Current Flow Group Closeness Centrality for Complex Networks[*]

Huan Li
Fudan University
Shanghai, China
huanli16@fudan.edu.cn

Richard Peng
Georgia Institute of Technology
Atlanta, Georgia, United States
rpeng@cc.gatech.edu

Liren Shan
Fudan University
Shanghai, China
13307130150@fudan.edu.cn

Yuhao Yi
Fudan University
Shanghai, China
yhyi15@fudan.edu.cn

Zhongzhi Zhang
Fudan University
Shanghai, China
zhangzz@fudan.edu.cn

ABSTRACT

The problem of selecting a group of vertices under certain constraints that maximize their joint centrality arises in many practical scenarios. In this paper, we extend the notion of current flow closeness centrality (CFCC) to a set of vertices in a graph, and investigate the problem of selecting a subset S to maximizes its CFCC $C(S)$, with the cardinality constraint $|S| = k$. We show the NP-hardness of the problem, but propose two greedy algorithms to minimize the reciprocal of $C(S)$. We prove the approximation ratios by showing the monotonicity and supermodularity. A proposed deterministic greedy algorithm has an approximation factor $(1 - \frac{k}{k-1} \cdot \frac{1}{e})$ and cubic running time. To compare with, a proposed randomized algorithm gives $(1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon)$-approximation in nearly-linear time, for any $\epsilon > 0$. Extensive experiments on model and real networks demonstrate the effectiveness and efficiency of the proposed algorithms, with the randomized algorithm being applied to massive networks with more than a million vertices.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; **Discrete optimization**; • **Information systems → Data mining**.

KEYWORDS

Social Networks, Centrality, Combinatorial Optimization, Spectral Graph Theory

ACM Reference Format:
Huan Li, Richard Peng, Liren Shan, Yuhao Yi, and Zhongzhi Zhang. 2019. Current Flow Group Closeness Centrality for Complex Networks. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313490

[*]Huan Li, Liren Shan, Yuhao Yi, and Zhongzhi Zhang are with Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433.

1 INTRODUCTION

A fundamental problem in network science and graph mining is to identify crucial vertices [35, 38]. It is an important tool in network analysis and found numerous applications in various areas [44]. The first step of finding central vertices is to define suitable indices measuring relative importance of vertices. Over the past decades, many centrality measures were introduced to characterize and analyze the roles of vertices in networks [5, 8, 9, 53]. Among them, a popular one is closeness centrality [3, 4]: the closeness of a vertex is the reciprocal of the sum of shortest path distances between it and all other vertices. However, this metric considers only the shortest paths, and more importantly, neglects contributions from other paths. Therefore it can produce some odd effects, or even counterintuitive results [7]. To avoid this shortcoming, Brandes and Fleischer presented current flow closeness centrality [11] based on electrical networks [19] , which takes into account contributions from all paths between vertices. Current flow based closeness has been shown to better discriminate vertices than its traditional counterparts [7].

While most previous works focus on measures and algorithms for the importance of individual vertices in networks [38], the problem of determining a group of k most important vertices arises frequently in data mining and graph applications. For example, in social networks, retailers may want to choose k vertices as promoters of product, such that the number of the potentially influenced customers is maximized [31]. Another example is P2P networks, where one wants to place resources on a fixed number of k peers so they are easily accessed by others [25]. In order to measure the importance of a group of vertices, Everett and Borgatti [21] extended the idea of individual centrality to group centrality, and introduced the concepts of group centrality, for example, group closeness. Recently, some algorithms have been developed to compute or estimate group closeness [6, 13, 56]. However, similar to the case of individual vertices, these notions of group centrality also disregard contributions from paths that are not shortest.

In this paper, we extend current flow closeness of individual vertices [7] by proposing current flow closeness centrality (CFCC) for group of vertices. In a graph with n vertices and m edges, the CFCC $C(S)$ of a vertex group $S \subset V$ is equal to the ratio of n to the sum of effective resistances between S and all vertices u in $V \setminus S$. We then consider the optimization problem: how can we find a group

S^* of k vertices so as to maximize $C(S^*)$. We solve this problem by considering an equivalent problem of minimizing the reciprocal of $C(S^*)$. We show that the problem is NP-hard in Section 4, but also prove that the problem is an instance of supermodular set function optimization with cardinality constraint in Section 5. The latter allows us to devise greedy algorithms to solve this problem, leading to two greedy algorithms with provable approximation guarantees:

(1) A deterministic algorithm with a $(1 - \frac{k}{k-1} \cdot \frac{1}{e})$ approximation factor and $O(n^3)$ running time (Section 6);
(2) A randomized algorithm with a $(1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon)$-approximation factor and $\widetilde{O}(km\epsilon^{-2})$ running time[1] for any small $\epsilon > 0$ (Section 7).

A key ingredient of our second algorithm is nearly linear time solvers for Laplacians and symmetric, diagonally dominant, M-matrices (SDDM) [16, 48], which has been used in various optimization problems on graphs [17, 30, 41].

We perform extensive experiments on some networks to evaluate our algorithm, and some of their results are in Section 8. These results show that both algorithms are effective. Moreover, the second algorithm is efficient and is scalable to large networks with more than a million vertices.

1.1 Related Works

There exist various measures for centrality of a group of vertices, based on graph structure or dynamic processes, such as betweenness [18, 22, 39, 55], absorbing random-walk centrality [37, 40, 57], and grounding centrality [14, 46]. Since the criterion for importance of a vertex group is application dependent [24], many previous works focus on selecting (or deleting) a group of k vertices (for some given k) in order to optimize related quantities. These quantities are often measures of vertex group importance motivated by the applications, including minimizing the leading eigenvalue of adjacency matrix for vertex immunization [12, 51], minimizing the mean steady-state variance for first-order leader-follower noisy consensus dynamics [15, 45], maximizing average distance for identifying structural hole spanners [47, 54], and others.

Previous works on closeness centrality and related algorithms are most directly related to our focus on the group closeness centrality in this paper. The closeness centrality for an individual vertex was proposed [3] and formalized [4] by Bavelas. For a given vertex, its closeness centrality is defined as the reciprocal of the sum of shortest path distances of the vertex to all the other vertices. Everett and Borgatti [21] extended the individual closeness centrality to group closeness centrality, which measures how close a vertex group is to all other vertices. For a graph with n vertices and m edges, exactly computing the closeness centrality of a group of vertices involves calculating all-pairwise shortest path length, the time complexity of the state-of-the-art algorithm [28] for which is $O(nm + n^2 \log n)$. To reduce the computation complexity, various approximation algorithms were developed. A greedy $O(n^3)$ algorithm with approximation ratio $\left(1 - \frac{k}{k-1} \cdot \frac{1}{e}\right)$ was devised [13], and a sampling algorithm that scales better to large networks, but without approximation guarantee was also proposed in the same paper. Very recently, new techniques [6] have been developed to speed up the greedy algorithm in [13] while preserving its theoretical guarantees.

Conventional closeness centrality is based on the shortest paths, omitting the contributions from other paths. In order to overcome this drawback, Brandes and Fleischer introduced current flow closeness centrality for an individual vertex [11], which essentially considers all paths between vertices, but still gives large weight to short paths. Our investigation can be viewed as combining this line of current based centrality measures with the study of selecting groups of k vertices.

2 PRELIMINARIES

In this section, we briefly introduce some useful notations and tools for the convenience of description of our problem and algorithms.

2.1 Notations

We use normal lowercase letters like a, b, c to denote scalars in $\mathbb{R}$, normal uppercase letters like A, B, C to denote sets, bold lowercase letters like $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ to denote vectors, and bold uppercase letters like $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ to denote matrices. We write $\boldsymbol{a}_{[i]}$ 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 write $\boldsymbol{A}_{[i,:]}$ to denote the i^{th} row of $\boldsymbol{A}$ and $\boldsymbol{A}_{[:,j]}$ to denote the j^{th} column of $\boldsymbol{A}$.

We write sets in matrix subscripts to denote submatrices. For example, $\boldsymbol{A}_{[I,J]}$ denotes the submatrix of $\boldsymbol{A}$ with row indices in I and column indices in J. To simplify notation, we also write $\boldsymbol{A}_{-i}$ to denote the submatrix of $\boldsymbol{A}$ obtained by removing the i^{th} row and i^{th} column of $\boldsymbol{A}$. For example, for an $n \times n$ matrix $\boldsymbol{A}$, $\boldsymbol{A}_{-n}$ denotes the submatrix $\boldsymbol{A}_{[1:n-1,1:n-1]}$.

Note that the precedence of matrix subscripts is the lowest. Thus, $\boldsymbol{A}_{-n}^{-1}$ denotes the inverse of $\boldsymbol{A}_{-n}$ instead of a submatrix of $\boldsymbol{A}^{-1}$.

For two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we write $\boldsymbol{A} \preceq \boldsymbol{B}$ to denote that $\boldsymbol{B} - \boldsymbol{A}$ is positive semidefinite, i.e., $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \leq \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x}$ holds for every real vector $\boldsymbol{x}$.

We use $\boldsymbol{e}_i$ to denote the i^{th} standard basis vector of appropriate dimension, and $\boldsymbol{1}_S$ to denote the indicator vector of S.

2.2 Graphs, Laplacians, and Effective Resistances

We write $G = (V, E, w)$ to denote a positively weighted undirected graph with n vertices, m edges, and edge weight function $w : E \to \mathbb{R}^+$. The Laplacian matrix $\boldsymbol{L}$ of G is defined as $\boldsymbol{L}_{[u,v]} = -w(u, v)$ if $u \sim v$, $\boldsymbol{L}_{[u,v]} = \deg(u)$ if $u = v$, and $\boldsymbol{L}_{[u,v]} = 0$ otherwise, where $\deg(u) \overset{\text{def}}{=} \sum_{u \sim v} w(u, v)$ is the weighted degree of u and $u \sim v$ means $(u, v) \in E$. Let $w_{\max}$ and $w_{\min}$ denote, respectively, the maximum weight and minimum weight among all edges. If we orient each edge of G arbitrarily, we can also write it's Laplacian as $\boldsymbol{L} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B}$, where $\boldsymbol{B}_{m \times n}$ is the signed edge-vertex incidence matrix defined by $\boldsymbol{B}_{[e,u]} = 1$ if u is e's head, $\boldsymbol{B}_{[e,u]} = -1$ if u is e's tail, and $\boldsymbol{B}_{[e,u]} = 0$ otherwise, and $\boldsymbol{W}_{m \times m}$ is a diagonal matrix with $\boldsymbol{W}_{[e,e]} = w(e)$. It is not hard to show that quadratic forms of $\boldsymbol{L}$ can be written as $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \sum_{u \sim v} w(u, v) \left(\boldsymbol{x}_{[u]} - \boldsymbol{x}_{[v]}\right)^2$, which immediately implies that $\boldsymbol{L}$ is positive semidefinite, and $\boldsymbol{L}$ only has one zero eigenvalue if G is a connected graph.

[1]We use the $\widetilde{O}(\cdot)$ notation to hide the poly log factors.

The following fact shows that submatrices of Laplacians are always positive definite and inverse-positive.

Fact 2.1. *Let $\boldsymbol{L}$ be the Laplacian of a connected graph and let $\boldsymbol{X}$ be a nonnegative, diagonal matrix with at least one nonzero entry. Then, $\boldsymbol{L} + \boldsymbol{X}$ is positive definite, and every entry of $(\boldsymbol{L} + \boldsymbol{X})^{-1}$ is positive.*

Let $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$ be eigenvalues of $\boldsymbol{L}$ of a connected graph G, and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ be the corresponding orthonormal eigenvectors. Then we can decompose $\boldsymbol{L}$ as $\boldsymbol{L} = \sum_{i=2}^{n} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T$ and define its pseudoinverse as $\boldsymbol{L}^\dagger = \sum_{i=2}^{n} \frac{1}{\lambda_i} \boldsymbol{v}_i \boldsymbol{v}_i^T$.

It is not hard to verify that if $\boldsymbol{L}$ and $\boldsymbol{H}$ are Laplacians of connected graphs supported on the same vertex set, then $\boldsymbol{L} \preceq \boldsymbol{H}$ implies $\boldsymbol{H}^\dagger \preceq \boldsymbol{L}^\dagger$.

The pseudoinverse of Laplacian matrix can be used to define effective resistance between any pair of vertices [32].

Definition 2.2. For a connected graph $G = (V, E, w)$ with Laplacian matrix $\boldsymbol{L}$, the effective resistance between vertices u and v is defined as $R_{\text{eff}}(u, v) = (\boldsymbol{e}_u - \boldsymbol{e}_v)^T \boldsymbol{L}^\dagger (\boldsymbol{e}_u - \boldsymbol{e}_v)$.

The effective resistance between two vertices can also be expressed in term of the diagonal elements of the inverse for submatrices of $\boldsymbol{L}$.

Fact 2.3 ([27]). $R_{\text{eff}}(u, v) = \left(\boldsymbol{L}_{-u}^{-1}\right)_{[v,v]} = \left(\boldsymbol{L}_{-v}^{-1}\right)_{[u,u]}$.

2.3 Current Flow Closeness Centrality

The current flow closeness centrality was proposed in [11]. It is based on the assumption that information spreads efficiently like an electrical current.

To define current flow closeness, we treat the graph G as a resistor network via replacing every edge e by a resistor with resistance $r_e = 1/w(e)$. Let $v_{st}(u)$ denote the voltage of u when a unit current enters the network at s and leaves it at t.

Definition 2.4. The current flow closeness $C(u)$ of a vertex u is defined as $C(u) = n / \sum_{v \in V} (v_{uv}(u) - v_{uv}(v))$.

It has been proved [11] that the current flow closeness of vertex u equals the ratio of n to the sum of effective resistances between u and other vertices.

Fact 2.5. $C(u) = n / \sum_{v \in V} R_{\text{eff}}(u, v)$.

Actually, current flow closeness centrality is equivalent to information centrality [50].

2.4 Supermodular Functions

We now give the definitions for monotone and supermodular set functions. For simplicity, we write $S + u$ to denote $S \cup \{u\}$ and $S - u$ to denote $S \setminus \{u\}$.

Definition 2.6 (Monotonicity)**.** A set function $f : 2^V \to \mathbb{R}$ is monotone if $f(S) \geq f(T)$ holds for all $S \subseteq T$.

Definition 2.7 (Supermodularity)**.** A set function $f : 2^V \to \mathbb{R}$ is supermodular if $f(S) - f(S + u) \geq f(T) - f(T + u)$ holds for all $S \subseteq T \subseteq V$ and $u \in V$.

3 CURRENT FLOW CLOSENESS OF A GROUP OF VERTICES

We follow the idea of [11] to define current flow closeness centrality (CFCC) of a group of vertices.

To define current flow closeness centrality for a vertex set $S \subseteq V$, we treat the graph G as a resistor network in which all vertices in S are grounded. Thus, vertices in S always have voltage 0. For a vertex $u \notin S$, let $v_{uS}(v)$ be the voltage of v when a unit current enters the network at u and leaves it at S (i.e. the ground). Then, we define the current flow closeness of S as follows.

Definition 3.1. Let $G = (V, E, w)$ be a connected weighted graph. The current flow closeness centrality $C(S)$ of a vertex group $S \subseteq V$ is defined as $C(S) = n / \sum_{u \in V} (v_{uS}(u) - v_{uS}(S)) = n / \sum_{u \in V} v_{uS}(u)$.

Note that there are different variants of the definition of CFCC for a vertex group. For example, we can use$(n - |S|) / (\sum_{u \in V} v_{uS}(u))$ as the measure of CFCC for a vertex set S. Definition 3.1 adopts the standard form as the classic closeness centrality [13].

We next show that $C(S)$ is in fact equal to the ratio of n to a sum of effective resistances as in Fact 2.5.

Let $u \in V$ be a fixed vertex. Suppose there is a unit current enters the network at u and leaves it at S. Let $\boldsymbol{v} \in \mathbb{R}^n$ be a vector of voltages at vertices. By Kirchhoff's Current Law and Ohm's Law, we have $\boldsymbol{L}\boldsymbol{v} = \boldsymbol{e}_u - \boldsymbol{i}_S$, where $\boldsymbol{i}_S$ denotes the amount of current flowing out of S. Since vertices in S all have voltage 0, we can restrict this equation to vertices in $V - S$ as $\boldsymbol{L}_{[V-S, V-S]} \boldsymbol{v}_{[V-S]} = \boldsymbol{e}_u$, which leads to $\boldsymbol{v}_{[V-S]} = \boldsymbol{L}_{[V-S, V-S]}^{-1} \boldsymbol{e}_u$. This gives the expression of voltage at u as $v_{uS}(u) = \boldsymbol{e}_u^T \boldsymbol{v} = \boldsymbol{e}_u^T \boldsymbol{L}_{-S}^{-1} \boldsymbol{e}_u$. Now we can write the CFCC of S as

$$C(S) = \frac{n}{\sum_{u \in V} v_{uS}(u)} = \frac{n}{\sum_{u \in V - S} \boldsymbol{e}_u^T \boldsymbol{L}_{-S}^{-1} \boldsymbol{e}_u} = \frac{n}{\operatorname{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)}.$$

Note that the diagonal entry $\left(\boldsymbol{L}_{-S}^{-1}\right)_{[u,u]}$ of $\left(\boldsymbol{L}_{-S}^{-1}\right)$ is exactly the effective resistance $R_{\text{eff}}(u, S)$ between vertex u and vertex set S [15], with $R_{\text{eff}}(u, S) = 0$ for any $u \in S$. Then we have the following relation governing $C(S)$ and $R_{\text{eff}}(u, S)$.

Fact 3.2. $C(S) = n / \operatorname{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right) = n / \sum_{u \in V} R_{\text{eff}}(u, S)$.

Being able to define CFCC of a vertex set raises the problem of maximizing current flow closeness subject to a cardinality constraint, which we state below.

Problem 1 (Current Flow Closeness Maximization, CFCM). *Given a connected graph $G = (V, E, w)$ with n vertices, m edges, and edge weight function $w : E \to \mathbb{R}^+$ and an integer $1 \leq k \leq n$, find a vertex group $S^* \in V$ such that the CFCC $C(S^*)$ is maximized, that is $S^* \in \underset{S \subseteq V, |S| = k}{\arg\max} \; C(S)$.*

4 HARDNESS OF CURRENT FLOW CLOSENESS MAXIMIZATION

In this section, we prove that Problem 1 is NP-hard. We will give a reduction from vertex cover on 3-regular graphs (graphs whose vertices all have degree 3), which is an NP-complete problem [23]. The decision version of this problem is stated below.

Problem 2 (Vertex Cover on 3-regular graphs, VC3). *Given a connected 3-regular graph $G = (V, E)$ and an integer k, decide whether or not there is a vertex set $S \subset V$ such that $|S| \leq k$ and S is a vertex cover of G (i.e. every edge in E is incident with at least one vertex in S).*

We then give the decision version of Problem 1.

Problem 3 (Current Flow Closeness Maximization, Decision Version, CFCMD). *Given a connected graph $G = (V, E, w)$, an integer k, and a real number $r \in \mathbb{R}$, decide whether or not there is a vertex set $S \subset V$ such that $|S| \leq k$ and $C(S) \geq r$.*

To give the reduction, we will need the following lemma.

Lemma 4.1. *Let $G = (V, E, w)$ be a connected 3-regular graph with all edge weights being 1 (i.e. $w(e) = 1$ for all $e \in E$). Let $S \subset V$ be a nonempty vertex set, and $k = |S|$. Then, $C(S) \leq 3n/(n-k)$ and the equality holds if and only if S is a vertex cover of G.*

Proof. We first show that if S is a vertex cover of G then $C(S) = 3n/(n-k)$. When S is a vertex cover, $V \setminus S$ is an independent set. Thus, $\boldsymbol{L}_{-S}$ is a diagonal matrix with all diagonal entries being 3. So we have

$$C(S) = n/\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right) = n/\mathrm{Tr}\left((\mathrm{diag}(3, \ldots, 3))^{-1}\right) = 3n/(n-k).$$

We then show that if S is not a vertex cover of G then $C(S) < 3n/(n-k)$. When S is not a vertex cover, $V \setminus S$ is not an independent set. Thus, $\boldsymbol{L}_{-S}$ is a block diagonal matrix, with each block corresponding to a connected component of $G[V \setminus S]$, the induced graph of G on $V \setminus S$. Let $T \subseteq V \setminus S$ be a connected component of $G[V \setminus S]$ such that $|T| > 1$. Then, the block of $\boldsymbol{L}_{-S}$ corresponding to T is $\boldsymbol{L}_{[T,T]}$. For a vertex $u \in T$, let the uth column of $\boldsymbol{L}_{[T,T]}$ be $\begin{pmatrix} 3 \\ -\boldsymbol{a} \end{pmatrix}$. Then, we can write $\boldsymbol{L}_{[T,T]}$ into block form as

$$\boldsymbol{L}_{[T,T]} = \begin{pmatrix} 3 & -\boldsymbol{a}^T \\ -\boldsymbol{a} & \boldsymbol{A} \end{pmatrix},$$

where $\boldsymbol{A} \overset{\mathrm{def}}{=} \boldsymbol{L}_{[T-u,T-u]}$. By blockwise matrix inversion we have

$$\left(\boldsymbol{L}_{[T,T]}^{-1}\right)_{[u,u]} = 1/\left(3 - \boldsymbol{a}^T \boldsymbol{A}^{-1} \boldsymbol{a}\right).$$

Since $\boldsymbol{L}_{[T,T]}^{-1}$ is positive definite, we have $1/(3 - \boldsymbol{a}^T \boldsymbol{A}^{-1} \boldsymbol{a}) > 0$ and hence $\boldsymbol{a}^T \boldsymbol{A}^{-1} \boldsymbol{a} < 3$. Since T is a connected component, $\boldsymbol{a}$ is not a zero vector, which coupled with the fact that $\boldsymbol{A}^{-1}$ is positive definite gives $\boldsymbol{a}^T \boldsymbol{A}^{-1} \boldsymbol{a} > 0$. Thus, $1/(3 - \boldsymbol{a}^T \boldsymbol{A}^{-1} \boldsymbol{a}) > 1/3$. Since this holds for all $u \in T$, we have $\mathrm{Tr}\left(\boldsymbol{L}_{[T,T]}^{-1}\right) > |T|/3$. Also, since T can be any connected component of $G[V \setminus S]$ with at least two vertices, and a block of an isolate vertex in $G[V \setminus S]$ contributes a $1/3$ to $\mathrm{Tr}\left((\boldsymbol{L}_{-S})^{-1}\right)$, we have for any S which is not a vertex cover of G

$$\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right) > |V \setminus S|/3 = (n-k)/3,$$

which implies $C(S) = n/\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right) < 3n/(n-k)$. This completes the proof. □

The following theorem then follows by Lemma 4.1.

Theorem 4.2. *Maximizing current flow closeness subject to a cardinality constraint is NP-hard.*

Proof. We give a polynomial reduction

$$p : \{(G = (V, E), k)\} \to \{(G = (V, E, w), k, r)\}$$

from instances of VC3 to instances of CFCMD. For a connected 3-regular graph $G = (V, E)$ with n vertices, we construct a weighted graph $G' = (V, E, w_1)$ with the same vertex set and edge set and an edge weight function $w_1 : E \to \{1\}$ mapping all edges to weight 1. Then, we construct a reduction p as

$$p\left((G = (V, E), k)\right) = (G' = (V, E, w_1), k, 3n/(n-k)).$$

By Lemma 4.1, p is a polynomial reduction from VC3 to CFCMD, which implies that CFCM is NP-hard. □

5 SUPERMODULARITY OF THE RECIPROCAL OF CURRENT FLOW GROUP CLOSENESS

In this section, we prove that the reciprocal of current flow group closeness, i.e., $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)/n$, is a monotone supermodular function. Our proof uses the following lemma, which shows that $\boldsymbol{L}_{-S}^{-1}$ is entrywise supermodular.

Lemma 5.1. *Let $u, v \in V$ be an arbitrary pair of vertices. Then, the entry $\left(\boldsymbol{L}_{-S}^{-1}\right)_{[u,v]}$ is a monotone supermodular function. Namely, for vertices $u, v \neq w$ and nonempty vertex sets $S \subseteq T \subseteq V$ such that $u, v, w \notin T$,*

$$\left(\boldsymbol{L}_{-S}^{-1}\right)_{[u,v]} \geq \left(\boldsymbol{L}_{-T}^{-1}\right)_{[u,v]}$$

and

$$\left(\boldsymbol{L}_{-S}^{-1}\right)_{[u,v]} - \left(\boldsymbol{L}_{-(S+w)}^{-1}\right)_{[u,v]} \geq \left(\boldsymbol{L}_{-T}^{-1}\right)_{[u,v]} - \left(\boldsymbol{L}_{-(T+w)}^{-1}\right)_{[u,v]}.$$

To prove Lemma 5.1, we first define a linear relaxation $\boldsymbol{H} : [0,1]^n \to \mathbb{R}^{n \times n}$ of $\boldsymbol{L}_{-S}$ as

$$(\boldsymbol{H}(\boldsymbol{x}))_{[u,v]} = \begin{cases} \boldsymbol{L}_{[u,u]} & \text{if } u = v, \\ \left(1 - \boldsymbol{x}_{[u]}\right) \cdot \left(1 - \boldsymbol{x}_{[v]}\right) \cdot \boldsymbol{L}_{[u,v]} & \text{if } u \neq v. \end{cases} \quad (1)$$

Remark 5.2. We remark the intuition behind this relaxation $\boldsymbol{H}(\boldsymbol{x})$. Let $\boldsymbol{x}_{[=1]}$ denote the indices of entries of $\boldsymbol{x}$ equal to one, and let $\boldsymbol{x}_{[<1]}$ denote the indices of entries of $\boldsymbol{x}$ less than one. Then, by the definition in (1), we can write $\boldsymbol{H}(\boldsymbol{x})$ into a block diagonal matrix as

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{pmatrix} (\boldsymbol{H}(\boldsymbol{x}))_{[\boldsymbol{x}_{[=1]}, \boldsymbol{x}_{[=1]}]} & \boldsymbol{0} \\ \boldsymbol{0} & (\boldsymbol{H}(\boldsymbol{x}))_{[\boldsymbol{x}_{[<1]}, \boldsymbol{x}_{[<1]}]} \end{pmatrix},$$

where $(\boldsymbol{H}(\boldsymbol{x}))_{[\boldsymbol{x}_{[=1]}, \boldsymbol{x}_{[=1]}]}$ is itself a diagonal matrix. This means that if $\boldsymbol{x} = \boldsymbol{1}_S$ for some nonempty vertex set $S \subseteq V$, the following statement holds:

$$\left(\boldsymbol{H}^{-1}(\boldsymbol{1}_S)\right)_{-S} = \boldsymbol{L}_{-S}^{-1}.$$

The condition that every entry of $\boldsymbol{x}$ is in $[0,1]$ coupled with Fact 2.1 also implies that all submatrices of $\boldsymbol{H}(\boldsymbol{x})$ are positive definite and inverse-positive.

Now for vertices $u, v \neq w$ and nonempty vertex set $S \subseteq V$ such that $u, v, w \notin S$, we can write the marginal gain of a vertex w as

$$\left(\boldsymbol{L}_{-S}^{-1} - \boldsymbol{L}_{-(S+w)}^{-1}\right)_{[u,v]} = \left(\boldsymbol{H}^{-1}(\boldsymbol{1}_S) - \boldsymbol{H}^{-1}(\boldsymbol{1}_{S+w})\right)_{[u,v]}. \quad (2)$$

We can further write the matrix on the rhs of (2) as an integral by

$$\begin{aligned}
&\boldsymbol{H}^{-1}(\mathbf{1}_S) - \boldsymbol{H}^{-1}(\mathbf{1}_{S+w}) = - \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)\Big|_0^1 \\
&= - \int_0^1 \frac{\mathrm{d}\boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)}{\mathrm{d}t} \mathrm{d}t \\
&= \int_0^1 \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w) \frac{\mathrm{d}\boldsymbol{H}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)}{\mathrm{d}t} \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w) \mathrm{d}t \\
&= \int_0^1 \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)\left(-t\boldsymbol{H}(\mathbf{1}_V - \boldsymbol{e}_w)\right) \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w) \mathrm{d}t, \quad (3)
\end{aligned}$$

where the second equality follows by the identity

$$\frac{\mathrm{d}\boldsymbol{A}^{-1}}{\mathrm{d}x} = -\boldsymbol{A}^{-1} \frac{\mathrm{d}\boldsymbol{A}}{\mathrm{d}x} \boldsymbol{A}^{-1}$$

for any invertible matrix $\boldsymbol{A}$. To prove Lemma 5.1, we will also need the following lemma, which shows the entrywise monoticity of $\boldsymbol{H}^{-1}(\boldsymbol{x})$.

Lemma 5.3. *For $0 \leq t \leq 1$, the following statement holds for any vertices $u, v \neq w$ and nonempty vertex sets $S \subseteq T \subseteq V$ such that $u, v, w \notin T$:*

$$\left(\boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)\right)_{[u,v]} \geq \left(\boldsymbol{H}^{-1}(\mathbf{1}_T + t \cdot \boldsymbol{e}_w)\right)_{[u,v]}.$$

Proof. For simplicity, we let $\boldsymbol{S} \overset{\text{def}}{=} \boldsymbol{H}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)$ and $\boldsymbol{T} \overset{\text{def}}{=} \boldsymbol{H}(\mathbf{1}_T + t \cdot \boldsymbol{e}_w)$. We also write $P \overset{\text{def}}{=} V \setminus T$, $Q \overset{\text{def}}{=} V \setminus S$, and $F \overset{\text{def}}{=} T \setminus S$. Due to the block diagonal structures of $\boldsymbol{S}$ and $\boldsymbol{T}$, we have

$$\left(\boldsymbol{S}^{-1}\right)_{[u,v]} = \left(\boldsymbol{S}_{[Q,Q]}^{-1}\right)_{[u,v]} \quad (4)$$

and

$$\left(\boldsymbol{T}^{-1}\right)_{[u,v]} = \left(\boldsymbol{T}_{[P,P]}^{-1}\right)_{[u,v]}. \quad (5)$$

Since $\boldsymbol{S}$ and $\boldsymbol{T}$ agree on entries with indices in P, we can write the submatrix $\boldsymbol{S}_{[Q,Q]}$ of $\boldsymbol{S}$ in block form as

$$\boldsymbol{S}_{[Q,Q]} = \begin{pmatrix} \boldsymbol{T}_{[P,P]} & \boldsymbol{S}_{[P,F]} \\ \boldsymbol{S}_{[F,P]} & \boldsymbol{S}_{[F,F]} \end{pmatrix}.$$

By blockwise matrix inversion, we have

$$\begin{aligned}
&\left(\boldsymbol{S}_{[Q,Q]}^{-1}\right)_{[P,P]} \\
&= \boldsymbol{T}_{[P,P]}^{-1} + \boldsymbol{T}_{[P,P]}^{-1} \boldsymbol{S}_{[P,F]} \left(\boldsymbol{S}_{[Q,Q]}^{-1}\right)_{[F,F]} \boldsymbol{S}_{[F,P]} \boldsymbol{T}_{[P,P]}^{-1} \\
&= \boldsymbol{T}_{[P,P]}^{-1} + \boldsymbol{T}_{[P,P]}^{-1} \left(-\boldsymbol{S}_{[P,F]}\right) \left(\boldsymbol{S}_{[Q,Q]}^{-1}\right)_{[F,F]} \left(-\boldsymbol{S}_{[F,P]}\right) \boldsymbol{T}_{[P,P]}^{-1},
\end{aligned}$$

where the second equality follows by negating both $\boldsymbol{S}_{[P,F]}$ and $\boldsymbol{S}_{[F,P]}$. By definition the matrix $-\boldsymbol{S}_{[P,F]}$ is entrywise nonnegative. By Fact 2.1, every entry of $\boldsymbol{S}_{[Q,Q]}^{-1}$ and $\boldsymbol{T}_{[P,P]}^{-1}$ is also nonnegative. Thus, the matrix

$$\boldsymbol{T}_{[P,P]}^{-1} \left(-\boldsymbol{S}_{[P,F]}\right) \left(\boldsymbol{S}_{[Q,Q]}^{-1}\right)_{[F,F]} \left(-\boldsymbol{S}_{[F,P]}\right) \boldsymbol{T}_{[P,P]}^{-1}$$

is entrywise nonnegative, which coupled with (4) and (5), implies $\left(\boldsymbol{S}^{-1}\right)_{[u,v]} \geq \left(\boldsymbol{T}^{-1}\right)_{[u,v]}$. □

We are now ready to prove Lemma 5.1.

Proof of Lemma 5.1. By definition the matrix $-t\boldsymbol{H}(\mathbf{1}_V - \boldsymbol{e}_w)$ is entrywise nonnegative when $t \geq 0$. By Fact 2.1, the matrix $\boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)$ is also entrywise nonnegative when $0 \leq t \leq 1$. Thus, the derivative in (3) is entrywise nonnegative, which implies the the monotonicity of $\left(\boldsymbol{L}_{-S}^{-1}\right)_{[u,v]}$ for any pair of vertices $u, v \in V$.

We then prove the supermodularity, i.e.,

$$\left(\boldsymbol{L}_{-S}^{-1} - \boldsymbol{L}_{-(S+w)}^{-1}\right)_{[u,v]} \geq \left(\boldsymbol{L}_{-T}^{-1} - \boldsymbol{L}_{-(T+w)}^{-1}\right)_{[u,v]} \quad (6)$$

for any $S \subseteq T \subseteq V$ and $u, v \notin T$. Lemma 5.3, coupled with the fact that $\boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)$ and $-t\boldsymbol{H}(\mathbf{1}_V - \boldsymbol{e}_w)$ are both entrywise nonnegative, gives the entrywise monotonicity of the derivative in (3) as

$$\begin{aligned}
&\left(\boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)\left(-t\boldsymbol{H}(\mathbf{1}_V - \boldsymbol{e}_w)\right) \boldsymbol{H}^{-1}(\mathbf{1}_S + t \cdot \boldsymbol{e}_w)\right)_{[u,v]} \\
&\geq \left(\boldsymbol{H}^{-1}(\mathbf{1}_T + t \cdot \boldsymbol{e}_w)\left(-t\boldsymbol{H}(\mathbf{1}_V - \boldsymbol{e}_w)\right) \boldsymbol{H}^{-1}(\mathbf{1}_T + t \cdot \boldsymbol{e}_w)\right)_{[u,v]}.
\end{aligned}$$

Integrating both sides of the above inequality with respect to t on the interval $[0, 1]$ gives (6). □

The following theorem then directly follows by Lemma 5.1.

Theorem 5.4. *The reciprocal of current flow group centrality, i.e., $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)/n$, is a monotone supermodular function.*

We note that [15] has previously proved that $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)$ is monotone and supermodular by using the connection between effective resistance and commute time for random walks. However, our proof is fully algebraic. Moreover, we present a more general result that $\boldsymbol{L}_{-S}^{-1}$ is entrywise supermodular.

By Theorem 5.4, one can obtain a $(1 - \frac{k}{k-1} \cdot \frac{1}{e})$-approximation to the optimum $\mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)$ by a simple greedy algorithm, by picking the vertex with the maximum marginal gain each time [42]. However, since computing $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)$ involves matrix inversions, a naive implementation of this greedy algorithm will take $O(kn^4)$ time, assuming that one matrix inversion runs in $O(n^3)$ time. We will show in the next section how to implement this greedy algorithm in $O(n^3)$ time using blockwise matrix inversion.

6 A DETERMINISTIC GREEDY ALGORITHM

We now consider how to accelerate the naive greedy algorithm. Suppose that after the i^{th} step, the algorithm has selected a set S_i containing i vertices. We next compute the marginal gain $\mathrm{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-(S_i+u)}^{-1}\right)$ of each vertex $u \notin S_i$.

For a vertex $u \notin S_i$, let $\begin{pmatrix} d_u \\ -\boldsymbol{a} \end{pmatrix}$ denote the u^{th} column of the submatrix $\boldsymbol{L}_{-S_i}$. Then we write $\boldsymbol{L}_{-S_i}$ in block form as $\boldsymbol{L}_{-S_i} = \begin{pmatrix} d_u & -\boldsymbol{a}^T \\ -\boldsymbol{a} & \boldsymbol{A} \end{pmatrix}$, where $\boldsymbol{A} \overset{\text{def}}{=} \boldsymbol{L}_{-(S_i+u)}$. By blockwise matrix inversion, we have

$$\boldsymbol{L}_{-S_i}^{-1} = \begin{pmatrix} \frac{1}{s} & \frac{1}{s}\boldsymbol{a}^T\boldsymbol{A}^{-1} \\ \frac{1}{s}\boldsymbol{A}^{-1}\boldsymbol{a} & \boldsymbol{A}^{-1} + \frac{1}{s}\boldsymbol{A}^{-1}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{A}^{-1} \end{pmatrix}, \quad (7)$$

where $s = d_u - \boldsymbol{a}^T \boldsymbol{A}^{-1}\boldsymbol{a}$. Then the marginal gain of u can be further expressed as

$$\begin{aligned}
&\operatorname{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-(S_i+u)}^{-1}\right)\\
&= \left(\boldsymbol{L}_{-S_i}^{-1}\right)_{[u,u]} + \sum_{v \in V \setminus (S_i+u)} \left(\left(\boldsymbol{L}_{-S_i}^{-1}\right)_{[v,v]} - \left(\boldsymbol{L}_{-(S_i+u)}^{-1}\right)_{[v,v]}\right)\\
&= \frac{1}{s} + \frac{1}{s}\operatorname{Tr}\left(\boldsymbol{A}^{-1}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{A}^{-1}\right) = \frac{1}{s} + \frac{1}{s}\boldsymbol{a}^T\boldsymbol{A}^{-1}\boldsymbol{A}^{-1}\boldsymbol{a}\\
&= \left(\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-2}\boldsymbol{e}_u\right) / \left(\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_u\right),
\end{aligned}$$

where the second equality and the fourth equality follow by (7), while the third equality follows by the cyclicity of trace.

By (7), we can also update the inverse $\boldsymbol{L}_{-S_i}^{-1}$ upon a vertex u by

$$\boldsymbol{L}_{-(S_i+u)}^{-1} = \left(\boldsymbol{L}_{-S_i}^{-1} - \left(\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_u\boldsymbol{e}_u^T\boldsymbol{L}_{-S_i}^{-1}\right) / \left(\boldsymbol{e}_u^T\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_u\right)\right)_{-u}.$$

At the first step, we need to pick a vertex u_1 with minimum $\sum_{v \in V} R_{\text{eff}}(u_1, v)$, which can be done by computing $\sum_{v\in V} R_{\text{eff}}(u, v)$ for all $u \in V$ using the relation [10]

$$\sum_{v \in V} R_{\text{eff}}(u, v) = n\left(\boldsymbol{L}^\dagger\right)_{[u,u]} + \operatorname{Tr}\left(\boldsymbol{L}^\dagger\right).$$

We give the $O(n^3 + kn^2)$-time algorithm as follows.

```
$S_k$ = ExactGreedy $(G, \boldsymbol{L}, k)$

  (1) Compute $\boldsymbol{L}^\dagger$ by inverting $\boldsymbol{L}$ in $O(n^3)$ time.
  (2) $S_1 \leftarrow \{u_1\}$ where $u_1 = \arg\min_{u \in V} n\left(\boldsymbol{L}^\dagger\right)_{[u,u]} + \operatorname{Tr}\left(\boldsymbol{L}^\dagger\right)$.
  (3) Compute $\boldsymbol{L}_{-S_1}^{-1}$ in $O(n^3)$ time.
  (4) Repeat the following steps for $i = 1, \ldots, k-1$:
      (a) $u_{i+1} \leftarrow \arg\max_{u \in (V \setminus S_i)} \frac{\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-2}\boldsymbol{e}_u}{\boldsymbol{e}_u^T\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_u}$, $S_{i+1} \leftarrow S_i + u_{i+1}$
      (b) Compute $\boldsymbol{L}_{S_{i+1}}^{-1}$ in $O(n^2)$ time by
          $\boldsymbol{L}_{-(S_i+u_{i+1})}^{-1} = \left(\boldsymbol{L}_{-S_i}^{-1} - \frac{\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_{u_{i+1}}\boldsymbol{e}_{u_{i+1}}^T\boldsymbol{L}_{-S_i}^{-1}}{\boldsymbol{e}_{u_{i+1}}^T\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_{u_{i+1}}}\right)_{-u_{i+1}}$.
  (5) Return $S_k$.
```

The performance of ExactGreedy is characterized in the following theorem.

Theorem 6.1. *The algorithm S_k = ExactGreedy$(G, \boldsymbol{L}, k)$ takes an undirected positive weighted graph $G = (V, E, w)$ with associated Laplacian $\boldsymbol{L}$ and an integer $2 \le k \le n$, and returns a vertex set $S_k \subseteq V$ with $|S_k| = k$. The algorithm runs in time $O(n^3)$. The vertex set S_k satisfies*

$$\operatorname{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S_k}^{-1}\right) \ge \left(1 - \frac{k}{k-1}\cdot\frac{1}{e}\right)\left(\operatorname{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right)$$

where

$$S^* \stackrel{\text{def}}{=} \arg\min_{|S| \le k} \operatorname{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right), \quad u^* \stackrel{\text{def}}{=} \arg\min_{u \in V} \sum_{v \in V} R_{\text{eff}}(u, v).$$

Proof. The running time is easy to verify. We only need to prove the approximation ratio.

By supermodularity, for any $i \ge 1$

$$\operatorname{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S_{i+1}}^{-1}\right) \ge \frac{1}{k}\left(\operatorname{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S*}^{-1}\right)\right),$$

which implies

$$\operatorname{Tr}\left(\boldsymbol{L}_{-S_{i+1}}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right) \le \left(1 - \frac{1}{k}\right)\left(\operatorname{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right).$$

Then, we have

$$\begin{aligned}
&\operatorname{Tr}\left(\boldsymbol{L}_{-S_k}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\\
&\le \left(1 - \frac{1}{k}\right)^{k-1}\left(\operatorname{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right)\\
&\le \frac{k}{k-1}\cdot\frac{1}{e}\left(\operatorname{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right),
\end{aligned}$$

which coupled with $\operatorname{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) = \operatorname{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right)$ finishes the proof. □

7 A RANDOMIZED GREEDY ALGORITHM

The deterministic greedy algorithm ExactGreedy has a time complexity $O(n^3)$, which is still not acceptable for large networks. In this section, we provide an efficient randomized algorithm, which achieves a $(1 - \frac{k}{k-1}\cdot\frac{1}{e} - \epsilon)$ approximation factor in time $\widetilde{O}(km\epsilon^{-2})$.

To further accelerate algorithm ExactGreedy we need to compute the marginal gains

$$\operatorname{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \operatorname{Tr}\left(\boldsymbol{L}_{-(S_i+u)}^{-1}\right) = \frac{\boldsymbol{e}_u^T\boldsymbol{L}_{-S_i}^{-2}\boldsymbol{e}_u}{\boldsymbol{e}_u^T\boldsymbol{L}_{-S_i}^{-1}\boldsymbol{e}_u} \tag{8}$$

for all $u \in V$ and a vertex set $S_i \subseteq V$ more quickly. We also need a faster way to compute

$$\sum_{v \in V} R_{\text{eff}}(u, v) = n\left(\boldsymbol{L}^\dagger\right)_{[u,u]} + \operatorname{Tr}\left(\boldsymbol{L}^\dagger\right) \tag{9}$$

for all $u \in V$ at the 1st step. We will show how to solve both problems in nearly linear time using Johnson-Lindenstrauss Lemma and Fast SDDM Solvers. Our routines are motivated by the effective resistance estimation routine in [33, 49].

Lemma 7.1 (Johnson-Lindenstrauss Lemma [29]). *Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n \in \mathbb{R}^d$ be fixed vectors and $0 < \epsilon < 1$ be a real number. Let q be a positive integer such that*

$$q \ge 4(\epsilon^2/2 - \epsilon^3/3)^{-1}\ln n$$

and $\boldsymbol{Q}_{q\times d}$ be a random matrix obtained by first choosing each of its entries from Gaussian distribution $N(0, 1)$ independently and then normalizing each of its columns to a length of 1. With high probability, the following statement holds for any $1 \le i, j \le n$:

$$(1-\epsilon)\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2 \le \left\|\boldsymbol{Q}\boldsymbol{v}_i - \boldsymbol{Q}\boldsymbol{v}_j\right\|^2 \le (1+\epsilon)\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2.$$

Lemma 7.2 (Fast SDDM Solvers [16, 48]). *There is a routine $\boldsymbol{x}$ = Solve$(\boldsymbol{S}, \boldsymbol{b}, \epsilon)$ which takes a Laplacian or an SDDM matrix $\boldsymbol{S}_{n\times n}$ with m nonzero entries, a vector $\boldsymbol{b} \in \mathbb{R}^n$, and an error parameter $\delta > 0$, and returns a vector $\boldsymbol{x} \in \mathbb{R}^n$ such that*

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

holds with high probability, where $\|\boldsymbol{x}\|_{\boldsymbol{S}} \stackrel{\text{def}}{=} \sqrt{\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x}}$, and $\boldsymbol{S}^{-1}$ denotes the pseudoinverse of $\boldsymbol{S}$ when $\boldsymbol{S}$ is a Laplacian. The routine runs in expected time $\widetilde{O}(m\log(1/\delta))$.

7.1 Approximation of (9)

To approximate (9) we need to approximate all diagonal entries of $\boldsymbol{L}^{\dagger}$, i.e., $\boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{e}_u$ for all $u \in V$. We first write $\boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{e}_u$ in an Euclidean norm as

$$\begin{aligned}\boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{e}_u &= \boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{L} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u = \boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u \\ &= \boldsymbol{e}_u^T \boldsymbol{L}^{\dagger} \boldsymbol{B}^T \boldsymbol{W}^{1/2} \boldsymbol{W}^{1/2} \boldsymbol{B} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u = \left\| \boldsymbol{W}^{1/2} \boldsymbol{B} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u \right\|^2.\end{aligned}$$

Then we use Johnson-Lindenstrauss Lemma to reduce the dimensions. Let $\boldsymbol{Q}_{q\times m}$ be a random Gaussian matrix where $q = \lceil 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \rceil$. By Lemma 7.1,

$$\left\| \boldsymbol{W}^{1/2} \boldsymbol{B} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u \right\|^2 \approx_{1+\epsilon} \left\| \boldsymbol{Q} \boldsymbol{W}^{1/2} \boldsymbol{B} \boldsymbol{L}^{\dagger} \boldsymbol{e}_u \right\|^2$$

holds for all $u \in V$ with high probability. Here we can use sparse matrix multiplication to compute $\boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}$, and Fast SDDM solvers to compute $\boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}\boldsymbol{L}^{\dagger}$.

We give the routine for approximating (9) as follows.

```
$\{r_u\}_{u\in V}$ = ERSumsEst $(G, \boldsymbol{L}, \epsilon)$
  (1) Set $\delta = \frac{\epsilon}{9n^2} \left( \frac{(1-\epsilon/3) w_{\min}}{(1+\epsilon/3) w_{\max}} \right)^{1/2}$.
  (2) Generate a random Gaussian matrix $\boldsymbol{Q}_{q\times m}$ where $q = \lceil 4((\epsilon/3)^2/2 - (\epsilon/3)^3/3)^{-1} \ln n \rceil$.
  (3) Compute $\left(\boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}\right)_{q\times n}$ by sparse matrix multiplication in $O(qm)$ time.
  (4) Compute an approximation $\widetilde{\boldsymbol{Z}}_{q\times n}$ to $\boldsymbol{Z}_{q\times n} \stackrel{\text{def}}{=} \boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}\boldsymbol{L}^{\dagger}$ by
          $\widetilde{\boldsymbol{Z}}_{[i,:]} \leftarrow$ Solve$(\boldsymbol{L}, \left(\boldsymbol{Q}\boldsymbol{W}^{1/2}\boldsymbol{B}\right)^T_{[i,:]}, \delta)^T$
      where $1 \le i \le q$.
  (5) $r_u \leftarrow \left\|\widetilde{\boldsymbol{Z}}\right\|_F^2 + n \left\|\widetilde{\boldsymbol{Z}}\boldsymbol{e}_u\right\|^2$ for all $u \in V$ and return $\{r_u\}_{u\in V}$.
```

The performance of ERSumsEst is characterized in the following lemma.

Lemma 7.3. *The routine ERSumsEst runs in time $\widetilde{O}(m)$. For $0 < \epsilon \le 1/2$, the $\{r_u\}_{u\in V}$ returned by ERSumsEst satisfies*

$$(1-\epsilon) \sum\nolimits_{v\in V} R_{\text{eff}}(u,v) \le r_u \le (1+\epsilon) \sum\nolimits_{v\in V} R_{\text{eff}}(u,v)$$

with high probability.

7.2 Approximation of (8)

We first approximate the numerator of (8), which can be recast in an euclidian norm as $\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-2} \boldsymbol{e}_u = \left\| \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2$. We then once again use Johnson-Lindenstrauss Lemma to reduce the dimensions. Let $\boldsymbol{P}_{p\times n}$ be a random Gaussian matrix where $p = \lceil 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \rceil$. By Lemma 7.1, we have that for all $u \in V$,

$$(1-\epsilon) \left\| \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2 \le \left\| \boldsymbol{P}\boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2 \le (1+\epsilon) \left\| \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2,$$

where $\boldsymbol{P}\boldsymbol{L}_{-S_i}^{-1}$ can be computed using Fast SDDM Solvers.

We continue to approximate the denominator of (8). Since $\boldsymbol{L}_{-S_i}$ is an SDDM matrix, we can express it in terms of the sum of a Laplacian and a nonnegative diagonal matrix as $\boldsymbol{L}_{-S_i} = \boldsymbol{B}'^T \boldsymbol{W}' \boldsymbol{B}' + \boldsymbol{X}$. Then we can write

$$\begin{aligned} &\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u = \boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{L}_{-S_i} \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \\ =& \boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \left(\boldsymbol{B}'^T \boldsymbol{W}' \boldsymbol{B}' + \boldsymbol{X} \right) \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \\ =& \boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{B}'^T \boldsymbol{W}' \boldsymbol{B}' \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u + \boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{X} \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \\ =& \left\| \boldsymbol{W}'^{1/2} \boldsymbol{B}' \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2 + \left\| \boldsymbol{X}^{1/2} \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|^2. \end{aligned} \tag{10}$$

Let $\boldsymbol{Q}_{q\times m}$ and $\boldsymbol{R}_{r\times n}$ be random Gaussian matrices where

$$q = r = \lceil 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \rceil.$$

By Lemma 7.1, we have for all $u \in V$

$$(10) \approx_{1+\epsilon} \left\| \boldsymbol{Q}\boldsymbol{W}'^{1/2} \boldsymbol{B}' \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|_2^2 + \left\| \boldsymbol{R}\boldsymbol{X}^{1/2} \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u \right\|_2^2.$$

Here $\boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S_i}^{-1}$ and $\boldsymbol{R}\boldsymbol{X}^{1/2}\boldsymbol{L}_{-S_i}^{-1}$ can be computed using Fast SDDM Solvers.

We give the routine for approximating (8) as follows.

```
$\{g_u\}_{u\in V}$ = GainsEst $(G, \boldsymbol{L}, S_i, \epsilon)$
  (1) Set $\delta_1 = \frac{w_{\min}\epsilon}{27 w_{\max} n^4} \left( \frac{(1-\epsilon/9)}{(1+\epsilon/9)n} \right)^{1/2}$ and $\delta_2 = \delta_3 = \frac{1}{4n^4 w_{\max}} \left( \frac{\epsilon w_{\min}^{3/2}}{9n} \right)^{1/2}$.
  (2) Generate random Gaussian matrices $\boldsymbol{P}_{p\times n}, \boldsymbol{Q}_{q\times m}, \boldsymbol{R}_{r\times n}$ where
          $p, q, r = \lceil 4((\epsilon/9)^2/2 - (\epsilon/9)^3/3)^{-1} \ln n \rceil$.
  (3) Let $\boldsymbol{X} = \text{Diag}\left(\boldsymbol{L}_{-S_i}\boldsymbol{1}\right)$ and denote $\boldsymbol{L}_{-S_i} - \boldsymbol{X}$ by $\boldsymbol{B}'^T\boldsymbol{W}'\boldsymbol{B}'$.
  (4) Compute $\boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'$ and $\boldsymbol{R}\boldsymbol{X}^{1/2}$ by sparse matrix multiplication in $O(pm)$ time.
  (5) Compute approximations $\widetilde{\boldsymbol{Z}}^{(1)}$ to $\boldsymbol{Z}^{(1)} \stackrel{\text{def}}{=} \boldsymbol{P}\boldsymbol{L}_{-S_i}^{-1}$, $\widetilde{\boldsymbol{Z}}^{(2)}$ to $\boldsymbol{Z}^{(2)} \stackrel{\text{def}}{=} \boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'\boldsymbol{L}_{-S_i}^{-1}$, and $\widetilde{\boldsymbol{Z}}^{(3)}$ to $\boldsymbol{Z}^{(3)} \stackrel{\text{def}}{=} \boldsymbol{R}\boldsymbol{X}^{1/2}\boldsymbol{L}_{-S_i}^{-1}$ by
      (a) $\widetilde{\boldsymbol{Z}}^{(1)}_{[i,:]} \leftarrow$ Solve$(\boldsymbol{L}_{-S_i}, \boldsymbol{P}^T_{[i,:]}, \delta_1)^T$,
      (b) $\widetilde{\boldsymbol{Z}}^{(2)}_{[i,:]} \leftarrow$ Solve$(\boldsymbol{L}_{-S_i}, \left(\boldsymbol{Q}\boldsymbol{W}'^{1/2}\boldsymbol{B}'\right)^T_{[i,:]}, \delta_2)^T$, and
      (c) $\widetilde{\boldsymbol{Z}}^{(3)}_{[i,:]} \leftarrow$ Solve$(\boldsymbol{L}_{-S_i}, \left(\boldsymbol{R}\boldsymbol{X}^{1/2}\right)^T_{[i,:]}, \delta_3)^T$.
  (6) $g_u \leftarrow \frac{\left\|\widetilde{\boldsymbol{Z}}^{(1)}\boldsymbol{e}_u\right\|^2}{\left\|\widetilde{\boldsymbol{Z}}^{(2)}\boldsymbol{e}_u\right\|^2 + \left\|\widetilde{\boldsymbol{Z}}^{(3)}\boldsymbol{e}_u\right\|^2}$ for all $u \in V$ and return $\{g_u\}_{u\in V}$.
```

Lemma 7.4. *The routine GainsEst runs in time $\widetilde{O}(m)$. For $0 \le \epsilon \le 1/2$, the $\{g_u\}_{u\in V}$ returned by GainsEst satisfies*

$$(1-\epsilon) \frac{\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-2} \boldsymbol{e}_u}{\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u} \le g_u \le (1+\epsilon) \frac{\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-2} \boldsymbol{e}_u}{\boldsymbol{e}_u^T \boldsymbol{L}_{-S_i}^{-1} \boldsymbol{e}_u}$$

with high probability.

We now give the $\widetilde{O}(mk)$-time greedy algorithm as follows.

```
$S_k$ = ApproxGreedy $(G, \boldsymbol{L}, k, \epsilon)$

  (1) $\{r_u\}_{u \in V} \leftarrow$ ERSumsEst $(G, \boldsymbol{L}, \epsilon/3)$
  (2) $S_1 \leftarrow \{u_1\}$ where $u_1 = \arg\min_{u \in V} r_u$.
  (3) Repeat the following steps for $i = 1, \ldots, k-1$:
    (a) $\{g_u\}_{u \in V} \leftarrow$ GainsEst $(G, \boldsymbol{L}, S_i, \epsilon/2)$
    (b) $S_{i+1} \leftarrow S_i + u_{i+1}$ where
            $u_{i+1} = \arg\max_{u \in (V \setminus S_i)} g_u$.
  (4) Return $S_k$.
```

The performance of ApproxGreedy is characterized in the following theorem.

Theorem 7.5. *The algorithm S_k = ApproxGreedy$(G, \boldsymbol{L}, k, \epsilon)$ takes an undirected positively weighted graph $G = (V, E, w)$ with associated Laplacian $\boldsymbol{L}$, an integer $2 \leq k \leq n$, and an error parameter $0 < \epsilon \leq 1/2$, and returns a vertex set $S_k \subseteq V$ with $|S_k| = k$. The algorithm runs in time $\widetilde{O}(km)$. With high probability, the vertex set S_k satisfies*

$$(1+\epsilon)\mathrm{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S_k}^{-1}\right) \geq \left(1 - \frac{k}{k-1} \cdot \frac{1}{e} - \epsilon\right)\left((1+\epsilon)\mathrm{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right),$$

where

$$S^* \stackrel{\mathrm{def}}{=} \arg\min_{|S| \leq k} \mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right), \quad u^* \stackrel{\mathrm{def}}{=} \arg\min_{u \in V} \sum_{v \in V} R_{\mathrm{eff}}(u, v).$$

Proof. The running time is easy to verify. It follows by the running times in Lemma 7.3 and 7.4.

We now prove the approximation ratio. The main difference between ApproxGreedy and ExactGreedy is that at each step, ExactGreedy picks a vertex with maximum marginal gain, while ApproxGreedy picks a vertex with at least $\frac{1-\epsilon/2}{1+\epsilon/2} \geq 1 - \epsilon$ times maximum marginal gain. By supermodularity we have for any $i \geq 1$

$$\mathrm{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S_{i+1}}^{-1}\right) \geq \frac{1-\epsilon}{k}\left(\mathrm{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right),$$

which implies

$$\mathrm{Tr}\left(\boldsymbol{L}_{-S_{i+1}}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right) \leq \left(1 - \frac{1-\epsilon}{k}\right)\left(\mathrm{Tr}\left(\boldsymbol{L}_{-S_i}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right).$$

Then, we have

$$\begin{aligned} &\mathrm{Tr}\left(\boldsymbol{L}_{-S_k}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right) \\ &\leq \left(1 - \frac{1-\epsilon}{k}\right)^{k-1}\left(\mathrm{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right) \\ &\leq \frac{k}{k-1} \cdot \frac{1}{e^{(1-\epsilon)}}\left(\mathrm{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right) \\ &\leq \frac{k}{k-1} \cdot \left(\frac{1}{e} + \epsilon\right)\left(\mathrm{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) - \mathrm{Tr}\left(\boldsymbol{L}_{-S^*}^{-1}\right)\right). \end{aligned}$$

By Lemma 7.3, we obtain

$$\mathrm{Tr}\left(\boldsymbol{L}_{-S_1}^{-1}\right) \leq \frac{1+\epsilon/3}{1-\epsilon/3}\mathrm{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right) \leq (1+\epsilon)\,\mathrm{Tr}\left(\boldsymbol{L}_{-u^*}^{-1}\right),$$

which plus the above inequality finishes the proof. □

8 EXPERIMENTS

In this section, we study the performance of our algorithms by conducting experiments on some classic network models and real-world networks taken from KONECT [34] and SNAP [36]. We run our experiments on the largest components of these networks. Related information of these networks is shown in Table 1, where networks are shown in increasing order of their numbers of vertices.

Table 1: Information of datasets. For a network with n vertices and m edges, we use n' and m' to denote the number of vertices and edges in its largest connected component, respectively.

Network	n	m	n'	m'
Zachary karate club	34	78	34	78
Windsufers	43	336	43	336
Contiguous USA	49	107	49	107
Barabási-Albert	50	94	50	94
Watts-Strogatz	50	100	50	100
Erdös-Rényi	50	95	50	95
Regular ring lattice	50	100	50	100
Dolphins	62	159	62	159
David Copperfield	112	425	112	425
Jazz musicians	198	2742	195	1814
Virgili	1,133	5,451	1,133	5,451
Euroroad	1,174	1,417	1,039	1,305
Protein	1,870	2,277	1,458	1,948
Hamster full	2,426	16,631	2,000	16,098
ego-Facebook	2,888	2,981	2,888	2,981
Vidal	3,133	6,726	2,783	6,007
Powergrid	4,941	6,594	4,941	6,594
Reactome	6,327	147,547	5,973	145,778
ca-HepTh	9,877	25,998	8,638	24,806
PG-Privacy	10,680	24,316	10,680	24,316
CAIDA	26,475	53,381	26,475	53,381
ego-Twitter	81,306	1,342,296	81,306	1,342,296
com-DBLP	317,080	1,049,866	317,080	1,049,866
roadNet-PA	1,087,562	1,541,514	1,087,562	1,541,514
com-Youtube	1,134,890	2,987,624	1,134,890	2,987,624
roadNet-TX	1,379,917	1,921,660	1,351,137	1,879,201
roadNet-CA	1,965,206	2,766,607	1,957,027	2,760,388

We implement our algorithms in Julia to facilitate interactions with the SDDM solver contained in the Laplacian.jl package[2]. All of our experiments were run on a Linux box with 4.2 GHz Intel i7-7700 CPU and 32G memory, using a single thread.

8.1 Accuracy of routine GainsEst

We first show the accuracy of routine GainsEst. We use quantity JLfactor to denote $q/\ln n$ in GainsEst. We run GainsEst with different JLfactors on network arXiv High Energy Physics - Theory collaboration network (ca-HepTh). And we set $S_i = \{u^*\}$ where

$$u^* \stackrel{\mathrm{def}}{=} \arg\min_{u \in V} \sum_{v \in V} R_{\mathrm{eff}}(u, v).$$

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

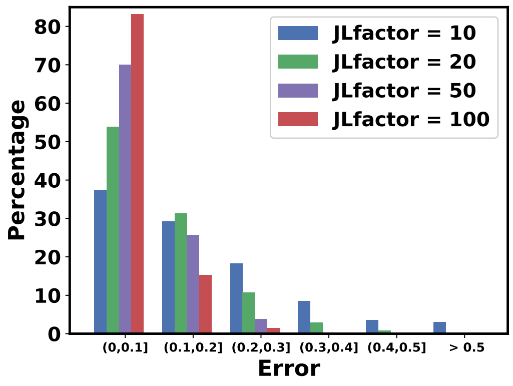

Figure 1: Error distribution for routine GainsEst with different JLfactors on network ca-HepTh.

For each JLfactor, we compute the relative error of gain of each vertex $u \neq u^*$, given by

$$\frac{\left|g_u - \left(e_u^T L_{-u^*}^{-2} e_u\right) / \left(e_u^T L_{-u^*}^{-1} e_u\right)\right|}{\left(e_u^T L_{-u^*}^{-2} e_u\right) / \left(e_u^T L_{-u^*}^{-1} e_u\right)}.$$

We then draw the distribution of relative errors with different JLfactors in a histogram. The results are shown in Figure 1. We observe that the errors become small when JLfactor gets larger. Moreover, almost all errors are in the range $(0, 0.5]$ when JLfactor ≥ 20.

We set JLfactor $= 20$ in all other experiments. We will show that this is sufficient for our greedy algorithm to obtain good solutions empirically.

8.2 Effectiveness of Greedy Algorithms

We show the effectiveness of our algorithms by comparing the results of our algorithms with the optimum solutions on four small model networks (Barabási-Albert (BA) [2], Watts-Strogatz (WS) [52], Erdös-Rényi (ER) [20], and a regular ring lattice [52]) and four small realistic networks (Dolphins, Contiguous USA, Zachary karate club, and Windsurfers). We are able to compute the optimum solutions on these networks because of their small sizes.

For each $k = 1, 2, \ldots, 6$, we first find the set of k vertices with the optimum current flow closeness by brute-force search. We then compute the current flow closeness of the vertex sets returned by ExactGreedy and ApproxGreedy. Also, we compute the solutions returned by the random scheme, which chose k vertices uniformly at random. The results are shown in Figure 2 and 3. We observe that the current flow closenesses of the solutions returned by our two greedy algorithms and the optimum solution are almost the same. This means that the approximation ratios of our greedy algorithms are significantly better than their theoretical guarantees. Moreover, the solutions returned by our greedy algorithms are much better than those returned by the random scheme.

We then demonstrate the effectiveness of our algorithms further by comparing it to the random scheme, as well as two other schemes, Top-degree and Top-cent, on six larger networks. Here the Top-degree scheme chooses k vertices with highest degrees, while the Top-cent scheme chooses k vertices with highest current flow closeness centrality according to Definition 2.4. A comparison of results for these five algorithms is shown in Figure 4. We observe

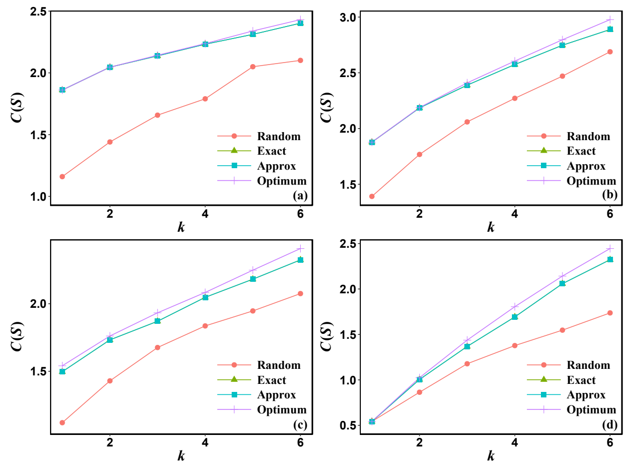

Figure 2: Current flow closeness of vertex sets returned by ExactGreedy, ApproxGreedy, random and optimum strategies on four models: BA (a), WS (b), ER (c), and a regular ring lattice (d).

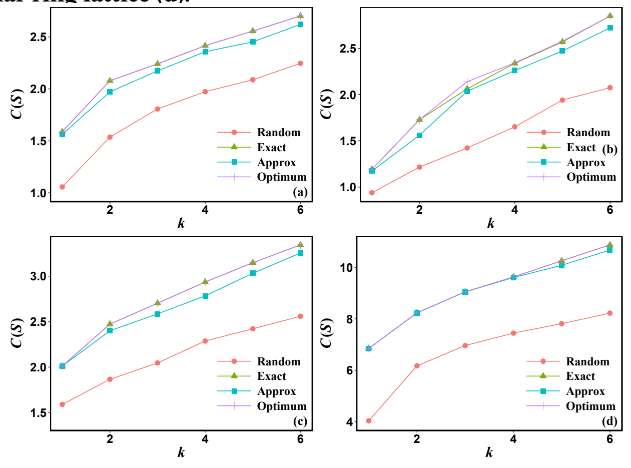

Figure 3: Current flow closeness of vertex sets returned by ExactGreedy, ApproxGreedy, random and optimum strategies on four networks: Dolphins (a), Contiguous USA (b), Zachary karate club (c), and Windsurfers (d).

that our two greedy algorithms obtain similar approximation ratios, and both outperform the other three schemes (random, Top-degree, and Top-cent).

8.3 Efficiency of Greedy Algorithms

We now show that algorithm ApproxGreedy runs much faster than ExactGreedy, especially on large-scale networks. We run our two greedy algorithms on a larger set of real-world networks. For each network, we use both greedy algorithms to choose $k = 10$ vertices, and then compare their running times as well as the solutions they returned. For networks with more than 30000 vertices, we compute the current flow closeness of the vertex set returned by ApproxGreedy by combining Fast Laplacian solvers and Hutchinson's trace estimation [1, 26]. We list the results in Table 2. We observe that the ratio of the running time of ExactGreedy to

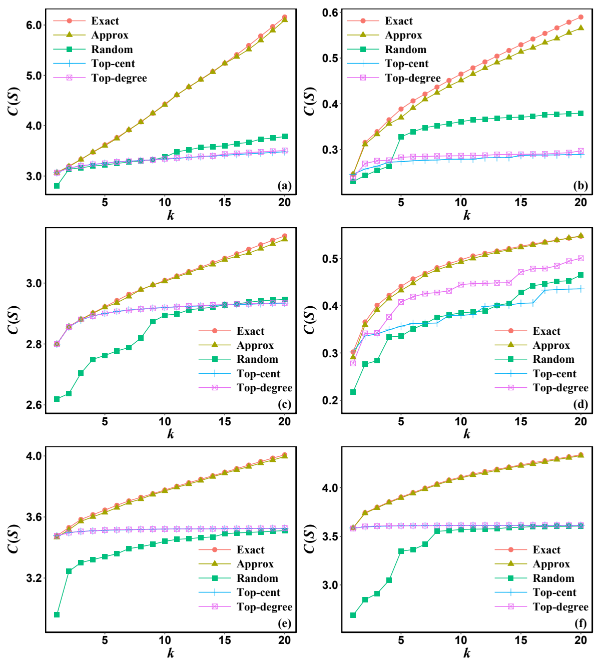

Figure 4: Current flow closeness of vertex sets returned versus the number k of vertices chosen for the five algorithms on David Copperfield (a), Euroroad (b), U. Rovira i Virgili (c), US power grid (d), Hamsterster full (e), and Reactome (f).

Table 2: The average running times and results of ExactGreedy (Exact) and ApproxGreedy (Approx) algorithms on a larger set of real-world networks, as well as the ratios of running times of ExactGreedy to those of ApproxGreedy, and ratios of current-flow closenesses achieved by ApproxGreedy to those achieved by ExactGreedy.

Network	Time (seconds)			Current flow closeness		
	Exact	Approx	Ratio	Exact	Approx	Ratio
Protein	1.35	1.34	1.01	0.7451	0.7388	0.9915
Vidal	8.04	3.40	2.36	1.3179	1.3138	0.9968
ego-Facebook	13.26	2.56	5.17	1.0195	1.0195	1.00
ca-Hepth	196.67	15.80	12.44	1.5340	1.5267	0.9952
PG-Privacy	367.71	17.25	21.31	0.7155	0.7141	0.9980
CAIDA	5530.32	30.64	464.96	1.3948	1.3945	0.9997
ego-Twitter	-	562.27	-	-	5.5052	-
com-DBLP	-	1700.13	-	-	1.6807	-
com-Youtube	-	6374.02	-	-	1.0382	-
roadNet-PA	-	9802.54	-	-	0.3089	-
roadNet-TX	-	13671.05	-	-	0.2260	-
roadNet-CA	-	22853.20	-	-	0.2630	-

that of ApproxGreedy increases rapidly when the network size becomes larger, while the vertex sets they returned have approximately the same current flow closeness. Moreover, ApproxGreedy is able to solve current flow closeness maximization for networks with more than 10^6 vertices in a few hours, whereas ExactGreedy fails because of its high time and space complexity.

9 CONCLUSIONS

In this paper, we extended the notion of current flow closeness centrality (CFCC) to a group of vertices. For a vertex group S in an n-vertex graph with m edges, its CFCC $C(S)$ equals the ratio of n to the sum of effective resistances between S and all other vertices. We then considered the problem of finding the set S^* of k vertices with an aim to maximize $C(S^*)$, and solved it by considering an equivalent problem of minimizing $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)/n$. We showed that the problem is NP-hard, and proved that the objective function is monotone and supermodular. We devised two approximation algorithms for minimizing $\mathrm{Tr}\left(\boldsymbol{L}_{-S}^{-1}\right)/n$ by iteratively selected k vertices in a greedy way. The first one achieves a $(1-\frac{k}{k-1}\cdot\frac{1}{e})$ approximation ratio in time $O(n^3)$; while the second one obtains a $(1-\frac{k}{k-1}\cdot\frac{1}{e}-\epsilon)$ approximation factor in time $\widetilde{O}(mk\epsilon^{-2})$. We conducted extensive experiments on model and realistic networks, the results of which show that both algorithms can often give almost optimal solutions. In particular, our second algorithm is able to scale to huge networks, and quickly gives good approximate solutions in networks with more than 10^6 vertices. In future works, we plan to introduce and study the betweenness group centrality based on current flow [43], which takes into account all possible paths.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R & D Program of China (No. 2018YFB1305104), the National Natural Science Foundation of China (No. 61872093), the National Science Foundation (No. 1637566), and Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

REFERENCES

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

[2] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.

[3] Alex Bavelas. 1948. A mathematical model for group structures. *Hum. Oran.* 7, 3 (1948), 16–30.

[4] Alex Bavelas. 1950. Communication patterns in task-oriented groups. *J. Acoust. Soc. Am.* 22, 6 (1950), 725–730.

[5] Michele Benzi and Christine Klymko. 2015. On the limiting behavior of parameter-dependent network centrality measures. *SIAM J. Matrix Anal. Appl.* 36, 2 (2015), 686–706.

[6] Elisabetta Bergamini, Tanya Gonser, and Henning Meyerhenke. 2018. Scaling up Group Closeness Maximization. In *Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX '18)*. SIAM, 209–222.

[7] Elisabetta Bergamini, Michael Wegner, Dimitar Lukarski, and Henning Meyerhenke. 2016. Estimating Current-Flow Closeness Centrality with a Multigrid Laplacian Solver. In *Proceedings of the 7th SIAM Workshop on Combinatorial Scientific Computing (CSC '16)*. SIAM, 1–12.

[8] Paolo Boldi and Sebastiano Vigna. 2014. Axioms for centrality. *Internet Math.* 10, 3-4 (2014), 222–262.

[9] Francesco Bonchi, Gianmarco De Francisci Morales, and Matteo Riondato. 2016. Centrality measures on big graphs: Exact, approximated, and distributed algorithms. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. ACM, 1017–1020.

[10] Enrico Bozzo and Massimo Franceschet. 2013. Resistance distance, closeness, and betweenness. *Soc. Networks* 35, 3 (2013), 460–469.

[11] Ulrik Brandes and Daniel Fleischer. 2005. Centrality Measures Based on Current Flow. In *22nd Annual Symposium on Theoretical Aspects of Computer Science, Proceedings (STACS '05)*. Springer, 533–544.

[12] Chen Chen, Hanghang Tong, B Aditya Prakash, Charalampos E Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. 2016. Node immunization on large graphs: Theory and algorithms. *IEEE Trans. Knowl. Data Eng.* 28, 1 (2016), 113–126.

[13] Chen Chen, Wei Wang, and Xiaoyang Wang. 2016. Efficient maximum closeness centrality group identification. In *Databases Theory and Applications - 27th Australasian Database Conference, Proceedings (ADC '16)*. Springer, 43–55.

[14] Andrew Clark, Qiqiang Hou, Linda Bushnell, and Radha Poovendran. 2017. A submodular optimization approach to leader-follower consensus in networks with negative edges. In *Proceedings of American Control Conference (ACC '17)*. IEEE, 1346–1352.

[15] Andrew Clark and Radha Poovendran. 2011. A submodular optimization framework for leader selection in linear multi-agent systems. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC '11)*. IEEE, 3614–3621.

[16] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup Rao, and Shen Chen Xu. 2014. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th annual ACM Symposium on Theory of Computing (STOC '14)*. ACM, 343–352.

[17] Samuel I. Daitch and Daniel A. Spielman. 2008. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*. ACM, 451–460.

[18] Shlomi Dolev, Yuval Elovici, Rami Puzis, and Polina Zilberman. 2009. Incremental deployment of network monitors based on group betweenness centrality. *Inform. Process. Lett.* 109, 20 (2009), 1172–1176.

[19] Peter G Doyle and J Laurie Snell. 1984. *Random Walks and Electric Networks.* Mathematical Association of America.

[20] Paul Erdös and Alfréd Rényi. 1959. On random graphs, I. *Publ. Math. Debrecen* 6 (1959), 290–297.

[21] Martin G Everett and Stephen P Borgatti. 1999. The centrality of groups and classes. *J. Math. Sociol.* 23, 3 (1999), 181–201.

[22] Martin Fink and Joachim Spoerhase. 2011. Maximum betweenness centrality: approximability and tractable cases. In *Algorithms and Computation - 5th International Workshop, Proceedings (WALCOM '11)* . Springer, 9–20.

[23] Gerd Fricke, Stephen T. Hedetniemi, and David Pokrass Jacobs. 1998. Independence and Irredundance in k-Regular Graphs. *Ars Comb.* 49 (1998).

[24] Rumi Ghosh, Shang-hua Teng, Kristina Lerman, and Xiaoran Yan. 2014. The interplay between dynamics and networks: centrality, communities, and cheeger inequality. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, 1406–1415.

[25] Christos Gkantsidis, Milena Mihail, and Amin Saberi. 2006. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Evaluation* 63, 3 (2006), 241–263.

[26] MF Hutchinson. 1989. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Commun. Stat. Simul. Comput.* 18, 3 (1989), 1059–1076.

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

[28] Donald B. Johnson. 1977. Efficient algorithms for shortest paths in sparse networks. *J. ACM* (1977).

[29] William Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.* 26, 189-206 (1984), 1.

[30] Jonathan A. Kelner, Gary L. Miller, and Richard Peng. 2012. Faster approximate multicommodity flow using quadratically coupled flows. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012)*. ACM, 1–18.

[31] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. ACM, 137–146.

[32] Douglas J Klein and Milan Randić. 1993. Resistance distance. *J. Math. Chem.* 12, 1 (1993), 81–95.

[33] Ioannis Koutis, Alex Levin, and Richard Peng. 2016. Faster Spectral Sparsification and Numerical Algorithms for SDD Matrices. *ACM Trans. Algorithms* 12, 2 (2016), 17:1–17:16.

[34] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *22nd International World Wide Web Conference, Companion Volume (WWW '13)* . ACM, 1343–1350.

[35] Amy N Langville and Carl D Meyer. 2012. *Who's# 1?: the science of rating and ranking.* Princeton University Press.

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

[37] Rong-Hua Li, Jeffrey Xu Yu, Xin Huang, and Hong Cheng. 2014. Random-walk domination in large graphs. In *Proceedings of IEEE 30th International Conference on Data Engineering (ICDE '14)*. IEEE, 736–747.

[38] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. 2016. Vital nodes identification in complex networks. *Phys. Rep.* 650 (2016), 1–63.

[39] Ahmad Mahmoody, Charalampos E Tsourakakis, and Eli Upfal. 2016. Scalable betweenness centrality maximization via sampling. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 1765–1773.

[40] Charalampos Mavroforakis, Michael Mathioudakis, and Aristides Gionis. 2015. Absorbing random-walk centrality: Theory and algorithms. In *Proceedings of IEEE International Conference on Data Mining (ICDM '15)*. IEEE, 901–906.

[41] Gary L. Miller and Richard Peng. 2013. Approximate Maximum Flow on Separable Undirected Graphs. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '13)*. SIAM, 1151–1170.

[42] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.* 14, 1 (1978), 265–294.

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

[44] Mark E. J. Newman. 2010. *Networks: An Introduction.* Oxford University Press.

[45] Stacy Patterson and Bassam Bamieh. 2010. Leader selection for optimal network coherence. In *Proceedings of 49th IEEE Conference on Decision and Control (CDC '10)*. IEEE, 2692–2697.

[46] Mohammad Pirani and Shreyas Sundaram. 2014. Spectral properties of the grounded Laplacian matrix with applications to consensus in the presence of stubborn agents. In *Proceedings of American Control Conference (ACC '14)*. IEEE, 2160–2165.

[47] Mojtaba Rezvani, Weifa Liang, Wenzheng Xu, and Chengfei Liu. 2015. Identifying top-k structural hole spanners in large-scale social networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, 263–272.

[48] Daniel A. Spielman and Shanghua 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.

[49] Daniel A. Spielman and Nikhil Srivastava. 2011. Graph Sparsification by Effective Resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.

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

[51] Hanghang Tong, B Aditya Prakash, Charalampos Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. 2010. On the vulnerability of large graphs. In *Proceedings of IEEE 10th International Conference on Data Mining (ICDM '10)*. IEEE, 1091–1096.

[52] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of small-world networks. *Nature* 393, 6684 (1998), 440.

[53] Scott White and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. ACM, 266–275.

[54] Wenzheng Xu, Mojtaba Rezvani, Weifa Liang, Jeffrey Xu Yu, and Chengfei Liu. 2017. Efficient Algorithms for the Identification of Top-k Structural Hole Spanners in Large Social Networks. *IEEE Trans. Knowl. Data Eng.* 29, 5 (2017), 1017–1030.

[55] Yuichi Yoshida. 2014. Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, 1416–1425.

[56] Junzhou Zhao, John Lui, Don Towsley, and Xiaohong Guan. 2014. Measuring and maximizing group closeness centrality over disk-resident graphs. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. ACM, 689–694.

[57] Pengpeng Zhao, Yongkun Li, Hong Xie, Zhiyong Wu, Yinlong Xu, and John CS Lui. 2017. Measuring and Maximizing Influence via Random Walk in Social Activity Networks. In *Database Systems for Advanced Applications - 22nd International Conference, Proceedings, Part II (DASFAA '17)* . Springer, 323–338.