Resistance Eccentricity in Graphs: Distribution, Computation and Optimization

Zenan Lu
School of Computer Science
Fudan University
Shanghai, 200433, China
20110240073@fudan.edu.cn

Xiaotian Zhou
School of Computer Science
Fudan University
Shanghai, 200433, China
22110240080@m.fudan.edu.cn

Ahad N. Zehmakan
School of Computing
Australian National University
Canberra, Australia
ahadn.zehmakan@anu.edu.au

Zhongzhi Zhang
School of Computer Science
Fudan University
Shanghai, 200433, China
zhangzz@fudan.edu.cn

***Abstract*—We study resistance eccentricity, a fundamental metric in network science for measuring the structural significance of a node. For a node in a graph, the resistance eccentricity is its maximum resistance distance to all other nodes. Fast computation of resistance eccentricity for a given subset of nodes is essential for a wide range of applications. However, a naive computation, requiring the pseudoinverse of the graph Laplacian, takes cubic time and is thus infeasible for huge networks with millions of nodes. In this paper, we devise a near-linear time algorithm to approximate the resistance eccentricity for one or multiple given nodes, accompanied by a theoretically guaranteed error bound. Furthermore, we investigate the problem of minimizing the resistance eccentricity for a given node by adding k missing edges to the graph, for a budget k. We show that while the objective function is monotone, it does not possess the submodularity property, ruling out the classical hill-climbing algorithm with theoretical guarantees. Instead, we propose four fast heuristic algorithms to approximately solve this problem. Then, we conduct extensive experiments on different networks with sizes up to several million nodes, demonstrating the superiority of our algorithms in terms of efficiency and effectiveness.**

***Index Terms*—Resistance distance, resistance eccentricity, combinatorial optimization, graph mining, social networks.**

I. Introduction

Resistance distance stands out as a pivotal metric in graph theory, which has found applications in a vast spectrum of computer science areas. Within algorithmic graph theory, the concept of resistance distance has been a catalyst for pioneering solutions to foundational problems. This has led to significant advancements in fields such as spectral graph sparsification [1], flow approximation [2], random tree creation [3], tree optimization [4], and the travelling salesman problem [5]. On the more practical front, resistance distance has been instrumental in areas like recommendation systems [6], graph embeddings [7], image processing [8], identifying key nodes and connections in graphs [9], [10], [11], and predicting network links [12]. Given its wide-ranging influence and applicability, the exploration of resistance distance has been a focal point of research for many years [13], [14], [15].

We delve into studying the fundamental metric of resistance eccentricity. For a node in a network, resistance eccentricity is defined as its greatest resistance distance to any other node in the network. While various graph metrics, including well-known measures like closeness centrality [16], Kirchhoff index [17], [18], and information centrality [19], have been investigated extensively, recently, attention has shifted to the study of resistance eccentricity. This is mainly because resistance eccentricity offers insights distinct from traditional eccentricity, especially in contexts like disease propagation and complex network interactions, as highlighted in [20]. However, research on resistance eccentricity is still in its nascent stages, with many facets yet to be uncovered.

In contrast to traditional measures, our understanding of resistance eccentricity remains quite limited. For instance, while the eccentricity distribution has found applications in diverse fields such as routing networks [21], biological networks [22], hardware verification [23], and enhancing performance in social networks [24], there are no prior studies on the distribution of resistance eccentricity in different graphs. Furthermore, while there exists a plethora of approaches [25], [26], [27], [28] dedicated to computing/approximating the traditional, with some even efficiently handling graphs of billion-scale magnitudes, determining the resistance eccentricity in large graphs has remained unsolved. In particular, the direct computation of resistance eccentricity for each node is cubic in time complexity, which is computationally expensive for large graphs. Lastly, the problem of optimizing traditional eccentricity measures has been studied extensively, cf.[29], [30], [31], while there has been little progress on optimization of resistance eccentricity.

In this paper, we conduct a systematic study on the resistance eccentricity of a graph $\mathcal{G} = (V, E)$ with n nodes and m edges. The main contributions and work of this paper are summarized as follows.

- In our analysis of real-world networks, we found that the distribution of resistance eccentricity exhibits properties of asymmetry, rightward skewness, and a significant heavy tail. This aligns with the general understanding of real-world networks, where the majority of nodes are closely connected, and only a small subset of nodes is loosely connected.
- We propose a fast ϵ-approximation algorithm FAST-QUERY to query the resistance eccentricities of a set $Q \subseteq V$, which is based on the Johnson-Lindenstrauss Lemma,

Ahad N. Zehmakan is the corresponding author.
Zhongzhi Zhang is the corresponding author.

Laplacian solvers and convex hull. The algorithm has a nearly linear time complexity $\widetilde{O}\left((m+nl)/\epsilon^2+|Q|l\right)$, where the $\tilde{O}(\cdot)$ notation suppresses the $\mathrm{poly}(\log n)$ factors, $\epsilon > 0$ is the error parameter, and l is the number of nodes in the boundary of the approximated convex hull, which is small for most real-world networks.

- We investigate two fundamental optimization problems associated with the resistance eccentricity of nodes in a graph. Given a graph $\mathcal{G}$, a target node s, and an integer k with $k \ll n$, we consider the task of selecting a subset P of k edges to augment $\mathcal{G}$. In one scenario, we draw from the set $Q = \{(s,u)|u \in V$ and $(s,u) \in (V \times V)\backslash E\}$ while in the other, the selection is made from $Q = (V \times V)\backslash E$. In both cases, the objective is to minimize the resistance eccentricity of a specified node in the augmented graph. We then propose four efficient greedy heuristic algorithms FARMINRECC, CENMINRECC, CHMINRECC and MINRECC to approximately solve these combinatorial optimization problem.
- We execute extensive experiments on a large variety of real-world network data to evaluate the performance of proposed algorithms: FASTQUERY for querying resistance eccentricity, and FARMINRECC, CENMINRECC, CHMINRECC and MINRECC for minimizing resistance eccentricity of an given node by edge addition. Our experimental results show that our FastQuery algorithm and the four approximation optimization algorithms are both efficient and accurate. In particular, the approximation optimization algorithms provide near-optimal solutions for networks with over one million nodes.

Outline. The remainder of this paper is organized as follows. Section II provides an overview of related work. Preliminaries are covered in Section III. Section IV delves into the analysis of the properties of the resistance eccentricity distribution. Our fast querying algorithm for resistance eccentricity is detailed in Section V. Furthermore, in Section VI, we discuss the problem of minimizing resistance eccentricity through edge addition and introduce a foundational algorithm. Subsequent to that, Section VII presents fast algorithms specifically designed for this optimization problem. We conclude with Section VIII, where we share our experimental results and compare the performance of the proposed algorithms.

II. Related Work

In this section, we briefly review the literature related to our work.

The eccentricity of a node v in a graph signifies the maximum distance from v to any other node within that graph. When computed for all nodes, this metric forms the graph's eccentricity distribution, which stands as a pivotal characteristic of the graph's structure. Research by prior work [32] has highlighted that the eccentricity distribution in real-world graphs often follows a unimodal pattern, skewing towards a pronounced positive tail. A myriad of methodologies [25], [26], [27], [33] have been developed to compute this distribution, be it precisely or approximately. In recent advancements in the domain of graph eccentricities, the groundbreaking algorithm IFECC was unveiled. This algorithm not only outpaces its predecessors by two orders of magnitude, but also adeptly processes graphs on a billion-scale magnitude [28]. Additionally, optimization problems related to eccentricity have garnered significant attention in the research community [29], [30], [31]. For instance, strategies to reduce the eccentricity of a target node by adding edges to the graph have been explored [29]. Another intriguing area of study focuses on edge additions that aim to decrease the diameter of the graph (the maximum value of eccentricity among all nodes) [30]. The aforementioned studies on eccentricity are not directly applicable to resistance eccentricity. This necessitates a comprehensive and systematic analysis specifically tailored to resistance eccentricity.

In the analysis of resistance eccentricity for nodes within a graph, computing the resistance distance is a crucial and indispensable step. Fast query of resistance distances is the premise of various applications, and thus numerous algorithms have been developed in the literature to estimate resistance distances between node pairs. A random projection based method was proposed in [1], [34] for evaluating the resistance distances between the end nodes of every edge, which was accelerated in [35] by using Wilson's algorithm [36]. In [37], a local algorithm was developed to query pairwise resistance distance based on samplings of random walks and generation of random spanning trees. In [38], [39], Monte Carlo approaches were designed to improve the performance of the previous algorithm [37]. Although there are currently many efficient methods for calculating node-to-node resistance distance, querying the resistance eccentricities over all the nodes still requires calculating the resistance distances between quadratically many pairs of nodes, which is impractical for large networks.

In the context of optimizing resistance distances, a lot of existing works have focused on the problem of augmenting a graph by adding new edges, aiming to minimize either the resistance distance between a specific pair of nodes or the Kirchhoff index (the aggregate of resistance distances across all node pairs) in the resulting graph [17], [18]. The authors of [40] tackled the NP-hard problem of selecting up to k edges to minimize the resistance distance between a designated pair of nodes, offering a constant factor approximation algorithm for the solution. Concurrently, the Kirchhoff index's optimization problem has garnered significant attention owing to its broad applicability [41], [42], [43], [44]. Various prior works [45], [46], [47], [48], [49] have explored the problem of minimizing the Kirchhoff index by adding a fixed number of edges and have put forth diverse methodologies. However, these prior solutions [40], [45], [46], [47], [48], [49] are not directly applicable to the specific problem of minimizing the resistance eccentricity through the addition of edges.

Indeed, the operation of edge addition, motivated from link recommendation systems in online social networks, has been widely employed to obtain various objectives. This includes enhancing the centrality of a node [50], [51], [10], maximizing the quantity of spanning trees [52], and either maximizing the overall consensus in opinion dynamics [53] or reducing polarization and disagreement [54]. Recently, this edge addition technique has been applied to the problem of minimizing network diameter [31], [55]. Other similar problems involving the augmentation of graphs through edge additions have been explored, such as boosting algebraic connectivity [56]. Nevertheless, there has been no prior consideration of utilizing graph edit operations like edge addition to specifically minimize the resistance eccentricity of a given node.

III. Preliminaries

In this section, we provide a general overview of the notations, graphs definitions, related matrices, resistance distance and eccentricity.

A. Notations

We 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. Let $\boldsymbol{e}_i$ be the i^{th} standard basis vector of appropriate dimensions. Let $\boldsymbol{J}$ and $\mathbf{1}$ be, respectively, the matrix and vector of appropriate dimensions with all entries being ones. Let $\boldsymbol{a}^\top$ and $\boldsymbol{A}^\top$ denote, respectively, transpose of vector $\boldsymbol{a}$ and matrix $\boldsymbol{A}$. We use $\boldsymbol{A}_{[i,:]}$ and $\boldsymbol{A}_{[:,j]}$ to denote, respectively, the i^{th} row and the j^{th} column of matrix $\boldsymbol{A}$. We write $\boldsymbol{A}_{i,j}$ to denote the entry at row i and column j of $\boldsymbol{A}$. We use $\mathbb{R}^m$ to represent the m-dimensional Euclidean space, where each element is a real vector; and use $\mathbb{R}^{m\times n}$ to represent the set of all $m \times n$ real-valued matrices.

For a vector $\boldsymbol{a}$, its ℓ_2 norm $\|\boldsymbol{a}\|_2$ is defined as $\|\boldsymbol{a}\|_2 = \sqrt{\sum_i \boldsymbol{a}_i^2}$, its ℓ_0 norm $\|\boldsymbol{a}\|_0$ is defined as the number of nonzero elements in $\boldsymbol{a}$, and its norm with respect to a matrix $\boldsymbol{A}$ is definfed to be $||\boldsymbol{a}||_{\boldsymbol{A}} = \sqrt{\boldsymbol{a}^\top \boldsymbol{A}\boldsymbol{a}}$.

For two nonnegative scalars a and b, we use $a \overset{\epsilon}{\approx} b$ to denote that a is an ϵ-approximation of b obeying relation $(1-\epsilon)b \le a \le (1+\epsilon)b$.

B. Graph and Related Matrices

Let $\mathcal{G} = (V, E)$ denote a connected undirected unweighted graph with node set V and edge set E, whose numbers of nodes and edges are $n = |V|$ and $m = |E|$, respectively. For graph $\mathcal{G} = (V, E)$, an edge set $Q = (V \times V)\backslash E$ and an edge set $A \subset Q$, we use $\mathcal{G}(A) = (V, E \cup A)$ to denote the augmented graph of $\mathcal{G} = (V, E)$ with the same node set V as $\mathcal{G}$ but more edges than $\mathcal{G}$.

The n nodes in graph $\mathcal{G}$ are labeled by $1, 2, \ldots, n$, respectively. The adjacency relation between the n nodes is encoded in the adjacency matrix $\boldsymbol{A} = (a_{ij})_{n\times n}$ of graph $\mathcal{G}$, where $a_{ij} = 1$ if nodes i and j are directly connected by an edge in E, and $a_{ij} = 0$ otherwise. Thus, the degree of node i is $d_i = \sum_{j=1}^n a_{ij}$. Let $\boldsymbol{D}$ denote the diagonal degree matrix of $\mathcal{G}$. The i^{th} diagonal entry of $\boldsymbol{D}$ is d_i, while all other entries are zeros. Then, the Laplacian matrix $\boldsymbol{L}$ of $\mathcal{G}$ is defined to be $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

In graph $\mathcal{G}$, for every edge e with end nodes i and j, a direction is assigned arbitrarily. Then, we can define the edge-node incidence matrix $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ for graph $\mathcal{G}$. Let $\boldsymbol{b}_e^\top$ denote the row of matrix $\boldsymbol{B}$ corresponding to edge e. Then, the entry b_{eu} at the row associated with edge e and column corresponding to node u is: $b_{eu} = 1$ if node u is the tail of edge e, $b_{eu} = -1$ if node u is the head of edge e, and $b_{eu} = 0$, otherwise. Then, for an edge e linking two nodes i and j, $\boldsymbol{b}_e$ can also be represented as $\boldsymbol{b}_e = \boldsymbol{e}_i - \boldsymbol{e}_j$, and the Laplacian matrix $\boldsymbol{L}$ of $\mathcal{G}$ can be recast as $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{B} = \sum_{e\in E} \boldsymbol{b}_e \boldsymbol{b}_e^\top$, which indicates that $\boldsymbol{L}$ is symmetric and positive semidefinite.

Since matrix $\boldsymbol{L}$ is symmetric and positive semidefinite, all the eigenvalues of $\boldsymbol{L}$ are non-negative, with a unique zero eigenvalue. Let $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_n$ denote the n eigenvalues of matrix $\boldsymbol{L}$, and let $\boldsymbol{u}_k$, $k = 1, 2, \ldots, n$, denote their corresponding mutually orthogonal unit eigenvectors. The pseudoinverse of $\boldsymbol{L}$ is denoted by $\boldsymbol{L}^\dagger$ and calculated as $\boldsymbol{L}^\dagger = \left(\boldsymbol{L} + \frac{1}{n}\boldsymbol{J}\right)^{-1} - \frac{1}{n}\boldsymbol{J}$ [18], where $\boldsymbol{J}$ is an $n \times n$ matrix with all entries being 1.

Then, $\boldsymbol{L}$ has a spectral decomposition of the form $\boldsymbol{L} = \sum_{k=1}^n \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$. Since the Laplacian matrix $\boldsymbol{L}$ has a eigenvalue 0, it is singular and cannot be inverted. As a substitute for the inverse, we use the Moore-Penrose generalized inverse of $\boldsymbol{L}$, which we simply call pseudoinverse of $\boldsymbol{L}$ [57]. As customary, we use $\boldsymbol{L}^\dagger$ to denote the pseudoinverse of $\boldsymbol{L}$, which can be written as

$$\boldsymbol{L}^\dagger = \sum_{k=2}^n \frac{1}{\lambda_k} \boldsymbol{u}_k \boldsymbol{u}_k^\top.$$

C. Electrical Networks, Resistance Distance and Resistance Eccentricity

By replacing every edge in graph $\mathcal{G}$ with a unit resistance, we obtain an electrical network [58] associated with graph $\mathcal{G}$. We simply use $\mathcal{G}$ to denote the electrical network corresponding to graph $\mathcal{G}$, when it is clear from the context. For a graph $\mathcal{G} = (V, E)$, the resistance distance $r(u, v)$ between two nodes u and v is defined as the resistance distance between u and v in the corresponding electrical network [17], which can be expressed in terms of the entries of matrix $\boldsymbol{L}^\dagger$ as

$$r(u, v) = \boldsymbol{L}^\dagger_{uu} + \boldsymbol{L}^\dagger_{vv} - 2\boldsymbol{L}^\dagger_{uv}. \tag{1}$$

Based on resistance distance, graph metrics such as resistance eccentricity and resistance diameter have also been introduced.

Definition 3.1: [20] Given a node v of a graph $\mathcal{G} = (V, E)$, the resistance eccentricity of v is

$$c(v) = \max_{u\in V} r(v, u). \tag{2}$$

The farthest node from v is $f_v = \arg\max_{u\in V} r(v, u)$. If the farthest node of v is not unique, f_v can be any of the farthest nodes. The resistance eccentricity distribution $\mathcal{E}(\mathcal{G})$ of $\mathcal{G}$ is $\mathcal{E}(\mathcal{G}) = \{c(v) \mid v \in V\}$.

For a graph $\mathcal{G} = (V, E)$, the maximum value of resistance distances over all its node pairs is called the resistance diameter and is denoted by $R(\mathcal{G})$ [59]. That is,

$$R(\mathcal{G}) = \max_{u,v\in V} r(u, v). \tag{3}$$

In addition to these, we introduce a new metric: the resistance radius. This metric, like the traditional graph radius, represents the minimum resistance eccentricity among all nodes in the graph.

For a graph $\mathcal{G} = (V, E)$, the minimum value of resistance distances over all its node pairs is called the resistance radius and is denoted by $\phi(\mathcal{G})$. That is,

$$\phi(\mathcal{G}) = \min_{v \in V} c(v). \tag{4}$$

Similar to the concept of graph center based on eccentricity, we also define the resistance center of a graph based on resistance eccentricity. A node is resistance central if $c(u) = \phi(\mathcal{G})$ and the resistance center of $\mathcal{G}$ is the set of all resistance central nodes.

IV. The Distribution of Resistance Eccentricity

In this section, we focus on the distribution of resistance eccentricity based on simple graphs and analyze the properties of resistance eccentricity distribution in real-world networks.

A. *Resistance Eccentricity in Simple Graphs*

The use of resistance distance as a distance metric in graphs has recently gained attention, as it carries information essential for certain applications. This is due to the fact that as opposed to the shortest path distance, resistance distance takes into account all paths between u and v, not just the shortest one: the more paths connect u with v, the smaller their resistance distance becomes. For example, in small-world graphs, all nodes will be close to one another, which results in a very small variation in the distance between nodes, while the variation in the resistance distance could be much larger. See, for instance, the works in [60], [61], which use resistance distance/commute time as a distance measure in social network graphs.

From the resistance eccentricity distribution, the resistance radius ϕ and resistance diameter R can be driven in linear time, that is, $\phi = \min_{v \in V} c(v)$ and $R = \max_{v \in V} c(v)$. We initially investigate the resistance eccentricity of each node on some simple graphs. Figure 1 consists of three simple graphs: a line graph with $2n$ nodes and $(2n-1)$ edges, a cycle graph with $2n$ nodes and $2n$ edges, and a star graph with $2n$ nodes and $(2n-1)$ edges. For (a), if $1 \leq i \leq n$, $c(v_i) = 2n - i$; if $(n+1) \leq i \leq 2n$, $c(v_i) = i - 1$. The number of resistance central nodes is 2. For (b), the resistance eccentricity distribution is $c(v_1) = c(v_2) = \cdots = c(v_{2n}) = n/2$ and $\phi = R = n/2$. The number of resistance central nodes is $2n$. For (c), the resistance eccentricity distribution is $c(v_1) = 1$, $c(v_2) = c(v_3) = \cdots = c(v_{2n}) = 2$ and $\phi = 1, R = 2$. The number of resistance central node is 1.

B. *Resistance Eccentricity Distribution for Real-world Networks*

In this section, we analyze the resistance eccentricity distribution in real-world networks through extensive experiments. All the datasets used in our experiments are chosen

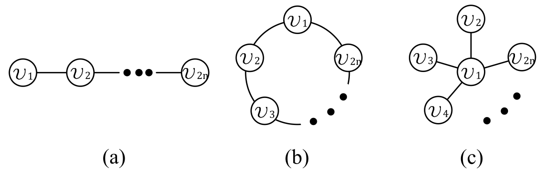

Fig. 1. The example graphs: line graph(a), cycle graph(b), star graph(c).

from Koblenz Network Collection [62] and Network Repository [63]. The collected networks are scale-free small-world, and are highly representative, spanning different fields such as social science, life science, and information science. Since we are only concerned with connected, undirected, unweighted simple graphs, we perform some preprocessings for the studied networks. For those directed or weighted networks, we convert directed or weighted edges to undirected and unweighted ones. And for each network, we only keep the largest connected component (LCC), deleting other small components and eliminating self-loops in the LCC. That is to say, we only study the resistance eccentricity distribution of the LCCs of obtained networks.

In Table I, we select four networks as examples for analysis. Figure 2 depicts the resistance eccentricity distribution across these four networks, complemented by the corresponding probability density function. In our analysis, we sought the fitting formula within MATLAB. Specifically, we employed the Burr distribution, also known as the Burr Type XII distribution or Singh-Maddala distribution. Its probability density function (pdf) is defined as:

$$f(x \mid c, k) = ckx^{-(c+1)} \left(1 + x^{-c}\right)^{-(k+1)}$$

where the parameters c and k are determined by MATLAB. From Figure 2, it is evident that the Burr distribution aptly models the resistance eccentricity. A distribution which can be an alternative for analyzing the right-skewed and heavy-tailed data is Burr distribution [64]. Upon examining the data, we notice that the resistance eccentricity distribution displays a distinct asymmetry. A significant portion of the nodes within the network register their resistance eccentricity values just above the resistance radius. As the value of resistance eccentricity increases, there is a substantial decline in the number of corresponding nodes. This culmination results in merely a handful of nodes achieving a resistance eccentricity that matches the resistance diameter. This dynamic contributes to the emergence of a significant heavy-tailed distribution. Therefore, the resistance eccentricity distribution can be characterized by its asymmetry, rightward skewness, and pronounced heavy tail. In practice, a large fraction of nodes clusters closely around the central node(s), leading to reduced resistance distances between them. On the other hand, a smaller set of nodes are positioned at the network's "fringes", possessing a larger resistance eccentricity.

From the last three columns of Table I, it can be observed that, under similar power-law exponents, the network's resis-

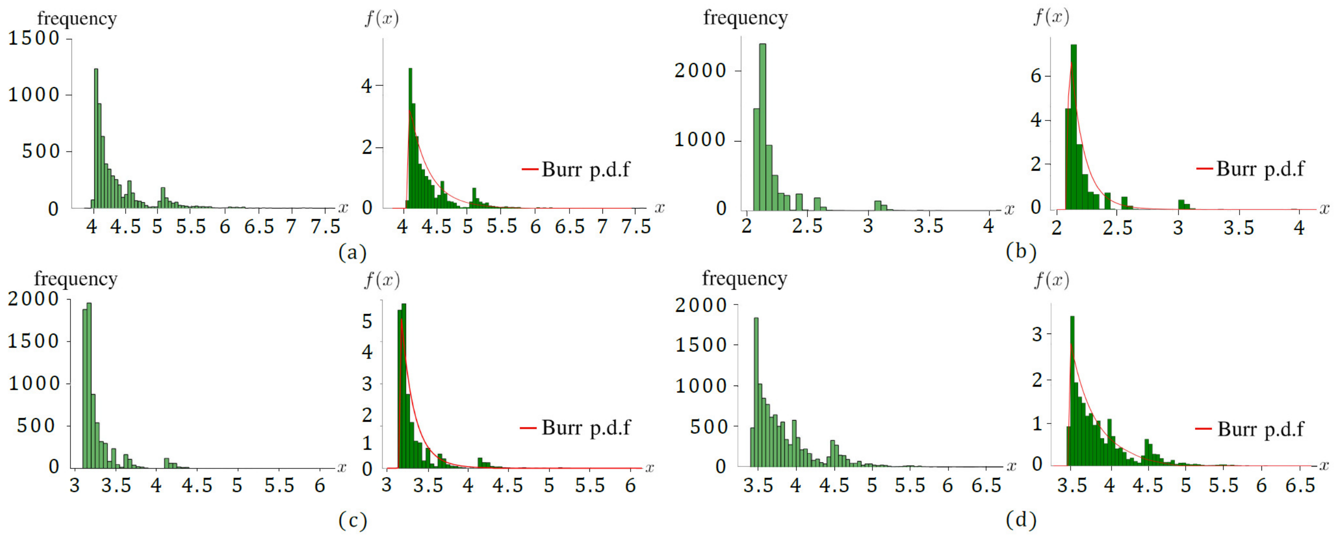

Fig. 2. Resistance eccentricity distribution and probability density function of four realistic scale-free networks: Politician (a), Musae-FR (b), Government (c) and HepPh (d).

tance radius and resistance diameter are close to each other and both have relatively small values. The resistance distance $r(i,j)$ between any two nodes, i and j, on the network is correlated with the degrees of these two nodes, that is, $r(i,j) \approx 1/d_i + 1/d_j$. In most real-world networks that exhibit properties of a power-law distribution, the majority of nodes have a similarly small degree and a high clustering coefficient. This leads to most nodes having resistance eccentricities that are close to the resistance radius. Meanwhile, a minority of nodes with higher degrees have the smallest resistance eccentricities, marking them as the resistance central points in the network. On the other hand, the few peripheral nodes at the network's outermost regions often lie on longer single chains, resulting in a larger resistance eccentricity.

TABLE I
STATISTICS OF SOME DATASETS AND THEIR RESISTANCE RADII AND DIAMETERS. FOR EACH NETWORK $\mathcal{G}$, WE INDICATE THE NUMBER OF NODES n, AND THE NUMBER OF EDGES m, THE AVERAGE DEGREE d_{AVG}, THE POWER-LAW EXPONENT γ, THE RESISTANCE RADIUS ϕ, AND THE RESISTANCE DIAMETER R FOR ITS LARGEST CONNECTED COMPONENT.

Network	n	m	d_{avg}	γ	ϕ	R
Politician	5908	41729	14.12	3.29	4.04	7.67
Musae-FR	6549	112666	34.41	2.64	2.07	4.13
Government	7057	89429	25.34	2.85	3.11	6.21
HepPh	11204	117619	21.00	2.09	3.42	6.75

V. QUERY ALGORITHMS FOR RESISTANCE ECCENTRICITY

In this section, we introduce three algorithms designed to query the resistance eccentricity of nodes within a graph. These algorithms provide efficient methodologies to compute this metric, offering insights into the structural characteristics of the network.

A. Exact Querying

To query the resistance eccentricity of any node $i \in V$, we can compute the resistance distance $r(i,j)$ for any $j \in V$ and find the maximum according to the definition of the resistance eccentricity. The core step is to compute the resistance distance which requires inverting the matrix $\boldsymbol{L}$. After computing the pseudoinverse $\boldsymbol{L}^{\dagger}$ in $O(n^3)$ time, we can compute the resistance distance $r(i,j)$ for each node $j \in V$ in $O(1)$ time; thus, assuming that we have computed $\boldsymbol{L}^{\dagger}$, the resistance eccentricity $c(i) = \max_{j\in V} r(i,j)$ can be computed in $O(n)$ for any node i.

According to above analysis, we can design an exact algorithm for computing the resistance eccentricity of any node. This algorithm consists of two steps. The first one is the preprocessing step, in which we precompute the pseudoinverse $\boldsymbol{L}^{\dagger}$ in $O(n^3)$. The second one is the querying step, where we compute the resistance eccentricity of any node in $O(n)$ time. If we want to query the resistance eccentricities of a set $Q \subseteq V$, we can call the query step $|Q|$ times. So, we can consequently propose an exact algorithm called EXACTQUERY, outlined in Algorithm 1. This algorithm takes $O(n^3 + |Q|n)$ time for answering queries from node set Q. Moreover, if we want to compute the resistance eccentricity distribution $\mathcal{E}(\mathcal{G})$ of graph $\mathcal{G}$, the time complexity of this algorithm becomes $O(n^3 + n^2) = O(n^3)$.

B. Approximate Querying

As highlighted, the matrix inversion step in Algorithm EXACTQUERY is the most time-intensive, taking $O(n^3)$ time. To bypass this costly operation, we introduce a randomized algorithm that approximates the resistance eccentricity of a target node in $\widetilde{O}(m\epsilon^{-2})$ time, where ϵ is an error factor.

```
Algorithm 1: ExactQuery($\mathcal{G}, Q$)
  Input  : A connected graph $\mathcal{G} = (V, E)$ with
           Laplacian matrix $\boldsymbol{L}$, a query set $Q \subseteq V$
  Output : $\{i, c(i) | i \in Q\}$
1 Compute the pseudoinverse $\boldsymbol{L}^\dagger$ of $\boldsymbol{L}$ by
    $\boldsymbol{L}^\dagger = \left(\boldsymbol{L} + \frac{1}{n}\boldsymbol{J}\right)^{-1} - \frac{1}{n}\boldsymbol{J}$
2 for $i \in Q$ do
3     $c(i) \leftarrow \max_{j \in V} r(i,j) = \boldsymbol{L}^\dagger_{i,i} + \boldsymbol{L}^\dagger_{j,j} - 2\boldsymbol{L}^\dagger_{i,j}$
4 return $\{i, c(i) | i \in Q\}$
```

```
Algorithm 2: ApproxQuery($\mathcal{G}, Q, \epsilon$)
  Input  : A connected graph $\mathcal{G} = (V, E)$, a query set
           $Q \subseteq V$, a parameter $\epsilon$
  Output : $\{i, \bar{c}(i) | i \in Q\}$
1 $d = \lceil 24 \log n / \epsilon^2 \rceil$
2 $\tilde{\boldsymbol{X}}_{d \times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
3 for $i \in Q$ do
4     Compute $\tilde{r}(i,j)$ for node $i$ and all other nodes in
        $\mathcal{G}$ by $\tilde{r}(i,j) = ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_i - \boldsymbol{e}_j)||_2^2$
5     $\bar{c}(i) \leftarrow \max_{j \in V/i} \tilde{r}(i,j)$
6 return $\{i, \bar{c}(i) | i \in Q\}$
```

Before introducing our algorithm, we first recast the resistance distance $r(u, v)$ as [1]

$$r(u,v) = \left\|\boldsymbol{B}\boldsymbol{L}^\dagger\left(\boldsymbol{e}_u - \boldsymbol{e}_v\right)\right\|_2^2 . \tag{5}$$

In other words, we can embed graph $\mathcal{G}$ to n points corresponding to n m-dimensional vectors $\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{e}_i$, $i = 1, 2, \dots, n$, in $\mathbb{R}^m$, which preserve the resistance distance. For two vectors $\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{e}_u$ and $\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{e}_v$ in $\mathbb{R}^m$, let $d(u, v)$ denotes their Euclidean distance. Then, we have $r(u, v) = d(u, v)^2$.

Equation (5) reduces the computation of resistance distance $r(u, v)$ to the calculation of the ℓ_2 norms $\left\|\boldsymbol{B}\boldsymbol{L}^\dagger\left(\boldsymbol{e}_u - \boldsymbol{e}_v\right)\right\|_2^2$ of two vectors in $\mathbb{R}^m$. However, the complexity for exactly computing this ℓ_2 norms is still high, since the dimension m of vectors $\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{e}_i$ ($i = 1, 2, \dots, n$) is high and it still needs inverting matrix $\boldsymbol{L} + \frac{1}{n}\boldsymbol{J}$ to obtain $\boldsymbol{L}^\dagger$.

To refine the dimensionality of vectors, we utilize the Johnson-Lindenstrauss (JL) Lemma [65], [66], [67] to approximate the ℓ_2 norms. For $\left\|\boldsymbol{B}\boldsymbol{L}^\dagger\left(\boldsymbol{e}_u - \boldsymbol{e}_v\right)\right\|_2^2$, when projecting the set of n m-dimensional vectors (namely the n column vectors of matrix $\boldsymbol{B}\boldsymbol{L}^\dagger$) onto a reduced d-dimensional subspace defined by the columns of a random matrix $\boldsymbol{Q} \in \mathbb{R}^{d \times m}$ with entries being $\pm 1/\sqrt{d}$ and where $d = \lceil 24 \log(n)/\epsilon^2 \rceil$ for a specified ϵ, we obtain an ϵ-approximation of $\left\|\boldsymbol{B}\boldsymbol{L}^\dagger\left(\boldsymbol{e}_u - \boldsymbol{e}_v\right)\right\|_2^2$ with a high probability. Moreover, to circumvent the matrix inversion, we can employ the fast symmetric, diagonally-dominant (SDD) linear system solver [68], [69], [70], [71], [72]. Given that $\boldsymbol{L}$ is an SDDM matrix, this facilitates the efficient computation of $\boldsymbol{Q}\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{e}_i$.

Based on the Laplacian solvers and the JL Lemma, an approximation algorithm ApproxER was proposed in [1] to estimate resistance distances in nearly linear time with respect to the number of edge, as stated in Lemma 5.1.

Lemma 5.1: There is an $\widetilde{O}(m\epsilon^{-2})$-time algorithm which on input $\epsilon > 0$ and $\mathcal{G} = (V, E)$ computes a $\lceil 24 \log n/\epsilon^2 \rceil \times n$ matrix $\tilde{\boldsymbol{X}}$ such that with probability at least $1 - 1/n$,

$$r(u,v) \overset{\epsilon}{\approx} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_u - \boldsymbol{e}_v)||_2^2$$

for every pair of nodes $u, v \in V$.

Utilizing Lemma 5.1, we are equipped to derive a query matrix $\tilde{\boldsymbol{X}}$ with a time complexity of $\widetilde{O}(m\epsilon^{-2})$. This, in turn, facilitates the querying of the resistance eccentricity for any node within the graph, achieving this in a time span of $O(n \log n\epsilon^{-2})$, all while maintaining an accuracy margin of ϵ. Leveraging our approach, if there's a need to query the resistance eccentricities of a subset $Q \subseteq V$, the query step can be executed $|Q|$ times. This leads us to introduce an approximation algorithm, called ApproxQuery, detailed in Algorithm 2. For addressing queries pertaining to the node set Q, this algorithm operates within a time frame of $\widetilde{O}\left((m + |Q|n)\epsilon^{-2}\right)$. Furthermore, to compute the resistance eccentricity distribution $\mathcal{E}(\mathcal{G})$ of the graph $\mathcal{G}$, the algorithm's time complexity is in $\widetilde{O}(m\epsilon^{-2} + n^2)$.

C. Fast Querying

In this subsection, we highlight that it's unnecessary to compute the resistance distances for all node pairs, especially when our primary interest lies in determining the maximum resistance distance for each node in graph $\mathcal{G}$. Consequently, we introduce a fast algorithm that approximates values of $c(i)$ in a given graph $\mathcal{G}$. This algorithm is characterized by its nearly-linear time and space complexity relative to the number of edges. Moreover, it offers a theoretically guaranteed error margin with a high probability of accuracy.

The ApproxER algorithm projects each node onto a d-dimensional Euclidean space. To query the resistance eccentricity of a single node, one must identify the furthest point from the given node in the Euclidean space $\mathbb{R}^d$, necessitating $(n - 1)$ computations. To optimize this, we employ convex hull techniques, aiming to reduce the number of distance computations required during each query.

Definition 5.2: [73] Given a set of n points $S = \{v_1, v_2, \dots, v_n\}$ in $\mathbb{R}^d$, its convex hull is the (unique) minimal convex polytope containing S.

For a given convex hull of point set S, its boundary is denoted by $C(S)$, and subset of points in S, which lie on the boundary of the convex hull is denoted by $\bar{S}$. By Definition 5.2, it can be observed that in an Euclidean space, for a given node set, the point farthest from any given point must belong to the set of points lying on the boundary of convex hull corresponding to that node set.

There are many methods to find the convex hull of a point set of size n in a low-dimensional Euclidean space. However, this is a computationally expensive task for high dimensions. For dimensions $d > 3$, the time for computing the convex hull is in $O(n^{\lfloor d/2 \rfloor})$, matching the worst-case output complexity of the problem [74], [75]. On the positive side, there is a fast algorithm APPROXCH which approximates the convex hull [76]. Before introducing this algorithm, we provide some more definitions. For any given set $S = \{v_i \in \mathbb{R}^d : i = 1, 2, \dots, n\}$, let $D(S)$ denote the diameter of S. That is, $D(S) = \max_{v_i, v_j \in S} ||v_i - v_j||_2$, which is the maximum distance between all pairs of points v_i and v_j, obeying relation $D(S) = D(\bar{S})$.

Lemma 5.3: [76] There is an algorithm APPROXCH(S, θ) which takes a set $S = \{v_i \in \mathbb{R}^d : i = 1, 2, \dots, n\}$ and an error parameter $\theta \in (0, 1)$, and returns an l-node subset $\hat{S}$ of $\bar{S}$. The algorithm runs in $O(nl(d + \theta^{-2}))$ time and the Euclidean distance for any $p \in \bar{S}$ to $C(\hat{S})$ is at most $\theta D(S)$, where $C(\hat{S})$ is the boundary of convex hull of $\hat{S}$.

Lemma 5.3 shows that using the APPROXCH algorithm, we can obtain an approximate point set $\hat{S}$ for $\bar{S}$, the boundary of convex hull $C(S)$ in a d-dimensional Euclidean space. Moreover, APPROXCH provides an upper bound for the distance between any point in $\bar{S}$ and any point in $C(\hat{S})$. Based on Lemma 5.3, we can derive an approximation for the farthest distance from a point $s \in S$ to the points in set $\bar{S}$ and the farthest distance from the same point s to the points in set $\hat{S}$.

Lemma 5.4: Consider a point set $S = \{v_i \in \mathbb{R}^d : i = 1, 2, \dots, n\}$, $s \in S$, a parameter $\theta = \frac{\epsilon}{12}$, a subset $\bar{S} \subseteq S$ of point whose points lie on the boundary of the convex hull $C(S)$ of set S, and $\hat{S} = $ APPROXCH(S, θ) that is an l-node subset of $\bar{S}$. Let u be the node in set $\hat{S}$ that is farthest from s, let v be the node in set $\bar{S}$ that is farthest from s. Then, we have

$$d(s, u) \overset{\frac{\epsilon}{6}}{\approx} d(s, v). \tag{6}$$

Proof. Let v' be the node that is closest to v and resides on the boundary of the convex hull $C(\hat{S})$. By Lemma 5.3, we have $d(s, v) - d(s, v') \le \theta D(S)$. Therefore, it can be concluded that $d(s, v) \le d(s, v') + \theta D(S) \le d(s, u) + \theta D(S)$. Then, we have $0 \le d(s, v) - d(s, u) \le \theta D(S)$. Let $D(S) = d(s, v) \cdot \frac{D(S)}{d(s,v)}$ and $1 \le \frac{D(S)}{d(s,u)} \le 2$, we then have $(1 - \frac{\epsilon}{6}) d(s, v) \le d(s, u)$. Moreover, since $\hat{S} \subseteq \bar{S}$, we can obtain $d(s, u) - d(s, v) \le 0$, which implies $\frac{d(s,u) - d(s,v)}{d(s,v)} \le \frac{\epsilon}{6}$. Finally, we have $(1 - \frac{\epsilon}{6}) d(s, v) \le d(s, u) \le (1 + \frac{\epsilon}{6}) d(s, v)$ which leads to Equation (6). □

Making use of Lemma 5.1 and Lemma 5.4, we obtain the following result.

Lemma 5.5: Consider a graph $\mathcal{G} = (V, E)$, a source node $s \in V$, a set $S = \{v_i \in \mathbb{R}^m : i = 1, 2, \dots, n\}$, where $v_i = \boldsymbol{QBL}^\dagger \boldsymbol{e}_i$, a parameter $\epsilon > 0$ and $\theta = \frac{\epsilon}{12}$. Suppose that $\hat{S} = $ APPROXCH(S, θ) is an l-point subset of $\bar{S}$. Let $\hat{c}(s) = \max_{u \in \hat{S}} d(s, u)^2$. Then, we have

$$\hat{c}(s) \overset{\epsilon/3}{\approx} \bar{c}(s). \tag{7}$$

Proof. Let $\hat{h}(s) = \max_{u \in \hat{S}} d(s, u)$ and $\bar{h}(s) = \max_{u \in \bar{S}} d(s, u)$. By Lemma 5.4, we have $\bar{h}(s) \ge \hat{h}(s) \ge (1 - \frac{\epsilon}{6}) \bar{h}(s)$. Then we obtain

$$\begin{aligned} \bar{c}(s)^2 \ge \hat{c}(s)^2 &\ge (1 - \frac{\epsilon}{6})^2 \bar{c}(s)^2 \\ &= (1 - \frac{\epsilon}{3} + \frac{\epsilon^2}{36}) \bar{c}(s)^2 \\ &\ge (1 - \frac{\epsilon}{3}) \bar{c}(s)^2. \end{aligned}$$

On the other hand, we have $\hat{c}(s) = \max_{u \in \hat{S}} d(s, u)^2 = \hat{h}(s)^2$ and $\bar{c}(s) = \max_{u \in \bar{S}} d(s, u)^2 = \bar{h}(s)^2$. Combining the above-obtained results, we have $\bar{c}(s) \ge \hat{c}(s) \ge (1 - \frac{\epsilon}{3}) \bar{c}(s)$ which leads to Equation (7). □

Algorithm 3: FASTQUERY$(\mathcal{G}, Q, \epsilon)$

```
Input  : A connected graph $\mathcal{G} = (V, E)$, a query set
         $Q \subseteq V$, a parameter $\epsilon$
Output : $\{i, \bar{c}(i) | i \in Q\}$
1 $d = \lceil 24 \log n / \epsilon^2 \rceil$, $\theta = \frac{\epsilon}{12}$
2 $\tilde{\boldsymbol{X}}_{d \times n} \leftarrow$ APPROXER$(\mathcal{G}, \epsilon)$
3 $S \leftarrow \{s_i \in \mathbb{R}^d | s_i = \tilde{\boldsymbol{X}}_{[:,i]}, i = 1, 2, \dots, n\}$
4 $\hat{S} \leftarrow$ APPROXCH$(S, \theta)$
5 for $i \in Q$ do
6     Compute $\tilde{r}(i, j)$ for node $i$ and all nodes in $\hat{S}$ by
        $\tilde{r}(i, j) = ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_i - \boldsymbol{e}_j)||_2^2$
7     $\hat{c}(i) = \max_{j \in \hat{S}} \tilde{r}(i, j)$
8 return $\{i, \hat{c}(i) | i \in Q\}$
```

Since $\hat{S}$ contains l points, we can obtain resistance eccentricity distribution by computing the distances between l points and each points in $\mathcal{G}$, instead of quadratically many pairs of points. In most real-world cases, l is much smaller than n; thus, using Lemma 5.5 to approximate the resistance eccentricity distribution of a graph, significantly reduces the time complexity.

Building on the preceding findings, we are ready to introduce a fast algorithm, FASTQUERY, designed to query the resistance eccentricity of a subset $Q \subseteq V$. The detailed pseudocode for this approach is delineated in Algorithm 3. In FASTQUERY, $\tilde{\boldsymbol{X}}$ is a $\lceil 24 \log n / \epsilon^2 \rceil \times n$ matrix, S is a set of points, which are the column vectors of matrix $\tilde{\boldsymbol{X}}$. Set $\hat{S}$ is an approximation of $\bar{S}$, which is the set of points lie on the boundary of the convex hull $C(S)$. The performance of the algorithm FASTQUERY is analyzed in Theorem 5.6.

Theorem 5.6: The FASTQUERY algorithm, when applied to graph $\mathcal{G}$ with a precision parameter ϵ, operates within a time complexity of $\tilde{O}\left((m + nl)/\epsilon^2 + |Q| l\right)$. The output of this algorithm is a matrix $\tilde{\boldsymbol{X}}_{d \times n}$ and a node set $\hat{S}$. To approximate the resistance eccentricity of a target node t, denoted as $\hat{c}(t)$, we can utilize the formula $\hat{c}(t) = \max_{u \in \hat{S}} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_t - \boldsymbol{e}_u)||_2^2$. This approximation ensures that:

$$(1 - \epsilon) c(t) \le \hat{c}(t) \le (1 + \epsilon) c(t).$$

Proof. We first analyze the time complexity of algorithm FASTQUERY$(\mathcal{G}, \epsilon)$, which includes two main operations: constructing matrix $\tilde{\boldsymbol{X}}$ and determining set $\hat{S}$. The construction of $\tilde{\boldsymbol{X}}$ takes $\widetilde{O}\left(m/\epsilon^2\right)$ time, while finding the set $\hat{S}$ of points needs $\widetilde{O}(nl/\epsilon^2)$ time. Lastly, we execute the query operation on the nodes within the set Q, which necessitates a time complexity of $O(|Q|l \log n\epsilon^{-2})$. Thus, the total running time of algorithm FASTQUERY is $\widetilde{O}\left((m+nl)\epsilon^{-2} + |Q|l\epsilon^{-2}\right)$.

We next prove the correctness of the approximation ratio. We first consider the error in the process of using APPROXER to convert the resistance distance into the low-dimensional Euclidean distance. Let $\tilde{r}(u,v)$ be the resistance distance obtained from the APPROXER algorithm, and for $t \in V$, let $\bar{c}(t)$ be the resistance eccentricity calculated using $\tilde{r}(u,v)$. We have $\bar{c}(t) \overset{\epsilon}{\approx} c(t)$.

Then, we use APPROXCH to find the nodes lie on the boundary of the convex hull, and let the resistance eccentricity of t as $\hat{c}(t)$. So, $\hat{c}(t) \overset{\epsilon/3}{\approx} \bar{c}(t)$.

Thus, we have $(1-\epsilon)c(t) \le \hat{c}(t) \le (1+\epsilon)c(t)$ which completes the proof. □

Additionally, when determining the approximation resistance eccentricity distribution $\hat{\mathcal{E}}(\mathcal{G})$ of the graph $\mathcal{G}$, the time complexity of the FASTQUERY algorithm extends to $\widetilde{O}\left((m+nl)/\epsilon^2 + nl\epsilon^{-2}\right)$.

VI. Minimizing Resistance Eccentricity of a Given Source Node

In this section, we formulate and study the problem of adding a fixed number edges in order to minimize the resistance eccentricity of a given source node.

A. Problem Statement, Optimal Solution, and Simple Greedy Algorithm

It is known that adding an edge to a graph does not increase the resistance distance between any pair of nodes [77]. Consequently, introducing any edge to a graph will not elevate the resistance eccentricity of any node. This underscores the potential of strategically adding edges to reduce the resistance eccentricity of specific nodes. Such targeted enhancements not only bolster the protection of "key" nodes but also elevate the overall connectivity and resilience of the network, see [20] for more details. This motivates our exploration into selecting a fixed number k of missing edges to minimize the resistance eccentricity of a given source node s.

Directly adding or removing edges to a single node is a simple and effective optimization approach. This method allows for targeted adjustments to the connectivity of the node within the network, which can significantly influence its centrality and other relevant metrics. This direct manipulation of a node's connectivity offers a versatile and practical way to fine-tune its position and role within the network, making it a commonly employed technique in various fields like social network analysis, transportation networks, and computational biology. For the resistance eccentricity metric, we also adopt the approach of directly adding edges to the source node s to minimize its resistance eccentricity.

Problem 1: (Resistance Eccentricity Minimization through Direct Edge Addition to Source Node (REMD)) Given a connected undirected graph $\mathcal{G} = (V,E)$, a given source $s \in V$, a candidate edge set $Q_1 = \{(s,u) | u \in V \text{ and } (s,u) \in (V \times V)\backslash E\}$ and an integer $1 \le k \le |Q_1|$, we aim to find an edge subset $P^* \subseteq Q_1$ with $|P^*| = k$, and add these chosen k edges to graph $\mathcal{G}$ forming a new graph $\mathcal{G}(P^*) = (V, E \cup P^*)$, so that the resistance eccentricity of s is minimized. Let $f_s(\mathcal{G}(P))$ be the $c(s)$ in the graph $\mathcal{G}(P)$, then, this set optimization problem can be formulated as:

$$P^* = \underset{P \subseteq Q_1, |P|=k}{\arg\min} \ f_s(\mathcal{G}(P)).$$

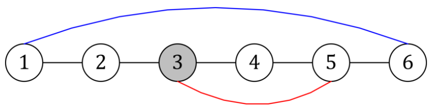

Fig. 3. A line graph with 6 nodes and 5 edges. The colored lines represent newly added edges.

In Problem 1, we solely focused on the approach of directly adding edges to the source node s. Nevertheless, it is worth to emphasize lifting this constraint and permitting the edges to be added anywhere in the graph can result in more optimal solutions. In order to demonstrate this point, we give an example of a graph $\mathcal{G}$ with 6 nodes and 5 edges in Figure 3. Let the source node s be node 3. We consider the case of adding only one edge. By directly adding an edge adjacent to the source node, we add edge $(3,5)$ to minimize the resistance eccentricity of node 3, resulting in $c(3) = 2$. However, by adding edge $(1,6)$, we find that $c(3) = 1.5$. To this point into account, we introduce Problem 2.

Problem 2: (Resistance Eccentricity Minimization (REM)) Given a connected undirected graph $\mathcal{G} = (V,E)$, a given source $s \in V$, a candidate edge set $Q_2 = (V \times V)\backslash E$ and an integer $1 \le k \le |Q_2|$, we aim to find the edge set $P^* \subseteq Q_2$ with $|P^*| = k$, and add these chosen P^* edges to graph $\mathcal{G}$ forming a new graph $\mathcal{G}(P^*) = (V, E \cup P^*)$, so that the resistance eccentricity of s is minimized. This set optimization problem can be formulated as:

$$P^* = \underset{P \subseteq Q_2, |P|=k}{\arg\min} \ f_s(\mathcal{G}(P)).$$

Problem 1 and Problem 2 are combinatorial optimization problems. We can naturally think of a direct solution method by exhausting all $\binom{|Q|}{k}$ cases, for $Q = Q_1$ and $Q = Q_2$. For each subset, we calculate the resistance eccentricity of s of the resultant graph, which requires $O(n^3)$ time. Then, output the optimal solution, which minimizes the resistance eccentricity of node s. Although this method is simple, it is computationally very expensive since the run time of $O\left(\binom{|Q|}{k} \cdot n^3\right)$ grows exponentially in k and in most real-world networks $|Q|$ is of quadratic size in n.

To tackle the undesirable complexity of brute-force search, it is conventional to resort to greedy approaches. We present a simple greedy algorithm for Problem 1 and Problem 2, which is outlined in Algorithm 4 and described as follows. Initially,

Algorithm 4: SIMPLE$(\mathcal{G}, Q, k, s)$

```
Input  : A connected graph $\mathcal{G} = (V, E)$, a
         candidate edge set $Q$, an integer
         $1 \leq k \leq |Q|$, a source node $s$
Output : $P$ : A subset of $Q$ with $|P| = k$ edges
1 Initialize solution $P = \emptyset$
2 for $i = 1$ to $k$ do
3     Select $e_i$ s.t. $e_i \leftarrow \arg\min_{e \in Q \setminus P} \{c(s) \mid \mathcal{G}(\{e\})\}$
4     Update solution $P \leftarrow P \cup \{e_i\}$
5     Update the graph $\mathcal{G} \leftarrow \mathcal{G}(\{e_i\})$
6 return $P$
```

we set the set P of added edges to be empty, then k edges are added from set $Q \backslash P$ iteratively. In each iteration i, edge e_i in set $Q \backslash P$ of candidate edges is selected, which minimize the $c(s)$ of the new graph. The algorithm terminates when k edges are selected to be added to P. For every candidate edge, it needs computation of $c(s)$. A direct calculation of $c(s)$ requires $O(n^3)$ time, leading to a total computation complexity of $O(k|Q|n^3)$. For Algorithm 4, when separately addressing Problem 1 and Problem 2, we only need to input different candidate edge sets Q, where the candidate edge set for Problem 1 is denoted as Q_1, and the candidate edge set for Problem 2 is denoted as Q_2. For Problem 1 and Problem 2, we call Algorithm 4 as SIM-REMD and SIM-REM, respectively.

B. Non-supermodular Property of the Objective Function

Two typical concepts related to an optimization problem with a cardinality constraint are monotone non-decreasing and supermodular set functions.

Definition 6.1: (Monotonicity) A set function $f : 2^Q \to \mathbb{R}$ is monotone non-decreasing if $f(S) \leq f(T)$ holds for all $S \subseteq T \subseteq Q$.

Definition 6.2: (Supermodularity) A set function $f : 2^Q \to \mathbb{R}$ is supermodular if

$$f(S) - f(S \cup \{e\}) \geq f(T) - f(T \cup \{e\})$$

holds for all $S \subseteq T \subseteq V$ and $e \in Q$.

For a combinatorial optimization problem, finding appropriate properties of its objective function is crucial to solving it effectively. For example, when the objective function is supermodular, a simple greedy algorithm by selecting one element with maximal marginal benefit in each iteration yields a solution with $(1-e^{-1})$ approximation ratio [78], which has been widely used in combinatorial optimization problems and has effectively solved many NP-hard problems in recent years.

However, there are still a lot of combinatorial optimization problems whose objective functions are not supermodular. For these problems, the greedy approach cannot guarantee a $(1 - e^{-1})$ approximation solution. Both Problem 1 and Problem 2 happen to be in this problem class. To show the non-supermodularity of the objective function for Problem 1, we give an example of the graph $\mathcal{G}$ with 6 nodes and 5 edges in Figure 4. Let set $A = \{(1, 6)\}$, $B = \{(1, 3), (1, 6)\}$

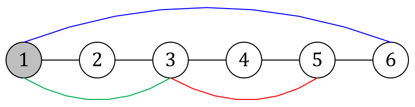

Fig. 4. A line graph with 6 nodes and 5 edges. The colored lines represent newly added edges.

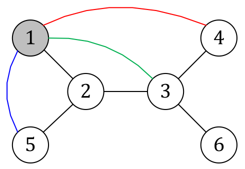

Fig. 5. A graph with 6 nodes and 5 edges. The colored lines represent newly added edges.

and $e = (3, 5)$. For ease of representation, we denote the resistance eccentricity of node 1 after adding set A as $c_A(1)$; the eccentricity of node 1 after adding set A and edge e as $c_{A'}(1)$; the eccentricity of node 1 after adding set B as $c_B(1)$; and the eccentricity of node 1 after adding set B and edge e as $c_{B'}(1)$. Simple computation leads to

$$c_A(1) = 1.5, \qquad c_{A'}(1) = 1.5,$$
$$c_B(1) = 1.14, \qquad c_{B'}(1) = 1.03,$$

which implies that

$$c_A(1) - c_{A'}(1) = 0 < 0.11 = c_B(1) - c_{B'}(1).$$

Therefore, the objective function of Problem 1 is non-supermodular.

To show the non-supermodularity of the objective function for Problem 2, we give an example of the graph $\mathcal{G}$ with 6 nodes and 5 edges in Figure 5. Let set $A = \{(1, 3)\}$, $B = \{(1, 3), (1, 4)\}$ and $e = (1, 5)$. Simple computation leads to

$$c_A(1) = 1.667, \qquad c_{A'}(1) = 1.625,$$
$$c_B(1) = 1.625, \qquad c_{B'}(1) = 1.476,$$

which means

$$c_A(1) - c_{A'}(1) = 0.042 < 0.149 = c_B(1) - c_{B'}(1).$$

Therefore, the objective function of Problem 2 is non-supermodular.

VII. FAST APPROXIMATION ALGORITHM

Although the computational complexity of Algorithm 4 is much lower than the brute-force algorithm, it still cannot handle large-scale networks since calculating $c(s)$ for each edge $e \in Q \backslash P$ takes cubic time in each iteration.

To reduce the computational complexity, we propose nearly linear time approximation algorithms for both problems.

```
Algorithm 5: FarMinrecc($\mathcal{G}, Q, k, \epsilon, s$)
Input  : A connected graph $\mathcal{G} = (V, E)$, a candidate
         edge set $Q$, an integer $1 \le k \le |Q|$, a
         parameter $\epsilon$, a source node $s$
Output : $P$ : A subset of $Q$ with $|P| = k$ edges
1 Initialize solution $P = \emptyset$
2 for $i = 1$ to $k$ do
3     $d = \lceil 24 \log n/\epsilon^2 \rceil$
4     $\tilde{\boldsymbol{X}}_{d\times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
5     Select
      $u_i \leftarrow \arg\max_{u\in V,(s,u)\in Q\setminus P} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_s - \boldsymbol{e}_u)||_2^2$
6     Update solution $P \leftarrow P \cup \{(s, u_i)\}$
7     Update the graph $\mathcal{G} \leftarrow \mathcal{G}(\{(s, u_i)\})$
8 return $P$
```

A. Fast Approximation Algorithm of REMD

For Problem 1, a natural approach is to directly connect the source node s with the node farthest from it in terms of resistance distance to reduce its resistance eccentricity. First, using ApproxER, we can convert the resistance distances between the nodes of the graph into low-dimensional Euclidean distances, where the coordinate of each node is represented by $\tilde{\boldsymbol{X}}\boldsymbol{e}_u$, where $\tilde{\boldsymbol{X}}$ is a $\left(24\log n/\epsilon^2\right) \times n$ matrix. According to farthest-first traversal technique, we need to calculate the Euclidean distance between node s and the other $(n-1)$ nodes, which takes $O(n\log n/\epsilon^2)$ time. By repeating the above two steps k times, we can obtain an edge set P containing k edges. The fast approximation algorithm FarMinrecc is presented in Algorithm 5. The overall time complexity of FarMinrecc is $\widetilde{O}\left(km/\epsilon^2\right)$.

On the other hand, algorithm CenMinrecc is a simple variant based on Algorithm FarMinrecc, which avoids running ApproxER for k iterations. Nodes in the same cluster of the graph have a small resistance distance, while two nodes in different clusters of the graph have a large resistance distance [79]. The algorithm CenMinrecc aims to compute a k-center on graph $\mathcal{G}$ with the first center chosen being source node s. First, we find the node farthest from the source node s in terms of resistance distance and add it to the set T. Then, in each iteration, we find the node farthest from all nodes in set T in terms of resistance distance and add it to T. This process is repeated k times. The fast approximation algorithm CenMinrecc is presented in Algorithm 6. The overall time complexity of CenMinrecc is $\widetilde{O}\left(m/\epsilon^2 + kn/\epsilon^2\right)$.

B. Fast Approximation Algorithm of REM

For Problem 2, as the candidate edge set is $Q_2 = (V \times V)\setminus E$, we can add edges that are not connected to the source node s. However, as shown in Figure 3, directly adding edges to the source node s may not necessarily be the optimal choice for Problem 2.

Before introducing our algorithm, based on Lemma 5.1, we propose a fast algorithm ApproxREcc for computing the resistance eccentricity of a given node s in the graph. The algorithm ApproxREcc($\mathcal{G}, \epsilon, s$) runs in $\widetilde{O}\left(m/\epsilon^2\right)$ time, and outputs an approximation value $\tilde{c}(s)$ of resistance eccentricity $c(s)$ for graph $\mathcal{G}$ satisfying $\tilde{c}(s) \overset{\epsilon}{\approx} c(s)$.

```
Algorithm 6: CenMinrecc($\mathcal{G}, Q, k, \epsilon, s$)
Input  : A connected graph $\mathcal{G} = (V, E)$, a candidate
         edge set $Q$, an integer $1 \le k \le |Q|$, a
         parameter $\epsilon$, a source node $s$
Output : $P$ : A subset of $Q$ with $|P| = k$ edges
1 Initialize solution $P = \emptyset$, $T = \emptyset$
2 $T \leftarrow T \cup \{s\}$
3 $d = \lceil 24 \log n/\epsilon^2 \rceil$
4 $\tilde{\boldsymbol{X}}_{d\times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
5 for $i = 1$ to $k$ do
6     Select $u_i \leftarrow$
        $\arg\max_{u\in V/T, v\in T,(s,u)\in Q\setminus P} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_u - \boldsymbol{e}_v)||_2^2$
7     Update $T \leftarrow T \cup \{u_i\}$
8     Update solution $P \leftarrow P \cup \{(s, u_i)\}$
9 return $P$
```

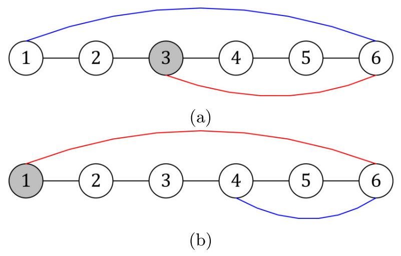

Fig. 6. Two identical line graphs, each with 6 nodes and 5 edges. The colored lines represent newly added edges.

Given the consideration of resistance distances, Problem 2 can be analyzed from the perspective of electrical networks. By replacing every edge in graph $\mathcal{G}$ with a unit resistance, we obtain electrical network associated with graph $\mathcal{G}$. In electrical networks, directly connecting the node farthest from the source node s in terms of resistance distance would lead to the largest resistance circuit being bypassed by a single edge, resulting in a reduction in the resistance eccentricity of s. However, connecting the two nodes farthest from s would cause two circuits with largest resistance to be formed in parallel, which can significantly reduce the resistance eccentricity of s. Hence, we propose the heuristic optimization strategy of connecting the two nodes farthest from s in terms of resistance distance at each step. Meanwhile, through experiments on several small graphs, we observed that for a given source node s, connecting the two nodes farthest from s in terms of resistance distance offers a more effective reduction of $c(s)$ compared to directly adding edges to s. As shown in Figure 6(a), directly adding an edge to the source node, we add edge $(3, 6)$ to minimize the resistance eccentricity of node 3, resulting in $c(3) = 2$. However, the two nodes farthest from node 3 in terms of resistance distance are node 1 and node 6. By adding the

Algorithm 7: ApproxREcc($\mathcal{G}, \epsilon, s$)

```
Input  : A connected graph $\mathcal{G} = (V, E)$, a
         parameter $\epsilon$, a source node $s$
Output : $\tilde{c}(s)$ : The approximate resistance
         eccentricity of $s$
1 $d = \lceil 24 \log n/\epsilon^2 \rceil$
2 $\tilde{\boldsymbol{X}}_{d\times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
3 $\tilde{c}(s) = \max_{u\in V} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_s - \boldsymbol{e}_u)||_2^2$
4 return $\tilde{c}(s)$
```

Algorithm 8: CHMinrecc($\mathcal{G}, Q, k, \epsilon, s$)

```
Input  : A connected graph $\mathcal{G} = (V, E)$, a candidate
         edge set $Q$, an integer $1 \leq k \leq |Q|$, a
         parameter $\epsilon$, a source node $s$
Output : $P$ : A subset of $Q$ with $|P| = k$ edges
1  Initialize solution $P = \emptyset$
2  for $i = 1$ to $k$ do
3      $d = \lceil 24 \log n/\epsilon^2 \rceil$, $\theta = \frac{\epsilon}{12}$
4      $\tilde{\boldsymbol{X}}_{d\times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
5      $S \leftarrow \{s_j \in \mathbb{R}^d \mid s_j = \tilde{\boldsymbol{X}}_{[:,j]}, j = 1, 2, \ldots, n\}$
6      $\hat{S} \leftarrow$ ApproxCH($S, \theta$)
7      Initialize solution $T = \emptyset$
8      $T \leftarrow T \cup \{e = (u, v) \mid u, v \in \hat{S}, (u, v) \in Q \backslash P\}$
9      Select
         $e_i \leftarrow \arg\min_{e\in T}$ ApproxREcc($\mathcal{G}(\{e\}), \epsilon, s$)
10     Update solution $P \leftarrow P \cup \{e_i\}$
11     Update the graph $\mathcal{G} \leftarrow \mathcal{G}(\{e_i\})$
12 return $P$
```

edge $(1, 6)$, we obtain $c(3) = 1.5$. Therefore, connecting the two nodes farthest from s in terms of resistance distance at each step seems to be a highly effective optimization strategy. Based on this idea, we propose a fast algorithm CHMinrecc for Problem 2, the pseudocode of which is illustrated in Algorithm 8. The algorithm CHMinrecc builds upon the ApproxCH algorithm as described in Lemma 5.3 and employs Algorithm 7 for approximating the resistance eccentricity. For n points in an Euclidean space, those points lie on the boundary of the convex hull of these point set are most distant from one another and the two points farthest from s are also among these points. Using ApproxCH, we can obtain a set $\hat{S}$ of l nodes, all of which lie on the boundary of the convex hull. Our algorithm, CHMinrecc, finds a pair of nodes from $\hat{S}$ and connect them with an edge to minimize the resistance eccentricity of node s. With this approach, we achieve a fast and efficient solution to Problem 2. The total running time of CHMinrecc is $\widetilde{O}\left(kl^2m/\epsilon^2\right)$.

However, the strategy of Algorithm 8 is not always the optimal solution for all cases of Problem 2. As shown in Figure 6(b), when the source node is node 1, the optimal solution is to add edge $(1, 6)$ to minimize $c(1)$ to 1.5, rather than adding the edge $(4, 6)$ obtained through Algorithm 8,

Algorithm 9: Minrecc($\mathcal{G}, Q, k, \epsilon, s$)

```
Input  : A connected graph $\mathcal{G} = (V, E)$, a candidate
         edge set $Q$, an integer $1 \leq k \leq |Q|$, a
         parameter $\epsilon$, a source node $s$
Output : $P$ : A subset of $Q$ with $|P| = k$ edges
1  Initialize solution $P = \emptyset$
2  for $i = 1$ to $k$ do
3      $d = \lceil 24 \log n/\epsilon^2 \rceil$, $\theta = \frac{\epsilon}{12}$
4      $\tilde{\boldsymbol{X}}_{d\times n} \leftarrow$ ApproxER($\mathcal{G}, \epsilon$)
5      $S \leftarrow \{s_j \in \mathbb{R}^d \mid s_j = \tilde{\boldsymbol{X}}_{[:,j]}, j = 1, 2, \ldots, n\}$
6      $\hat{S} \leftarrow$ ApproxCH($S, \theta$)
7      Initialize solution $T = \emptyset$
8      $T \leftarrow T \cup \{e = (u, v) \mid u, v \in \hat{S}, (u, v) \in Q \backslash P\}$
9      $e' = \arg\max_{u\in\hat{S},(s,u)\in Q\backslash P} ||\tilde{\boldsymbol{X}}(\boldsymbol{e}_s - \boldsymbol{e}_u)||_2^2$
10     $T \leftarrow T \cup \{e'\}$
11     $e_i = \arg\min_{e\in T}$ ApproxREcc($\mathcal{G}(\{e\}), \epsilon, s$)
12     Update solution $P \leftarrow P \cup \{e_i\}$
13     Update the graph $\mathcal{G} \leftarrow \mathcal{G}(\{e_i\})$
14 return $P$
```

which would result in $c(1) = 3.6$. Therefore, both the strategy of directly adding edges to the source node and selecting edges from the boundary of the convex hull can potentially achieve the optimal solution. Combining these two ideas, we propose Algorithm 9. Minrecc effectively integrates these strategies during each edge addition, selecting the more optimal solution to minimize the resistance eccentricity of the source node. The overall time complexity of Minrecc is $\widetilde{O}\left(kl^2m/\epsilon^2\right)$.

VIII. Experiments

In this section, we present experimental results to evaluate the performance of the proposed approximation algorithms FastQuery for querying the resistance eccentricity, as well as FarMinrecc and CenMinrecc for REMD (Problem 1), CHMinrecc and Minrecc for REM (Problem 2).

A. Experimental Setup

Datasets. To evaluate the performance (efficiency and accuracy) of our proposed approximation algorithms, we perform experiments on different real-world networks representatively selected from various domains, which are from Koblenz Network Collection [62] and Network Repository [63]. Since we are only concerned with connected graphs, for those networks with multiple components, we conduct experiments on the largest connected component (LCC). Table II reports the related information for the studied networks. The smallest network has 614 nodes, while the largest one consists of more than four millions nodes.

Implementation Details. All our experiments are conducted on a Linux box with an Intel i7-7700K @ 4.2-GHz (4 Cores) and with 128-GB RAM. All algorithms are implemented in *Julia v1.0.3*, where the Laplace Solver is used from [80].

B. Performance of Algorithm FASTQUERY Querying for Resistance Eccentricity

We now test the performance of algorithm FASTQUERY on real-world networks, in terms of efficiency and accuracy.

Let us first evaluate the efficiency of algorithm FASTQUERY. In Table II, we report the running time of EXACTQUERY and FASTQUERY on some real-world networks. To objectively evaluate the running time, for both EXACTQUERY and FASTQUERY on all considered networks (except the last seven ones marked with *), we enforce the program to run on a single thread. From Table II, we observe that for small networks with less than $10,000$ nodes, EXACTQUERY is more efficient than FASTQUERY. However, for larger networks and various approximation parameters ϵ, the computational time for FASTQUERY is significantly smaller than that for EXACTQUERY. For the seven networks denoted with an asterisk in Table II, for which the number of nodes is ranging from 10^6 to 10^7, the EXACTQUERY algorithm is not executable on our system due to memory and computational constraints. The EXACTQUERY algorithm necessitates the direct inversion of the graph's Laplacian matrix, a task that becomes computationally intractable for expansive networks. In contrast, for these networks, we can approximately compute their resistance eccentricity distribution by applying algorithm FASTQUERY, which further demonstrates that FASTQUERY is efficient and scalable to very large networks.

In addition to the high efficiency, algorithm FASTQUERY also provides a good approximation $\hat{\mathcal{E}}(\mathcal{G})$ for the resistance eccentricity distribution $\mathcal{E}(\mathcal{G})$. To show the accuracy of FASTQUERY, we compare the approximate values $\hat{\mathcal{E}}(\mathcal{G})$ returned by FASTQUERY with the exact results of $\mathcal{E}(\mathcal{G})$ returned by EXACTQUERY. In Table II, we report the relative errors σ of algorithm FASTQUERY, where σ is defined as

$$\sigma = \frac{1}{|V|}\sum_{v\in V}\left|\frac{\widetilde{c}(v)-c(v)}{c(v)}\right|. \qquad (8)$$

From Table II, we can see that the actual relative errors for all ϵ and all networks are very small, and are almost negligible for smaller ϵ. More interestingly, for all networks tested, the values of σ are much smaller than the theoretical guarantee. Therefore, the approximation algorithm FASTQUERY provides a very accurate result for resistance eccentricity distribution in practice.

Now, we provide resistance eccentricity distribution plots. We selected the four largest networks for experimental analysis, namely: Wikipedia-growth, Web-baidu-baike, Soc-orkut, and Live-journal. Given that these networks each has millions of nodes, we employed the algorithm FASTQUERY to approximate the calculation of the networks' resistance eccentricity distributions. Figure 7 plots the approximated resistance eccentricity distribution for each network: the x-axis represents the resistance eccentricity value which is between the resistance radius and the resistance diameter and the y-axis is the number of nodes with that value. From Figure 7, it is evident that the resistance eccentricity distribution of these

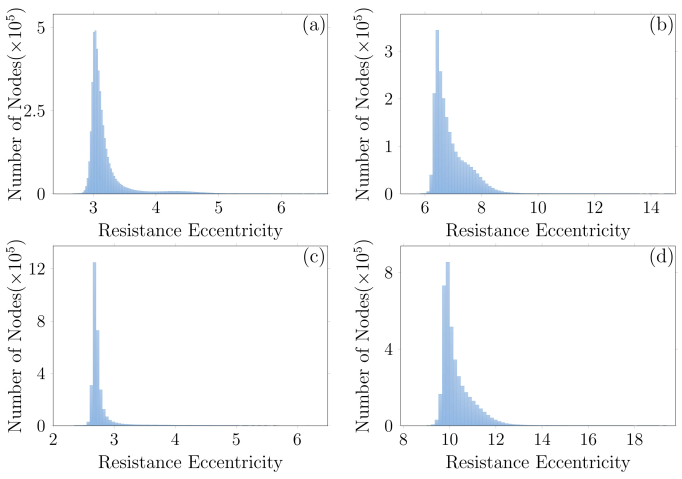

Fig. 7. The resistance eccentricity distribution of each network: Wikipedia-growth (a), Web-baidu-baike(b), Soc-orkut (c), and Live-journal (d).

four networks exhibits properties of asymmetry, rightward skewness, and a pronounced heavy tail, which aligns with our previous analysis. The simultaneous observation also supports the accuracy of our algorithm FASTQUERY.

C. Performance of Algorithms for Minimizing Resistance Eccentricity of a Given Node

In this section, we conduct extensive experiments to evaluate the effectiveness and efficiency of our proposed algorithms for minimizing resistance eccentricity of a given node on various real-world networks.

1) Baseline Methods: Our proposed algorithms, FARMINRECC, CENMINRECC, CHMINRECC and MINRECC, are compared with several baseline methods, which are briefly summarized as follows. For REMD and REM, our baseline methods are respectively based on the corresponding candidate edge sets, Q_1 and Q_2 (see Problem 1 and Problem 2), providing specific approaches for each.

- OPT-REMD: choose k edges from Q_1 forming the set P that minimizes the resistance eccentricity of s by exhaustive search.
- OPT-REM: choose k edges from Q_2 forming the set P that minimizes the resistance eccentricity of s by exhaustive search.
- DE-REMD: choose an edge from Q_1 of the updated graph each time, where the two endpoints of the edge are node s and the node with the lowest degree, and repeat this process k times.
- DE-REM: choose an edge from Q_2 of the updated graph each time, where the two endpoints of the edge are the nodes with the lowest degree, and repeat this process k times.
- PK-REMD: choose an edge from Q_1 in the updated graph each time, where the two endpoints of the edge are node s and the node with the lowest PageRank centrality, and repeat this process k times.

TABLE II

For each network $\mathcal{G}$, we indicate the number of nodes n and the number of edges m. The running time (seconds, s) of ExactQuery and FastQuery with various ϵ, Relative error σ ($\times 10^{-2}$) on various networks.

Network	n	m	Running time (s) for ExactQuery and FastQuery				Relative error σ		
			Exact	0.3	0.2	0.1	0.3	0.2	0.1
Unicode-language	614	1,252	0.111	2.01	2.98	4.65	0.82	0.34	0.02
EmailUN	1,133	5,451	0.425	2.821	3.125	4.045	1.14	0.82	0.18
MusaeRU	4,385	37,304	10.218	7.48	7.501	12.685	1.03	0.75	0.33
Bitcoinotc	5,875	35,587	20.836	7.509	8.498	18.189	1.02	0.88	0.09
Politician	5,908	41,706	21.221	14.35	15.335	20.191	0.74	0.64	0.15
Government	7,057	89,429	35.108	8.13	21.915	51.605	1.06	0.83	0.16
Wiki-Vote	7,066	103,663	39.875	9.324	19.289	29.615	0.96	0.77	0.25
MusaeENGB	7,126	35,324	36.782	11.42	22.469	114.909	0.89	0.57	0.07
HepTh	8,361	15,751	23.174	33.395	49.37	153.79	0.57	0.28	0.19
Cond-mat	13,861	44,619	242.199	42.405	54.95	122.39	1.07	0.88	0.47
Musae-facebook	22,470	170,823	315.303	114.42	175.145	189.325	1.01	0.85	0.24
HU	47,538	222,887	1718.067	233.07	263.255	451.085	0.97	0.72	0.66
HR	54,573	498,202	2689.555	187.08	237.915	613.35	1.04	0.76	0.28
Epinions	75,877	508,836	6101.568	178.789	381.704	551.629	0.99	0.82	0.37
Delicious*	536,108	1,365,961	–	1048.794	1341.102	8876.461	–	–	–
FourSquare*	639,014	3,214,986	–	1163.352	2864.142	6775.753	–	–	–
Youtube-snap*	1,134,890	2,987,624	–	6985	8123	15471	–	–	–
Wikipedia-growth*	1,870,521	39,953,004	–	8126	11891	21378	–	–	–
Web-baidu-baike*	2,107,689	17,758,243	–	7362	10274	18185	–	–	–
Soc-orkut*	2,997,166	106,349,209	–	10941	14517	29592	–	–	–
Live-journal*	4,033,137	27,933,062	–	10887	17851	32182	–	–	–

- PK-REM: choose an edge from Q_2 in the updated graph each time, where the two endpoints of the edge are the nodes with the lowest PageRank centrality, and repeat this process k times.
- Path-REMD: choose the node in Q_1 that has the longest shortest-path distance (i.e., maximum distance) from s in the updated graph and connect them each time; repeat this process k times.
- Path-REM: choose the nodes in Q_2 with the longest shortest-path distance (i.e., maximum distance) in the updated graph and connect them each time; repeat this process k times.

2) *Effectiveness:* We first study the effectiveness of our algorithms Simple, FarMinrecc, CenMinrecc, CHMinrecc and Minrecc, by comparing them with OPT-REM and OPT-REMD. We execute experiments on four small networks: Kangaroo with 17 nodes and 91 edges, Rhesus with 16 nodes and 111 edges, Cloister with 18 nodes and 189 edges and Tribes with 16 nodes and 58 edges. Their small size allows us to compute the optimal set of added edges. Since computing the optimal solution requires exponential time in k, we only consider $k = 0, 1, 2, 3, 4$. The results are shown in Figure **??**, which demonstrate that the resistance eccentricities of the source node in the augmented graphs returned by our greedy algorithms and the optimum solutions are almost the same.

In order to further demonstrate the effectiveness of FarMinrecc, CenMinrecc, CHMinrecc and Minrecc, we also compare their returned results with four baseline schemes: DE-REMD, DE-REM, PK-REMD, PK-REM, Path-REMD, and Path-REM, on four large real networks.

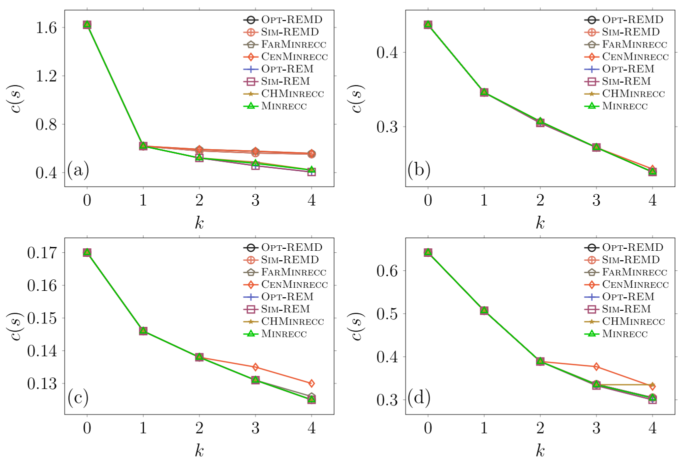

Fig. 8. The resistance eccentricity $c(s)$ of a given node s within augmented graphs—produced by our proposed algorithms for $k = 1, 2, 3, 4$ on four distinct networks: (a) Kangaroo, (b) Rhesus, (c) Cloister, and (d) Tribes.

For each network, we calculate the resistance eccentricity of s in the original graph. Then we decrease the resistance eccentricity of s by adding up to $k = 1, 2, \ldots, 50$ new edges, applying our greedy algorithms and the four baseline strategies of edge addition. After adding each edge by different methods, we compute and record the resistance eccentricity. The results are shown in Figure 9, which indicates that for each network, our greedy algorithms outperform the baseline strategies.

Finally, we execute experiments on four much larger networks to display the effectiveness of FarMinrecc, CenMinrecc, CHMinrecc and Minrecc. For these networks,

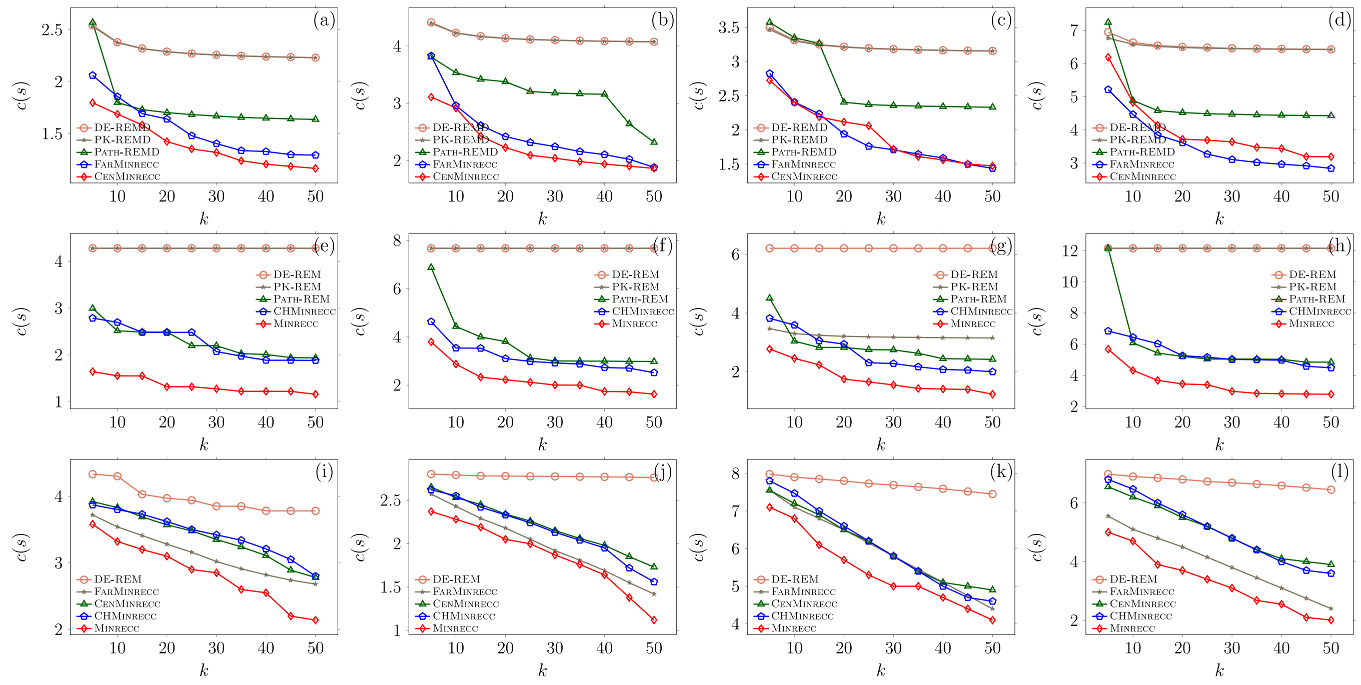

Fig. 9. The resistance eccentricity $c(s)$ of a given node s within augmented graphs—produced by our proposed algorithms for $k = 1, 2, \ldots, 50$ on eight networks, EmailUN (a)(e), Politician (b)(f), Government (c)(g), HepTh (d)(h), Wikipedia-growth (i), Web-baidu-baike (j), Soc-orkut (k), and Live-journal (l).

we cannot run baselines PK-REMD, PATH-REMD, PK-REM and PATH-REM, due to their high complexity, but only run DE-REMD and DE-REM. Since the candidate edge set of DE-REMD is encompassed within that of DE-REM, for convenience, we only employ DE-REM for comparison with our proposed algorithms. The results, as reported in Figure 9, demonstrate that FARMINRECC, CENMINRECC, CHMINRECC, and MINRECC yield significantly better outcomes compared to DE-REM.

3) Efficiency: As demonstrated above, in comparison with the baseline strategies of edge addition, our algorithms FARMINRECC, CENMINRECC, CHMINRECC and MINRECC exhibit good effectiveness. Here, we study the efficiency of our algorithms. For this purpose, we run them on four large-scale networks, with all over one million nodes. For each network, we select $k = 50$ and record the running time of both algorithms in Table III. The results show that FARMINRECC, CENMINRECC, CHMINRECC and MINRECC are all efficient, and are scalable to networks with millions of nodes. However, there is some difference in the performance between FARMINRECC and CENMINRECC. As shown in Table III, CENMINRECC runs faster than FARMINRECC. For example, the running time of FARMINRECC is nearly twice that of CENMINRECC for the network Live-journal. Although CENMINRECC is more efficient than FARMINRECC, Figures 9 illustrate that FARMINRECC is much effective than CENMINRECC. Simultaneously, as MINRECC encompasses CHMINRECC, it can be observed from Figures 9 that CHMINRECC is more efficient than MINRECC, but MINRECC is significantly more effective than CHMINRECC.

TABLE III
THE RUNNING TIME (SECONDS, s) OF FARMINRECC, CENMINRECC CHMINRECC AND MINRECC ON SOME LARGE-SCALE NETWORKS.

Network	Running time (s)			
	FARMINRECC	CENMINRECC	CHMINRECC	MINRECC
Wikipedia-growth	19111	7872	19978	50689
Web-baidu-baike	18932	7112	19325	47833
Soc-orkut	19771	8218	20417	51512
Live-journal	20135	11326	22382	68141

IX. CONCLUSIONS

In this work, we tackle the problem of computing the resistance eccentricity distribution, $\mathcal{E}(\mathcal{G})$, for large-scale graphs $\mathcal{G} = (V, E)$ comprising n nodes and m edges. Traditional methods for calculating resistance eccentricity are computationally expensive due to the need for matrix inversion and pairwise resistance distance calculations. To overcome this, we introduce FASTQUERY, an efficient approximation algorithm that estimates the resistance eccentricity for individual or multiple nodes with a guaranteed error bound. This algorithm utilizes the Johnson-Lindenstrauss Lemma, Laplacian solvers, and convex hull techniques to bypass direct matrix inversion, significantly reducing computational complexity.

Additionally, we explore the minimization of resistance eccentricity by adding k edges to the graph. We present two optimization problems, Problem 1 and Problem 2, with the aim of selecting k edges from different candidate sets to lower the resistance eccentricity of a specific node. Despite the non-supermodular yet monotone nature of the objective functions, we devised five heuristic algorithms—SIMPLE, FARMINRECC, CENMINRECC, CHMINRECC, and MINRECC—with near-linear time complexity that address these optimization problems.

Our comprehensive experiments on a variety of real-world networks validate the efficiency and effectiveness of our algorithms. The results confirm that our approaches, including FASTQUERY and the heuristics for edge addition, are scalable and practical for networks exceeding a million nodes, offering a significant advancement in the computation and optimization of resistance eccentricity for large-scale graphs.

Future research directions present exciting opportunities. Exploring additional innovative approximation and graph sparsification methods could enhance the speed of our algorithms without compromising their precision. Moreover, the proof strategies formulated in this study have the potential to be adapted for advancing algorithmic solutions for a variety of network optimization problems. Specifically, these methodologies may be instrumental in refining algorithms aimed at optimizing metrics such as Kemeny's constant.

REFERENCES

[1] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the fortieth Annual ACM Symposium on Theory of Computing*, 2008, pp. 563–568.

[2] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proc. 43rd Annu. ACM Symp. Theory Comput.*, San Jose, CA, USA, June 2011, pp. 273–282.

[3] A. Madry, D. Straszak, and J. Tarnawski, "Fast generation of random spanning trees and the effective resistance metric," in *Proc. 26th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Diego, CA, USA, Jan. 2015, pp. 2019–2036.

[4] H. Li, S. Patterson, Y. Yi, and Z. Zhang, "Maximizing the number of spanning trees in a connected graph," *IEEE Trans. Inf. Theory*, vol. 66, no. 2, pp. 1248–1260, 2020.

[5] N. Anari and S. O. Gharan, "Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp," in *Proc. 56th Annu. IEEE Symp. Found. Comput. Sci.*, Berkeley, CA, USA, Oct. 2015, pp. 20–39.

[6] F. Fouss, A. Pirotte, J.-m. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data. Eng.*, vol. 19, no. 3, pp. 355–369, Jan. 2007.

[7] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data. Eng.*, vol. 30, no. 9, pp. 1616–1637, 2018.

[8] R. Behmo, N. Paragios, and V. Prinet, "Graph commute times for image representation," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[9] U. Brandes and D. Fleischer, "Centrality measures based on current flow," in *Proceedings of Annual Symposium on Theoretical Aspects of Computer Science*, vol. 3404. Springer, 2005, pp. 533–544.

[10] L. Shan, Y. Yi, and Z. Zhang, "Improving information centrality of a node in complex networks by adding edges," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3535–3541.

[11] H. Li, R. Peng, Y. Shan, Lirenand Yi, and Z. Zhang, "Current flow group closeness centrality for complex networks?" in *Proc. World Wide Web Conf.*, San Francisco, USA, May 2019, pp. 961–971.

[12] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–33, 2016.

[13] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 60, no. 1, pp. 150–163, 2013.

[14] F. Dörfler, J. W. Simpson-Porco, and F. Bullo, "Electrical networks and algebraic graph theory: Models, properties, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 977–1005, 2018.

[15] K. Thulasiraman, M. Yadav, and K. Naik, "Network science meets circuit theory: Resistance distance, Kirchhoff index, and Foster's theorems with generalizations and unification," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 66, no. 3, pp. 1090–1103, 2019.

[16] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966.

[17] D. J. Klein and M. Randić, "Resistance distance," *Journal of Mathematical Chemistry*, vol. 12, pp. 81–95, 1993.

[18] A. Ghosh, S. Boyd, and A. Saberi, "Minimizing effective resistance of a graph," *SIAM Review*, vol. 50, no. 1, pp. 37–66, 2008.

[19] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social networks*, vol. 11, no. 1, pp. 1–37, 1989.

[20] M. Li, S. Zhou, D. Wang, and G. Chen, "Identifying influential nodes based on resistance distance," *Journal of Computational Science*, vol. 67, p. 101972, 2023.

[21] D. Magoni and J. J. Pansiot, "Analysis of the autonomous system network topology," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, pp. 26–37, 2001.

[22] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos, "Using graph theory to analyze biological networks," *BioData Mining*, vol. 4, pp. 1–27, 2011.

[23] M. Mneimneh and K. Sakallah, "Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution," in *Theory and Applications of Satisfiability Testing: 6th International Conference, SAT 2003, Santa Margherita Ligure, Italy, May 5-8, 2003, Selected Revised Papers 6*. Springer, 2004, pp. 411–425.

[24] Y. Liu, B. Yan, J. Liu, H. Su, H. Zheng, and Y. Cai, "From the periphery to the core: Information brokerage in an evolving network," in *IJCAI'18 Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 2018.

[25] T. Akiba, Y. Iwata, and Y. Kawata, "An exact algorithm for diameters of large real directed graphs," in *Experimental Algorithms: 14th International Symposium, SEA 2015, Paris, France, June 29–July 1, 2015, Proceedings 14*. Springer, 2015, pp. 56–67.

[26] K. Iwabuchi, G. Sanders, K. Henderson, and R. Pearce, "Computing exact vertex eccentricity on massive-scale distributed graphs," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 257–267.

[27] J. Shun, "An evaluation of parallel eccentricity estimation algorithms on undirected real-world graphs," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1095–1104.

[28] W. Li, M. Qiao, L. Qin, L. Chang, Y. Zhang, and X. Lin, "On scalable computation of graph eccentricities," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 904–916.

[29] S. Perumal, P. Basu, and Z. Guan, "Minimizing eccentricity in composite networks via constrained edge additions," in *MILCOM 2013-2013 IEEE Military Communications Conference*. IEEE, 2013, pp. 1894–1899.

[30] E. D. Demaine and M. Zadimoghaddam, "Minimizing the diameter of a network using shortcut edges," in *Algorithm Theory-SWAT 2010: 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings 12*. Springer, 2010, pp. 420–431.

[31] F. Frati, S. Gaspers, J. Gudmundsson, and L. Mathieson, "Augmenting graphs to minimize the diameter," *Algorithmica*, vol. 72, pp. 995–1010, 2015.

[32] F. W. Takes and W. A. Kosters, "Computing the eccentricity distribution of large graphs," *Algorithms*, vol. 6, no. 1, pp. 100–118, 2013.

[33] ——, "Determining the diameter of small world networks," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1191–1196.

[34] C. Mavroforakis, R. Garcia-Lebron, I. Koutis, and E. Terzi, "Spanning edge centrality: Large-scale computation and applications," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 732–742.

[35] T. Hayashi, T. Akiba, and Y. Yoshida, "Efficient algorithms for spanning tree centrality," in *IJCAI*, 2016, pp. 3733–3739.

[36] D. B. Wilson, "Generating random spanning trees more quickly than the cover time," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 1996, pp. 296–303.

[37] P. Peng, D. Lopatta, Y. Yoshida, and G. Goranci, "Local algorithms for estimating effective resistance," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1329–1338.

[38] M. Liao, R.-H. Li, Q. Dai, H. Chen, H. Qin, and G. Wang, "Efficient resistance distance computation: The power of landmark-based approaches," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–27, 2023.

[39] R. Yang and J. Tang, "Efficient estimation of pairwise effective resistance," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–27, 2023.

[40] P. H. Chan, L. C. Lau, A. Schild, S. C.-w. Wong, and H. Zhou, "Network design for $s-t$ effective resistance," *ACM Transactions on Algorithms*, vol. 18, no. 3, pp. 1–45, 2022.

[41] A. Tizghadam and A. Leon-Garcia, "Autonomic traffic engineering for network robustness," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 1, Jan. 2010.

[42] S. Patterson and B. Bamieh, "Consensus and coherence in fractal networks," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 4, pp. 338–348, Dec. 2014.

[43] Y. Qi, Z. Zhang, Y. Yi, and H. Li, "Consensus in self-similar hierarchical graphs and Sierpiński graphs: Convergence speed, delay robustness, and coherence," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 592–603, 2019.

[44] Y. Yi, Z. Zhang, and S. Patterson, "Scale-free loopy structure is resistant to noise in consensus dynamics in complex networks," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 190–200, 2020.

[45] T. Summers, I. Shames, J. Lygeros, and F. Dörfler, "Topology design for optimal network coherence," in *2015 European Control Conference*. IEEE, 2015, pp. 575–580.

[46] C. Yang, J. Mao, X. Qian, and P. Wei, "Designing robust air transportation networks via minimizing total effective resistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2353–2366, 2018.

[47] G. Huang, W. He, and Y. Tan, "Theoretical and computational methods to minimize Kirchhoff index of graphs with a given edge k-partiteness," *Applied Mathematics and Computation*, vol. 341, pp. 348–357, 2019.

[48] M. Predari, R. Kooij, and H. Meyerhenke, "Faster greedy optimization of resistance-based graph robustness," in *2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2022, pp. 1–8.

[49] M. Black, Z. Wan, A. Nayyeri, and Y. Wang, "Understanding oversquashing in gnns through the lens of effective resistance," in *International Conference on Machine Learning*. PMLR, 2023, pp. 2528–2547.

[50] P. Crescenzi, G. D'angelo, L. Severini, and Y. Velaj, "Greedily improving our own closeness centrality in a network," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 1, pp. 1–32, 2016.

[51] G. D'Angelo, M. Olsen, and L. Severini, "Coverage centrality maximization in undirected networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 501–508.

[52] H. Li, S. Patterson, Y. Yi, and Z. Zhang, "Maximizing the number of spanning trees in a connected graph," *IEEE Transactions on Information Theory*, vol. 66, no. 2, pp. 1248–1260, 2019.

[53] X. Zhou and Z. Zhang, "Maximizing influence of leaders in social networks," in *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2021, pp. 2400–2408.

[54] L. Zhu, Q. Bao, and Z. Zhang, "Minimizing polarization and disagreement in social networks via link recommendation," vol. 34, 2021, pp. 2072–2084.

[55] F. Adriaens and A. Gionis, "Diameter minimization by shortcutting with degree constraints," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 843–848.

[56] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 6605–6611.

[57] A. Ben-Israel and T. N. E. Greville, *Generalized inverses: Theory and applications*. J. Wiley, 1974.

[58] P. G. Doyle and J. L. Snell, *Random walks and electric networks*. American Mathematical Soc., 1984, vol. 22.

[59] W. Xu and Z. Zhang, "Optimal scale-free small-world graphs with minimum scaling of cover time," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 7, p. 93, 2023.

[60] A. Firat, S. Chatterjee, and M. Yilmaz, "Genetic clustering of social networks using random walks," *Computational Statistics & Data Analysis*, vol. 51, no. 12, pp. 6285–6294, 2007.

[61] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 355–369, 2007.

[62] J. Kunegis, "Konect: The koblenz network collection," in *Proceedings of the 22nd International World Wide Web Conference*. ACM, 2013, pp. 1343–1350.

[63] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI, 2015, pp. 4292–4293.

[64] S. Foss, D. Korshunov, S. Zachary, *et al.*, *An introduction to heavy-tailed and subexponential distributions*. Springer, 2011, vol. 6.

[65] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemp. Math.*, vol. 26, pp. 189–206, 1984.

[66] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 2001, pp. 274–281.

[67] ——, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[68] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the thirty-sixth Annual ACM Symposium on Theory of Computing*. ACM, 2004, pp. 81–90.

[69] ——, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 3, pp. 835–885, 2014.

[70] I. Koutis, G. L. Miller, and R. Peng, "A nearly-m log n time solver for SDD linear systems," in *Proc. IEEE 52nd Ann. Symp. Found. Comput. Sci.* IEEE, 2011, pp. 590–598.

[71] O. E. Livne and A. Brandt, "Lean algebraic multigrid (LAMG): Fast graph laplacian linear solver," *SIAM J. Sci. Comput.*, vol. 34, no. 4, pp. B499–B522, 2012.

[72] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving SDD linear systems in nearly m log 1/2 n time," in *Proc. 46th Ann. ACM Symp. Theory Comput.* ACM, 2014, pp. 343–352.

[73] F. P. Preparata and M. I. Shamos, "Computational geometry: An introduction," *Texts and Monographs in Computer Science*, 1985.

[74] B. Chazelle, "An optimal convex hull algorithm in any fixed dimension," *Discrete & Computational Geometry*, vol. 10, no. 4, pp. 377–409, 1993.

[75] d. B. Mark, C. Otfried, v. K. Marc, and O. Mark, *Computational geometry algorithms and applications*. Spinger, 2008.

[76] P. Awasthi, B. Kalantari, and Y. Zhang, "Robust vertex enumeration for convex hulls in high dimensions," *Ann. Oper. Res.*, vol. 295, no. 1, pp. 37–73, 2020.

[77] P. H. Chan, L. C. Lau, A. Schild, S. C.-w. Wong, and H. Zhou, "Network design for s-t effective resistance," *ACM Transactions on Algorithms (TALG)*, vol. 18, no. 3, pp. 1–45, 2022.

[78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical Programming*, vol. 14, pp. 265–294, 1978.

[79] U. Luxburg, A. Radl, and M. Hein, "Getting lost in space: Large sample analysis of the resistance distance," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[80] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for Laplacians-fast, sparse, and simple," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 573–582.