if for any nonempty set X, Y that satisfies $\forall X \subseteq Y \subseteq V, u \in V \setminus Y$, the inequality $f(X) - f(X + u) \geq f(Y) - f(Y + u)$ holds.*

2.3 Graphs and Laplacian Matrices

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 $w : E \to \mathbb{R}^+$ denotes the edge weight function. We use $e = (u, v)$ 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 vector can be defined as $\boldsymbol{d} = \boldsymbol{A}\mathbf{1} = \begin{pmatrix} d_1 & d_2 & \cdots & d_n \end{pmatrix}^\top$, where d_i represents degree of node i. Then the maximum degree can be denoted as $d_{\max} = \max\{d_u | u \in V\}$. If we denote the relevant degree diagonal matrix as $\boldsymbol{D} = \mathrm{diag}\left(d_1, d_2, \cdots, d_n\right)$, 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 $e = (i, j) \in E$, 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 diagonal matrix whose diagonal 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 $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$ and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ respectively, then $\boldsymbol{L}$ can be rewritten as $\boldsymbol{L} = \sum_{i=1}^n \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$. Since $\boldsymbol{L}$ 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 [?] and Kirchhoff index.

Moreover, Laplacian matrix has some useful properties. It is easy to verify that Laplacian matrix is Symmetric Diagonally Dominant(SDD). Also, for a connected undirected graph $\mathcal{G} = (V, E)$ and any nonempty node set S, its corresponding Laplacian submatrix $\boldsymbol{L}_{-S}$ is Symmetric Diagonally Dominant M-matrix(SDDM). For $\boldsymbol{L}_{-S}$, we also have the following lemma.

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

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

2.4 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 $\boldsymbol{P}_{[i,j]} = d_i^{-1} \boldsymbol{A}_{[i,j]}$. 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 [?]

$$\boldsymbol{\pi} = \begin{pmatrix} \pi_1 & \pi_2 & \ldots & \pi_n \end{pmatrix}^\top = \begin{pmatrix} \frac{d_1}{d_\Sigma} & \frac{d_2}{d_\Sigma} & \cdots & \frac{d_n}{d_\Sigma} \end{pmatrix}^\top,$$

where $d_\Sigma = \sum_{i=1}^n d_i$.

Hitting time is a fundamental quantity in random walks, which is also known as absorbing length. The hitting time $H_{u,v}$ 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.

In other words, if we denote the time steps for a walker starting from u to first reach v as the random variable $T_{u,v}$, then we have $H_{u,v} = \mathbb{E}\left[T_{u,v}\right]$. Many interesting quantities can be further obtained from hitting time, including Kemeny constant K, absorbing random-walk centrality H_u and random detour time $D_{i,j}(u)$.

For a connected undirected graph $\mathcal{G} = (V, E)$, its Kemeny constant K is defined as the expected hitting time for a random walker who starts from an arbitary node u to node v which is selected according to the stationary distribution $\boldsymbol{\pi}$. That is, $K = \sum_{v \in V} \boldsymbol{\pi}_v H_{u,v}$.

For node u in a connected undirected 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 to node u, where node i is selected according to the stationary distribution $\boldsymbol{\pi}$. In this case, $H_u = \sum_{i \in V} \boldsymbol{\pi}_i H_{i,u}$.

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. That is, $D_{i,j}(u) = H_{i,u} + H_{u,j}$.

3 RELATED WORK

Many researchers have noticed the connection between random detour time and their proposed centralities. Ranjan et al. [?] presented *topological centrality* $\mathscr{C}^*(u)$ as the reciprocal of $\boldsymbol{L}^{\dagger}_{[u,u]}$, and found that $\mathscr{C}^*(u)$ can be represented by random detour time and hitting time, that is, $\mathscr{C}^*(u)^{-1} \propto \sum_{i=1}^{n} \sum_{j=1}^{n} \left(D_{i,j}(u) - H_{i,j}\right)$. Gangemi et al. [?] simply took variants of random detour time as components of *pivotality*, which is the measure of node reachability. We prove that random detour time is also related to Kemeny constant and absorbing random-walk centrality. This connection also holds for the case of multiple nodes.

There exist various random walk-based centralities, such as the *Random Walk Decay Centrality* [?] and the *Group-to-group random walk betweenness centrality* [?], both of them are variants of existing centralities. Besides, Mavroforakis et al. [?] presented *k absorbing random-walk centrality*, which is the most directly related to our work. We give new proofs on the monotonicity and supermodularity of MANC from the perspective of numerical analysis, which is more concise and intuitive. Moreover, we design and implement a nearly linear algorithm for MANC minimization problem, which is proved to have an approximation guarantee.

Admittedly, many existing studies have focused on the notion of selecting k nodes in the network to optimize some quantities related to absorbing random walk, such as absorbing distance [? ? ?] and absorbing probability [? ?]. However, computing these quantities on infinite random walk models requires time-consuming matrix inversions. Most of researchers use finite random walk models to approximate the infinite case, except for Rosenfeld et al. [?], who use LU decomposition to reduce the complexity of computing absorbing probability. Instead of constructing finite random walk models, our work focus on the infinite random walk model, use a nearly linear algorithm to minimize MANC, which has a $1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon$ approximate factor for any $\epsilon \in (0, 1)$.

4 PROBLEM FORMULATION

4.1 Connections among quantities related to random walk

After proposing definitions in Section ??, 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 K, absorbing random-walk centrality H_u and random detour time $D_{i,j}(u)$.

Theorem 4.1. *For an arbitary node u in a connected graph $\mathcal{G} = (V, E)$ with n nodes,*

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

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

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

□

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 the definition of absorbing random-walk centrality and random detour time to multi-node case.

Similarly, for node set S in a connected graph $\mathcal{G} = (V, E)$, the group hitting time $H_{u,S}$ 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. Also, if we denote the time steps for a walker starting from u to first reach an arbitary node of S as random variable $T_{u,S}$, then we have $H_{u,S} = \mathbb{E}\left[T_{u,S}\right]$.

For node set S in graph $\mathcal{G}$, the group random detour time $D_{i,j}(S)$ 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 4.1 (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 $\boldsymbol{P'} \in \mathbb{R}^{n \times |S|}$, then we have*

$$D_{i,j}(S) = H_{i,S} + \sum_{k=1}^{|S|} \boldsymbol{P'}_{[i,k]} H_{k,j}.$$

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

Theorem 4.2. *For an arbitary node subset $S \subseteq V$ in a connected graph $\mathcal{G} = (V, E)$ with n nodes,*

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

Proof. To prove (??), we utilize the fundamental matrix $\boldsymbol{F}^*$ in the non-absorbing random walk model. According to [?], $\boldsymbol{F}^*$ can be represented as $\boldsymbol{F}^* = (\boldsymbol{I} - \boldsymbol{P} + \boldsymbol{1}\boldsymbol{\pi}^\top)^{-1}\boldsymbol{\Pi}^{-1}$, where $\boldsymbol{\Pi}$ is defined as $\operatorname{diag}(\boldsymbol{\pi})$. Subsequently, the hitting time $H_{i,j}$ can be represented by $\boldsymbol{F}^*$ as $H_{i,j} = \boldsymbol{F}^*_{[j,j]} - \boldsymbol{F}^*_{[i,j]}$ [?]. Then the right side of (??) can be rewritten as

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

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

Afterwards, we are able to rewrite Kemeny constant K as

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

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

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

4.2 Definition of MANC and Its Minimization Problem

Definition 4.2 (Multiple Absorbing Node Centrality,MANC). *For node set S in a connected undirected graph $\mathcal{G} = (V, E)$, its MANC $H(S)$ is defined as the expected hitting times of a random walker who starts from node u to an arbitary node in S, where node i is selected according to the stationary distribution $\boldsymbol{\pi}$. In this case,*

$$H(S) = \sum_{u \in V} \boldsymbol{\pi}_u H_{u,S}.$$

It is obvious that when S contains only one node v, MANC $H(S)$ automatically reduces to absorbing random-walk centrality H_v. 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 $k \ll n$, the goal is to find a node set $S^* \subseteq V$ such that MANC $H(S^*)$ is minimized:*

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

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

Fact 4.3. *Given the absorbing node set S and transition matrix $\boldsymbol{P}$, then the fundamental matrix $\boldsymbol{F}$ can be denoted as*

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

According to (??), 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 $H(S)$ can be written as

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

Furthermore, after simplification using (??), we get the formula of $H(S)$:

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

4.3 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 $H(\cdot)$ is monotone and supermodular, which provides us with a theorecical basis for designing greedy algorithms with approximation guarantees.

4.3.1 NP-hard.

To prove the NP-hardness of MANCM, we try to construct a reduction from Vertex Cover to a decision version of MANCM.

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.*

Theorem 4.4. *MANCM is NP-hard.*

Proof. For an arbitiary connected c-regular graph $\mathcal{G} = (V, E, w)$, where the weights of all edges are equal to 1. Let $S \subseteq V$ be a nonempty node set with capacity k, then we attempt to prove that $H(S) \geq (n-k)/n$, the equality holds if and only if S is a vertex cover of $\mathcal{G}$.

We first prove that if S is a vertex cover of $\mathcal{G}$, then $H(S) = (n-k)/n$. When S is a vertex cover of $\mathcal{G}$, because there are no edges between nodes in $V \setminus S$, we can simplify (??) as

$$H(S) = \boldsymbol{\pi}^\top_{-S} \boldsymbol{L}^{-1}_{-S} \boldsymbol{d}_{-S} = \boldsymbol{\pi}^\top_{-S} \boldsymbol{D}^{-1}_{-S} \boldsymbol{d}_{-S} = \boldsymbol{\pi}^\top_{-S} \mathbf{1}.$$

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

We then prove that if S is not a vertex cover of $\mathcal{G}$, then $H(S) > (n-k)/n$. For node $u \in S$ and node $v \in V \setminus S$, it is obvious that $H_{u,S} = 0$, $H_{v,S} \geq 1$. 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. When a walker starting from node u' moves to node v' with probability of at least $\frac{1}{d_{\max}} > \frac{1}{n}$, its walking length is greater than 1. That is,

$$H_{u',S} = H_{v',S} > \left(1 - \frac{1}{n}\right) + \frac{2}{n} = 1 + \frac{1}{n}.$$

Therefore, if we denote $S + u' + v'$ as S', then we are able to get the lower bound of MANC:

$$\begin{aligned} H(S) &= \sum_{v \in V} \boldsymbol{\pi}_v H_{v,S} = \sum_{v \in V \setminus S'} \boldsymbol{\pi}_v H_{v,S} + \boldsymbol{\pi}_{u'} H_{u',S} + \boldsymbol{\pi}_{v'} H_{v',S} \\ &> \boldsymbol{\pi}_{-S'} \mathbf{1} + \boldsymbol{\pi}_{u'} \left(1 + \frac{1}{n}\right) + \boldsymbol{\pi}_{v'} \left(1 + \frac{1}{n}\right) \\ &= \boldsymbol{\pi}_{-S}^{\top} \mathbf{1} + \frac{\boldsymbol{\pi}_{u'} + \boldsymbol{\pi}_{v'}}{n} > \boldsymbol{\pi}_{-S}^{\top} \mathbf{1} = (n-k)/n. \end{aligned}$$

Based on the above proposition, we can easily construct a polynomial reduction from Vertex Cover on c-regular graphs to a decision version of MANCM, which completes our proof of the NP-hardness of MANCM.

□

4.3.2 Monotonicity and Supermodularity.

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

Theorem 4.5 (Monotonicity). *Let S be an arbitary nonempty node set of connected weighted graph $\mathcal{G} = (V, E, w)$, then for node $u \in V \setminus S$,*

$$H(S) \geq H(S+u).$$

Proof. According to Definition **??**, $H(S) = \sum_{v \in V} \boldsymbol{\pi}_v H_{v,S}$. Since S is a subset of $S + u$, when a random walker reaches nodes in S, it must have reached nodes in $S + u$. Therefore $H_{v,S} \geq H_{v,S+u}$ holds for any node $v \in V$, which completes our proof. □

Theorem 4.6 (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 $u \in V \setminus T$,*

$$H(S) - H(S+u) \geq H(T) - H(T+u).$$

Proof. For a subset A of the probability space, we denote χ_A as the indicator function of the event A. Then we rewrite $H(S) - H(S+u)$ by discussing the hitting node of random walker.

$$\begin{aligned} & H(S) - H(S+u) \\ = & \sum_{v \in V} \boldsymbol{\pi}_v \left(\mathbb{E}\left[T_{v,S}\right] - \mathbb{E}\left[\min\left\{T_{v,S}, T_{v,u}\right\}\right]\right) \\ = & \sum_{v \in V} \boldsymbol{\pi}_v \left(\mathbb{E}\left[T_{v,S}\right] - \mathbb{E}\left[T_{v,S} \chi_{\{T_{v,S} \leq T_{v,u}\}}\right] - \mathbb{E}\left[T_{v,u} \chi_{\{T_{v,u} < T_{v,S}\}}\right]\right) \\ = & \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[(T_{v,S} - T_{v,u}) \chi_{\{T_{v,u} < T_{v,S}\}}\right]. \end{aligned}$$

Similarly, we also have

$$H(T) - H(T+u) = \sum_{v \in V} \boldsymbol{\pi}_v \mathbb{E}\left[(T_{v,T} - T_{v,u}) \chi_{\{T_{v,u} < T_{v,T}\}}\right].$$

As is mentioned in proof of Theorem **??**, $T_{v,S} \geq T_{v,T}$ holds for any node $v \in V$. It is also obvious that $\chi_{\{T_{v,u} < T_{v,S}\}} \geq \chi_{\{T_{v,u} < T_{v,T}\}}$. Combining the above two inequalities, we can prove that

$$(T_{v,S} - T_{v,u}) \chi_{\{T_{v,u} < T_{v,S}\}} \geq (T_{v,T} - T_{v,u}) \chi_{\{T_{v,u} < T_{v,T}\}}$$

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

5 SIMPLE GREEDY ALGORITHM

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

$$\Delta(u, S) = H(S) - H(S+u),$$

$$u^* = \underset{u \in V \setminus S}{\arg\max} \{\Delta(u, S)\}.$$

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 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

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

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

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

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

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

However, at the first iteration of greedy algorithm, $S = \varnothing$, then (**??**) is illegal due to the singularity of $\boldsymbol{L}$. Given that $H(\varnothing) = \infty$, we can define $\Delta(u, \varnothing)$ as $-H(S+u) = -H_u$, which is the negative of absorbing random-walk centrality for node u. It is easy to verify that [?]

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

(**??**) and (**??**) illustrate that for any node $u \in V \setminus S$, computing $\Delta(u, S)$ 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 **??**, reducing time complexity from $O(kn^4)$ to $O(kn^3)$.

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 $\boldsymbol{L}$ and $\boldsymbol{d}$
$d \leftarrow \boldsymbol{d}\boldsymbol{1}, \boldsymbol{\pi} \leftarrow d^{-1}\boldsymbol{d}$
$S_1 \leftarrow \left\{\arg\min_{u \in V} \left\{d(\boldsymbol{e}_u - \boldsymbol{\pi})^\top \boldsymbol{L}^\dagger (\boldsymbol{e}_u - \boldsymbol{\pi})\right\}\right\}$
for $i = 2, 3, \dots, k$ do
    foreach $u \in V \setminus S$ do
        $\Delta(u, S) \leftarrow \frac{(\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1} \boldsymbol{d}_{-S})^2}{d(\boldsymbol{e}_u^\top \boldsymbol{L}_{-S}^{-1} \boldsymbol{e}_u)}$
    $u^* \leftarrow \arg\max_{u \in V \setminus S} \{\Delta(u, S)\}$
    $S_i \leftarrow S_{i-1} \cup \{u^*\}$
return $S_k$
```

Afterwards, we prove that Algorithm **??** has a $1 - \frac{k}{k-1} \cdot \frac{1}{e}$ approximate factor.

Theorem 5.1. *The algorithm S_k = 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*

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

where

$$u^* \overset{\text{def}}{=} \underset{u \in V}{\arg\min}\, H\left(\{u\}\right), S^* \overset{\text{def}}{=} \underset{|S| = k}{\arg\min}\, H\left(S\right).$$

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

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

holds for any positive integer i, which indicates that

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

Subsequently, we can further obtain that

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

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

6 FASTER GREEDY ALGORITHM

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

Lemma 6.1 (Johnson-Lindenstrauss Lemma, JL Lemma [?]). *Given fixed vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n \in \mathbb{R}^d$ 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 **??** indicates that if we project n vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n$ into a lower dimensional space spanned by $O(\log n)$ random vectors, the distances between projected vectors will be preserved with high probability.

Lemma 6.2 (SDD Solver [? ? ?]). *There exists an algorithm $\boldsymbol{x}$ = Solve($\boldsymbol{S}, \boldsymbol{b}, \delta$) which takes a SDDM matrix or a Laplacian $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ with m nonzero elements, 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 **??**, $\|\boldsymbol{x}\|_{\boldsymbol{S}}$ 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 Solve is $\tilde{O}(m)$, where $\tilde{O}(\cdot)$ hides the poly($\log n$) factors.

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

6.1 Approximations of (??)

Given that (**??**) can be regarded as the absorbing random-walk centrality, we use an efficient algorithm ApproxH proposed by [?].

Lemma 6.3. *There exists an algorithm called ApproxH 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 $\tilde{H}_u$ of absorbing random-walk centrality H_u for any node u in graph $\mathcal{G}$ such that*

$$H_u \approx_\epsilon \tilde{H}_u.$$

6.2 Approximations of (??)

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

Lemma 6.4. *Given a connected weighted undirected graph $\mathcal{G}$ = (V, E, w), let $\boldsymbol{L}$ denote the Laplacian matrix of the graph. Then for an arbitary node set $S \subseteq V$ and an arbitary vector $\boldsymbol{v} \in \mathbb{R}^{n-|S|}$, $\boldsymbol{v}_i^2 \leq n w_{\min}^{-1} \|\boldsymbol{v}\|_{\boldsymbol{L}_{-S}}^2$ holds.*

Proof. As is mentioned in Section **??**, 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 $\boldsymbol{L}_{-S} = \boldsymbol{L}' + \boldsymbol{Z} = \boldsymbol{B}'^\top \boldsymbol{W}' \boldsymbol{B}' + \boldsymbol{Z}$, where $\boldsymbol{L}'$ denotes the Laplacian matrix of another graph, $\boldsymbol{Z}$ denotes a diagonal matrix.

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

```
Input  : A connected weighted undirected graph
         $\mathcal{G} = (V, E, w)$; an error parameter $\epsilon > 0$
Output : The approximation $\tilde{H}_u$ of absorbing random-walk
         centrality $H_u$ 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 $i = 1, 2, \dots, k$ do
    $\boldsymbol{X}'_{[i,:]} \leftarrow$ Solve($\boldsymbol{L}, \overline{\boldsymbol{X}}_{[i,:]}, \delta$)
foreach $u \in V$ do
    $\tilde{H}_u \leftarrow d\,\|\boldsymbol{X}'(\boldsymbol{e}_u - \boldsymbol{\pi})\|^2$
return $\{\tilde{H}_u | u \in V\}$
```

If $Z_{[i,i]} \neq 0$, then we have $Z_{[i,i]} \geq w_{\min}$, it is apparent that $\boldsymbol{v}_i^2 \leq n w_{\min}^{-1} \|\boldsymbol{v}\|^2_{\boldsymbol{L}_{-S}}$. If $Z_{[i,i]} = 0$, there must exist a node j that satisfies $Z_{[j,j]} \geq w_{\min}$ in the component of graph that contains node i after removing nodes in S. Let $\mathcal{P}_{ij}$ denotes the simple path connecting node i and node j, then we have

$$\|\boldsymbol{v}\|^2_{\boldsymbol{L}_{-S}} \geq w_{\min} \sum_{(a,b)\in\mathcal{P}_{ij}} (\boldsymbol{v}_a - \boldsymbol{v}_b)^2 + w_{\min}\boldsymbol{v}_j^2$$
$$\geq w_{\min} n^{-1} \left(\sum_{(a,b)\in\mathcal{P}_{ij}} (\boldsymbol{v}_a - \boldsymbol{v}_b) + \boldsymbol{v}_j \right)^2 \geq w_{\min} n^{-1} \boldsymbol{v}_i^2,$$

which completes our proof. □

We first approximate the numerator of (**??**), 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 **??** to avoid computing matrix inverse $\boldsymbol{L}_{-S}^{-1}$.

Lemma 6.5. *Given a connected weighted undirected graph $\mathcal{G} = (V, E, w)$,the nonempty node set S , the Laplacian matrix $\boldsymbol{L}$ and an error parameter $\epsilon \in (0, 1)$, let $\boldsymbol{x}' =$ 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 **??** and Lemma **??**, we can bound $(\boldsymbol{x}'_i - \boldsymbol{x}_i)^2$ as

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

where the last inequality is due to Lemma **??**. 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 $\boldsymbol{x}_i \geq 1$. Finally, we are able to give the approximation of $\boldsymbol{x}_i$ as

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

□

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

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

Although (**??**) indicates that we can avoid matrix inversion by using Lemma **??**, 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 **??** to reduce the dimension of the corresponding matrices.

From Lemma **??**, 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 **??**, where $q = r = \lceil 24\epsilon^{-2}\log n \rceil$, then we have

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

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 (**??**) and (**??**) 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 (**??**) and (**??**), we can efficiently approximate the denominator of (**??**).

Lemma 6.6. *Given a connected weighted undirected graph $\mathcal{G} = (V, E, w)$,the nonempty node set S , the Laplacian matrix $\boldsymbol{L}$ 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 $\boldsymbol{X}'_{[i,:]} =$ Solve$(\boldsymbol{L}_{-S}, \overline{\boldsymbol{X}}_{[i,:]}, \delta_2)$, $\boldsymbol{Y}'_{[i,:]} =$ 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

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

where the third inequality and the fourth inequality are due to Lemma **??** and Lemma **??** respectively.

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

$$n\delta_2 \sqrt{w_{\min}^{-1}} \left\| \tilde{X} \right\|_F \leq n\delta_2 \sqrt{w_{\min}^{-1}(1+\epsilon/7) \sum_{i \in V \setminus S} \| \boldsymbol{X} \boldsymbol{e}_i \|^2}$$

$$\leq n\delta_2 \sqrt{w_{\min}^{-1}(1+\epsilon/7) \mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1} \right)} \leq n^2 \delta_2 w_{\min}^{-1} \sqrt{1+\epsilon/7},$$

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

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

On the other hand, according to Lemma **??**,

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

where the last inequality is due to the scaling of $\boldsymbol{e}_i^\top \boldsymbol{L}_{-S}^{-1} \boldsymbol{d}_{-S} \geq 1$.

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 $\frac{1}{n} \sqrt{\frac{1-\epsilon/7}{2 w_{\max}}}$. Without loss of generality, we let $\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\| \geq \frac{1}{n} \sqrt{\frac{1-\epsilon/7}{2 w_{\max}}}$, then we have

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

According to the inequality above, we can further obtain

$$\frac{\left| \left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2 - \| \boldsymbol{X}' \boldsymbol{e}_i \|^2 \right|}{\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2} = \frac{\left| \left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\| - \| \boldsymbol{X}' \boldsymbol{e}_i \| \right| \left(\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\| + \| \boldsymbol{X}' \boldsymbol{e}_i \| \right)}{\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2}$$

$$\leq \frac{\epsilon}{31} \left(2 + \frac{\epsilon}{31} \right) \leq \frac{\epsilon}{15},$$

which means

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

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

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

thus indicating that

$$\frac{\left| \left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\|^2 - \| \boldsymbol{Y}' \boldsymbol{e}_i \|^2 \right|}{\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2 + \left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\|^2} \leq \frac{\left| \left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\| - \| \boldsymbol{Y}' \boldsymbol{e}_i \| \right| \left(\left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\| + \| \boldsymbol{Y}' \boldsymbol{e}_i \| \right)}{\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2}$$
$$\leq \frac{\epsilon}{31} \left(2 + \frac{\epsilon}{31} \right) \leq \frac{\epsilon}{15}. \tag{19}$$

Combining (**??**) and (**??**), we are finally able to get

$$\frac{\left| \left(\| \boldsymbol{X}' \boldsymbol{e}_i \|^2 + \| \boldsymbol{Y}' \boldsymbol{e}_i \|^2 \right) - \left(\left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2 + \left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\|^2 \right) \right|}{\| \boldsymbol{X}' \boldsymbol{e}_i \|^2 + \| \boldsymbol{Y}' \boldsymbol{e}_i \|^2}$$

$$\leq \frac{\left| \left\| \tilde{\boldsymbol{X}} \boldsymbol{e}_i \right\|^2 - \| \boldsymbol{X}' \boldsymbol{e}_i \|^2 \right| + \left| \left\| \tilde{\boldsymbol{Y}} \boldsymbol{e}_i \right\|^2 - \| \boldsymbol{Y}' \boldsymbol{e}_i \|^2 \right|}{\| \boldsymbol{X}' \boldsymbol{e}_i \|^2 + \| \boldsymbol{Y}' \boldsymbol{e}_i \|^2} \leq \frac{\epsilon}{7},$$

which completes our proof in conjunction with Lemma **??**. □

6.3 Nearly Linear Time Approximation Algorithms

Combining Lemmas **??** and **??**, we finally propose the approximate algorithm ApproxDelta of computing $\Delta(u, S)$ when $S \neq \emptyset$. Because ApproxDelta calls algorithm Solve in **??** for $q = \lceil 24(\epsilon/7)^{-2} \log n \rceil$ times, the complexity of ApproxDelta is $\tilde{O}\left(m\epsilon^{-2}\right)$, where $\tilde{O}(\cdot)$ hides the poly($\log n$) factors. Furthermore, we are able to provide its approximation guarantee.

```
Algorithm 3: ApproxDelta($\mathcal{G}, S, \epsilon$)
  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 $\boldsymbol{d}$
  $n \leftarrow |V|, m \leftarrow |E|$
  $w_{\min} \leftarrow \min\{w_e | e \in E\}, w_{\max} \leftarrow \max\{w_e | e \in E\}$
  $d \leftarrow \boldsymbol{d}\boldsymbol{1}$,$q, r \leftarrow \lceil 24(\epsilon/7)^{-2} \log n \rceil$
  $\delta_1 \leftarrow \frac{w_{\min}\epsilon}{7n^2 w_{\max}}$,$\delta_2 \leftarrow \frac{w_{\min}\epsilon}{31n^2} \sqrt{\frac{1-\epsilon/7}{2 w_{\max}(1+\epsilon/7)}}$
  $\boldsymbol{x}' \leftarrow$ Solve($\boldsymbol{L}_{-S}, \boldsymbol{d}_{-S}, \delta_1$)
  $\boldsymbol{Q} \leftarrow$ GenerateRandomMatrix($q, m$)
  $\boldsymbol{R} \leftarrow$ 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
      $\boldsymbol{X}'_{[i,:]} \leftarrow$ Solve($\boldsymbol{L}_{-S}, \overline{\boldsymbol{X}}_{[i,:]}, \delta_2$)
      $\boldsymbol{Y}'_{[i,:]} \leftarrow$ 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 $\{\Delta'(u, S) | u \in V \setminus S\}$
```

Lemma 6.7. *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 **??** and Algorithm **??**, we can give the faster greedy algorithm Approx with a $1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon$ approximate factor. Since Approx calls ApproxH once and calls ApproxDelta $k - 1$ times, the complexity of Approx is $\tilde{O}\left(km\epsilon^{-2}\right)$, which is nearly linear.

Theorem 6.8. *The algorithm S_k = 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*

$$(1+\epsilon) H\left(\{u^*\}\right) - H\left(S_k\right) \geq$$
$$\left(1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon\right) \left((1+\epsilon) H\left(\{u^*\}\right) - H\left(S^*\right)\right),$$

```
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 $|S_k| = k$
$d \leftarrow \boldsymbol{d}\mathbf{1}, \boldsymbol{\pi} \leftarrow d^{-1}\boldsymbol{d}$
$\{\tilde{H}_u | u \in V\} \leftarrow$ APPROXH($\mathcal{G}, \epsilon$)
$S_1 \leftarrow \{\arg\min_{u \in V} \{\tilde{H}_u\}\}$
for $i = 2, 3, \ldots, k$ do
    $\{\Delta'(u, S) | u \in V \setminus S\} \leftarrow$ APPROXDELTA($\mathcal{G}, S, \epsilon$)
    $u^* \leftarrow \arg\max_{u \in V \setminus S} \{\Delta'(u, S)\}$
    $S_i \leftarrow S_{i-1} \cup \{u^*\}$
return $S_k$
```

where

$$u^* \overset{\text{def}}{=} \underset{u \in V}{\arg\min}\, H(\{u\}), S^* \overset{\text{def}}{=} \underset{|S|=k}{\arg\min}\, H(S).$$

Proof. Since Algorithm **??** use $\Delta'(u, S)$ computed by Algorithm **??** instead of $\Delta(u, S)$, combining Lemma **??** and supermodularity, we are able to obtain that

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

holds for any positive integer i, which indicates that

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

Subsequently, we can further obtain that

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

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

7 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 [?] and SNAP [?]. The information of these datasets is shown in Table **??**. 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 256GiB RAM and 3.5GHz AMD EPYC Milan CPU, using 32 threads.

Table 1: Information of datasets as well as running time of two algorithms on datasets, where n, m denote the number of nodes and edges of a network's largest connected component respectively.

Network	n	m	Time (seconds)	
			Exact	Approx
Euroroads	1039	1305	0.95	0.89
Hamsterster friends	1788	12476	2.91	0.97
ego-Facebook	4039	88234	38.37	4.63
CA-GrQc	4158	13428	39.87	1.28
US power grid	4941	6594	73.03	1.13
Reactome	5973	146992	125.02	8.30
CA-HepTh	8638	24827	356.98	2.72
Sister cities	10320	17988	605.11	5.19
CA-HepPh	11204	117649	-	9.79
CAIDA	26475	53381	-	13.80
loc-Gowalla	196591	950327	-	341.21
com-Amazon	334863	925872	-	1026.41
Dogster friends	426485	8543321	-	2400.34
roadNet-PA	1087562	1541514	-	4647.36
YouTube	1134890	2987624	-	2919.23
roadNet-TX	1351137	1879201	-	6004.25
Skitter	1694616	11094209	-	7191.77
roadNet-CA	1957027	2760388	-	10391.41
Flixster	2523386	7918801	-	5674.25

7.1 Performance of Exact and Approx

We first compare the performance of Exact and Approx with optimum solution on four small networks [?]: *Zebra* with 23 nodes, *Zachary karate club* with 34 nodes, *Contiguous USA* with 49 nodes and *Les Misérables* with 77 nodes. 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 **??**. Figure **??** 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.

We subsequently compare the performance of Exact and Approx with three other algorithms: Top-Absorb, Top-Degree and Top-PageRank. Top-Absorb simply chooses k absorbing nodes with minimum absorbing random-walk centrality, while Top-Degree and Top-PageRank chooses them with biggest degrees and biggest PageRank value respectively. We run these four algorithms on four medium-sized networks, the results is shown in Figure **??**. Figure **??** demonstrates that both of our algorithms also get similar approximate solutions in larger networks, which outperform the solutions of other algorithms.

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

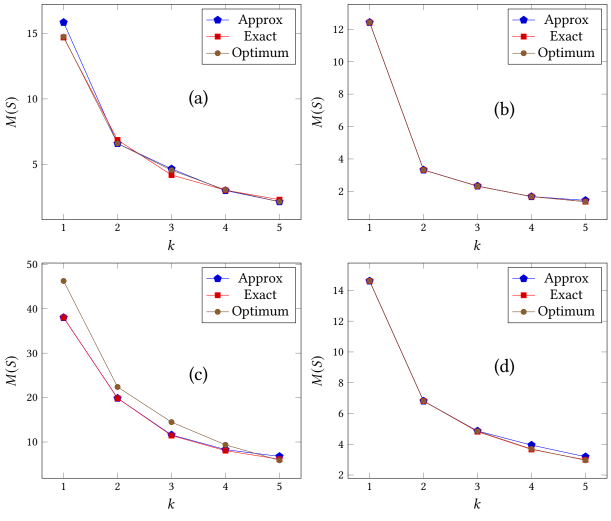

Figure 1: MANC $H(S)$ of node set S computed by three different algorithms(Exact,Approx and Optimum) on four networks: Zebra (a), Zachary karate club (b), Contiguous USA (c) and Les Misérables (d).

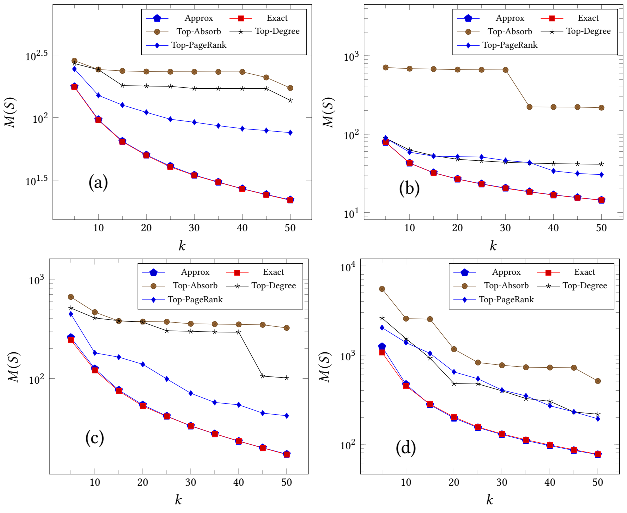

Figure 2: MANC $H(S)$ of node set S computed by four different algorithms(Exact,Approx,Top-Absorb and Top-Degree) on four networks: CA-GrQc (a), ego-Facebook (b), Euroroads (c) and US power grid (d).

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

8 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 $H(S)$ of the node set S is defined as the weighted sum 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 $H(S^*)$. 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.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.