Improved Algorithms for Effective Resistance Computation on Graphs

Yichun Yang YC.YANG@BIT.EDU.CN
Beijing Institute of Technology

Rong-Hua Li LIRONGHUABIT@126.COM
Beijing Institute of Technology

Meihao Liao MHLIAO@BIT.EDU.CN
Beijing Institute of Technology

Guoren Wang WANGGRBIT@GMAIL.COM
Beijing Institute of Technology

Editors: Nika Haghtalab and Ankur Moitra

Abstract

Effective Resistance (ER) is a fundamental tool in various graph learning tasks. In this paper, we address the problem of efficiently approximating ER on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices and m edges. First, we focus on local online-computation algorithms for ER approximation, aiming to improve the dependency on the approximation error parameter ϵ. Specifically, for a given vertex pair (s, t), we propose a local algorithm with a time complexity of $\tilde{O}(\sqrt{d}/\epsilon)$ to compute an ϵ-approximation of the s, t-ER value for expander graphs, where $d = \min\{d_s, d_t\}$. This improves upon the previous state-of-the-art, including an $\tilde{O}(1/\epsilon^2)$ time algorithm based on random walk sampling by Andoni et al. (ITCS'19) and Peng et al. (KDD'21). Our method achieves this improvement by combining deterministic search with random walk sampling to reduce variance. Second, we establish a lower bound for ER approximation on expander graphs. We prove that for any $\epsilon \in (0, 1)$, there exist an expander graph and a vertex pair (s, t) such that any local algorithm requires at least $\Omega(1/\epsilon)$ time to compute the ϵ-approximation of the s, t-ER value. Finally, we extend our techniques to index-based algorithms for ER computation. We propose an algorithm with $\tilde{O}(\min\{m + n/\epsilon^{1.5}, \sqrt{nm}/\epsilon\})$ processing time, $\tilde{O}(n/\epsilon)$ space complexity and $O(1)$ query complexity, which returns an ϵ-approximation of the s, t-ER value for any $s, t \in \mathcal{V}$ for expander graphs. Our approach improves upon the state-of-the-art $\tilde{O}(m/\epsilon)$ processing time by Dwaraknath et al. (NeurIPS'24) and the $\tilde{O}(m + n/\epsilon^2)$ processing time by Li and Sachdeva (SODA'23).

1. Introduction

Given an undirected, unweighted graph $\mathcal{G}$[1], the Effective Resistance (ER) $r_{\mathcal{G}}(s, t)$ of two vertices s and t is the potential difference between s and t, when a unit electric flow is sent from s to t. Formally, given the graph Laplacian matrix $\mathbf{L}$ and its pseudo-inverse $\mathbf{L}^\dagger$, the s, t ER value is defined as $r_{\mathcal{G}}(s, t) = (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^\dagger (\mathbf{e}_s - \mathbf{e}_t)$, where $\mathbf{e}_s$ (resp., $\mathbf{e}_t$) denotes the one-hot vector that takes value 1 at s (resp., t) and 0 elsewhere. ER has a wide range of applications in graph algorithms and graph learning tasks, including the design of maximum flow algorithms van den Brand et al. (2022); Madry (2016), graph clustering Alev et al. (2018); Saito and Herbster (2023), graph sparsification Spielman and Srivastava (2008), counting random spanning trees Li and Sachdeva (2023) and understanding oversquashing in GNNs Black et al. (2023); Topping et al. (2022). As a result,

1. Though this paper primarily focus on unweighted graphs, all the algorithms and analysis can be directly generalized to weighted graphs with $\min_e w_e = \tilde{\Omega}(1)$ and $\max_e w_e = \tilde{O}(1)$, where w_e denotes the weight of an edge e.

designing efficient algorithms for ER computation has become a fundamental problem, attracting significant attention in recent years Andoni et al. (2019); Peng et al. (2021); Yang and Tang (2023); Liao et al. (2023); Li and Sachdeva (2023); Dwaraknath et al. (2024). In this paper, we focus on computing an ϵ-approximation of ER on a given set of vertex pairs $S \subseteq \mathcal{V} \times \mathcal{V}$. We begin by formally defining the ER approximation problem.

Definition 1 *Given undirected, unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set of vertex pairs $S \subseteq \mathcal{V} \times \mathcal{V}$, a parameter $\epsilon \in (0, 1)$, one requires to find an approximation $\hat{r}_{\mathcal{G}} \in \mathbb{R}^S$ such that $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$[2] with high probability (w.h.p.) for any vertex pairs $(s,t) \in S$. We refer to such $\hat{r}_{\mathcal{G}} \in \mathbb{R}^S$ as an ϵ-approximation of $r_{\mathcal{G}}$ on S.*

The existing algorithms for computing ϵ-approximation of ER can be broadly classified into two categories: (i) online-computation algorithms; and (ii) index-based algorithms. Specifically, for the given graph $\mathcal{G}$, online-computation algorithms directly calculates the s,t-ER value of a given vertex pair s,t. In contrast, index-based algorithms preprocess the graph to construct a data structure, which is then used to efficiently query the s,t-ER value for any given vertex pair s,t. The runtime of the state-of-the-art algorithms are summarized in Table 1 and Table 2. In this paper, we assume all the algorithms support the classic adjacency graph model Ron (2019), which allows the following three types of queries on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in constant time:

- Degree query (given any $u \in \mathcal{V}$, get the degree $deg(u)$ in constant time);
- Neighbor query (given any $u \in \mathcal{V}$, get its i-th neighbor $Neigh(u,i)$ in constant time);
- Jump query (uniformly sample any $u \in \mathcal{V}$ in constant time).

Online-computation algorithms. We begin by focusing on online-computation algorithms to calculate ER value of a single vertex pair. For a given graph $\mathcal{G}$, given a vertex pair (s,t), by the definition $r_{\mathcal{G}}(s,t) = (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^\dagger (\mathbf{e}_s - \mathbf{e}_t)$. Using a nearly linear-time Laplacian solver Sachdeva et al. (2014), one can compute an approximation $\hat{r}_{\mathcal{G}}(s,t)$ such that $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$ in $O(m \log^{1/2} n \log \frac{1}{\epsilon})$ time, where n is the number of vertices, m is the number of edges, ϵ is the approximation error parameter. In recent years, there has been growing interests in the local online-computation of ER, which aims to compute $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$ by just exploring a small portion of the graph. These algorithms typically rely on the assumption that the graph is an expander. Andoni et al. (2019) proposed an algorithm that computes an ϵ-approximation of $r_{\mathcal{G}}(s,t)$ in $\tilde{O}(1/\epsilon^2)$[3] time for d-regular expander graphs. This result was later extended to expander graphs with unbounded degrees by Peng et al. (2021) and Yang and Tang (2023). However, it remains an open question whether the $\tilde{O}(1/\epsilon^2)$ runtime can be improved for ϵ-approximating the ER value for local algorithms. In this paper, we address this problem by proposing a local algorithm for ER computation that achieves a lower time complexity dependency on ϵ. The key idea of our approach is to combine deterministic search with random walks to reduce the variance produced by random walk sampling.

Theorem 2 (↓) *Given an undirected, unweighted expander graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $s, t \in \mathcal{V}$, $\epsilon \in (0,1)$. There exists an algorithm that outputs the approximation $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$ in $\tilde{O}(\sqrt{d}/\epsilon)$ time, where $d = \min\{d_s, d_t\}$.*

2. In this paper, we say $x \approx_\epsilon y$ if $(1-\epsilon)y \leq x \leq (1+\epsilon)y$.
3. The $\tilde{O}(.)$ notation represents the complexity omitting the log term of n, m and ϵ.

Table 1: ER computation of single vertex pair

Methods	Runtime	Assumption
Sachdeva et al. (2014)	$O(m \log^{1/2} n \log \frac{1}{\epsilon})$	none
Andoni et al. (2019)	$\tilde{O}(1/\epsilon^2)$	expanders
Peng et al. (2021)	$\tilde{O}(1/\epsilon^2)$	expanders
Yang and Tang (2023)	$\tilde{O}(1/\epsilon^2)$	expanders
This Paper	$\tilde{O}(\sqrt{d}/\epsilon)$	expanders
Lower Bound (This Paper)	$\Omega(1/\epsilon)$	expanders

There is an additional remark that without the assumption on expander graphs, the time complexity of our algorithm is $\tilde{O}(\kappa(\mathcal{L})^3\sqrt{d}/\epsilon)$, where $\kappa(\mathcal{L})$ is the condition number of the normalized Laplacian matrix $\mathcal{L}$ (see Section 2 for Definition). This condition number dependency is the same as the previous local algorithms Andoni et al. (2019); Peng et al. (2021); Yang and Tang (2023).

Lower bounds. We also study the lower bound for the ER computation of a single pair. Specifically, we prove that the time complexity of any local online-computation algorithm for approximating ER is at least $\Omega(1/\epsilon)$, even holds for expander graphs. This result implies that the $1/\epsilon$ dependency on the approximation parameter ϵ is the best possible, and the gap between our upper bound and the lower bound is only a factor $\sqrt{d}$.

Theorem 3 (↓) *Given any $\epsilon \in (0,1)$, there exists an expander graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two vertices s and t, such that for any (randomized) local algorithm that supports the adjacency model computes the approximation $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$ with success probability $\geq 2/3$ requires $\Omega(1/\epsilon)$ queries.*

We note that the conditional lower bounds by Dwaraknath et al. (2024) implies a $\Omega(n^2/\sqrt{\epsilon})$ lower bound for all pairs ER approximation (i.e., ϵ-approximates $r_{\mathcal{G}}(s,t)$ for all $s,t \in \mathcal{V}$) for non-fast matrix multiplication (FMM) algorithms. This further implies a $\Omega(1/\sqrt{\epsilon})$ lower bound for ER approximation of a single pair. However, by Theorem 3, we prove a stronger lower bound $\Omega(1/\epsilon)$. Furthermore, our construction is simpler and does not rely on the non-FMM assumption.

Index-based algorithms. We show that our techniques can be generalized to index-based ER computations. Following Li and Sachdeva (2023); Dwaraknath et al. (2024), we define the ER sketch algorithm as follows.

Definition 4 *(ER sketch) Given undirected, unweighted $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\epsilon \in (0,1)$, We call a (randomized) algorithm an (T_p, T_q, S_t)-ER sketch algorithm if in $O(T_p)$ processing time it creates a binary string of length $O(S_t)$, from which when queried with $\forall s,t \in \mathcal{V}$, it outputs the approximation $\hat{r}_{\mathcal{G}}(s,t)$ such that $\hat{r}_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{G}}(s,t)$ w.h.p. in $O(T_q)$ time.*

Note that this ER sketch problem is closely related to the computation of graph Laplacian sparsifiers, which has been extensively studied in previous studies Spielman and Srivastava (2008); Durfee et al. (2017); Jambulapati and Sidford (2018); Chu et al. (2018). The current main open question is how to improve the processing time dependency on n, m and ϵ. The state-of-the-art methods for this ER sketch problem on expander graphs include an algorithm with $(\tilde{O}(m + n/\epsilon^2), O(1), \tilde{O}(n/\epsilon))$ complexity based on random walk sampling Li and Sachdeva (2023) and an

Table 2: ER computation of multiple vertex pairs $S \subset \mathcal{V} \times \mathcal{V}$

Methods	Runtime	Assumption
Spielman and Srivastava (2008)	$\tilde{O}(m/\epsilon^2 + \lvert S\rvert/\epsilon^2)$	none
Durfee et al. (2017)	$\tilde{O}(m + n/\epsilon^2 + \lvert S\rvert/\epsilon^2)$	none
Jambulapati and Sidford (2018)	$\tilde{O}(n^2/\epsilon)$	none
Chu et al. (2018)	$\tilde{O}(m + n/\epsilon^{1.5} + \lvert S\rvert/\epsilon^{1.5})$	none
Li and Sachdeva (2023)	$\tilde{O}(m + n/\epsilon^2 + \lvert S\rvert)$	expanders
Dwaraknath et al. (2024)	$\tilde{O}(m/\epsilon + \lvert S\rvert)$	expanders
This Paper	$\tilde{O}(\min\{m + n/\epsilon^{1.5}, \sqrt{nm}/\epsilon\} + \lvert S\rvert)$	expanders
Lower Bound Dwaraknath et al. (2024)	$\Omega(n^{c_1}/\epsilon^{c_2})$ with $c_1 + 2c_2 = 3$	$S = \mathcal{V} \times \mathcal{V}$

algorithm with $(\tilde{O}(m/\epsilon), \tilde{O}(1), \tilde{O}(n/\epsilon))$ complexity based on Count Sketch and Laplacian solver Dwaraknath et al. (2024). In this paper, we show that we can also improve the runtime for the ER sketch problem by combining deterministic search with random walk sampling.

Theorem 5 (↓) *There is an* $(\tilde{O}(\min\{m + n/\epsilon^{1.5}, \sqrt{nm}/\epsilon\}), O(1), \tilde{O}(n/\epsilon))$ *ER sketch algorithm for undirected, unweighted expander graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, and* $\epsilon \in (0, 1)$.

From Theorem 5, we observe that the processing time of our ER sketch algorithm is $T_p = \tilde{O}(\min\{m + n/\epsilon^{1.5}, \sqrt{nm}/\epsilon\})$. For the first term $m + n/\epsilon^{1.5}$, this represents a speedup of $1/\sqrt{\epsilon}$ compared to the $\tilde{O}(m + n/\epsilon^2)$ algorithm proposed in Li and Sachdeva (2023). For the second term $\sqrt{nm}/\epsilon$, this achieves a $\sqrt{m/n}$ speedup over the $\tilde{O}(m/\epsilon)$ algorithm proposed in Dwaraknath et al. (2024). Consequently, our algorithm establishes an improved runtime bound compared to the state-of-the-art methods. There is an additional remark that without the assumption on expander graphs, the processing time of our algorithm is $\tilde{O}(\min\{m+\kappa(\mathcal{L})^3 n/\epsilon^{1.5}, \kappa(\mathcal{L})^3\sqrt{nm}/\epsilon\})$, the space complexity is $\tilde{O}(\kappa(\mathcal{L})n/\epsilon)$, and the query time remains $O(1)$. This condition number dependency is the same as the random walk-based method Li and Sachdeva (2023).

1.1. Technique overview

In this paper, we primarily focus on the Taylor expansion of $\mathbf{L}^\dagger(\mathbf{e}_s - \mathbf{e}_t)$ to compute the ER value $r_{\mathcal{G}}(s, t)$. Based on this Taylor expansion representation, existing local algorithms sampling random walks to approximate $\mathbf{L}^\dagger(\mathbf{e}_s - \mathbf{e}_t)$ to ϵ-approximates $r_{\mathcal{G}}(s, t)$ Andoni et al. (2019); Peng et al. (2021). Our main algorithmic contribution is to combine deterministic search with random walk sampling, thereby reducing the variance introduced by random walks. This approach is inspired by the two-phase algorithm for local PageRank computation Lofgren et al. (2016); Wang et al. (2024); Wei et al. (2024). Specifically, we use a variant of the classical coordinate gradient descent algorithm as the deterministic part to coarsely approximate $\mathbf{L}^\dagger(\mathbf{e}_s - \mathbf{e}_t)$. Subsequently, we sample random walks to refine the residuals produced by the coordinate gradient descent. We carefully analyze the error bounds and complexity of our proposed algorithm, proving that the runtime dependency on ϵ can be improved through the combination of coordinate gradient descent and random walk sampling. For the index-based algorithm, we have the following two key observations: (i) the approximation of $\mathbf{L}^\dagger(\mathbf{e}_s - \mathbf{e}_t)$ is a sparse vector; (ii) random walks can share the computation of the deterministic

part (i.e., coordinate gradient descent). Based on these two observations, we extend our techniques and design an index-based algorithm faster than the state-of-the-art.

Additionally, we study the lower bound for the local computation of single pair ER on expander graphs. Our key observation is the sensitivity of the parallel resistors. Roughly speaking, we consider the following two cases: (i) case C_1: we connect d resistors with resistance 1 in parallel; (ii) case C_2: we connect $(1-\epsilon)d$ resistors with resistance 1 and ϵd resistors with resistance 2 in parallel. We prove that the effective resistance of C_1 and C_2 differs by $\Omega(\epsilon)$ in terms of relative error. However, distinguishing C_1 and C_2 is challenging for local algorithms. This observation leads to the construction of our lower bound.

1.2. Related Work

Local graph algorithms. Local graph algorithms for several related problems have been extensively studied. For instance, Liu and Gleich (2020); Fountoulakis et al. (2019, 2020, 2023) investigate local algorithms for conductance-based graph clustering, while Lofgren and Goel (2013); Lofgren et al. (2016); Bressan et al. (2018); Wang et al. (2024); Wei et al. (2024) explore local algorithms for PageRank computation. Additionally, Cohen-Steiner et al. (2018); Jin et al. (2023) examine local algorithms for graph spectral density approximation, and Andoni et al. (2019) study local algorithms to solve Laplacian systems. The problem setting of our work, which focuses on the local computation of ER for a single-pair vertices, shares similarities with these problems. However, the techniques developed for these related problems cannot be applied directly in our setting.

Lower bounds for ER computation. For the local computation of ER, in addition to our lower bound concerning the approximation parameter ϵ, previous studies have also focused on establishing lower bounds related to the condition number $\kappa(\mathcal{L})$. Andoni et al. (2019) proved that any local algorithm that ϵ-approximates a coordinate of the Laplacian linear system requires at least $\Omega(\kappa(\mathcal{L})^2)$ queries, when setting $\epsilon = \Theta(1/\log n)$. However, since their problem setting differs from our local ER computation problem, their results cannot be directly generalized to our problem. In addition, Cai et al. (2023) provides a lower bound for the single-pair ER approximation problem for non-expander graphs. They proved that there exists a non-expander graph $\mathcal{G}$ with minimum degree $\delta \geq 3$, and an adjacent vertex pair $(s,t) \in \mathcal{E}$, such that any local algorithm requires $\Omega(n)$ queries to ϵ-approximates $r_{\mathcal{G}}(s,t)$ for any $\epsilon \leq 0.01$. It can be easily proved that the condition number of their construction is $\kappa(\mathcal{L}) = \Theta(n)$, thus implies the $\Omega(\kappa(\mathcal{L}))$ lower bound for the ϵ-approximation of s,t-ER value, for a given $\epsilon \leq 0.01$. However, it remains an open problem to study the relationship between ϵ and $\kappa(\mathcal{L})$ for local algorithms, so as to unify the different cases for expanders and non-expanders (since for expanders $\kappa(\mathcal{L}) = \tilde{O}(1)$).

Applications of ER computation. Since ER approximation algorithms can be used as a crucial subroutine for other graph analysis algorithms, many related studies have focused on ER approximation and its applications. Among them, recent advancements in graph clustering and counting random spanning trees are closely related to our results. For instance, Alev et al. (2018) proposed the ER-based graph clustering algorithm in $\tilde{O}(nm)$ time with non-trivial theoretical guarantees. Chu et al. (2018) design an $\tilde{O}(m + n^{1.875}/\delta^{1.75})$ time algorithm that computes a $(1+\delta)$-approximation of the number of spanning trees. This result was later improved by Li and Sachdeva (2023) to $\tilde{O}(m + n^{1.5}/\delta)$, achieving optimality for determinant sparsifier-based methods in the case of expander graphs. These works leverage the ER sketch algorithm as a key subroutine. Beyond these

applications, ER computation has also been utilized in graph sparsification Spielman and Srivastava (2008) and robust routing in road networks Sinop et al. (2023), among others. The potential impact of our algorithms on enhancing the efficiency of these applications remains an open direction for future research.

2. Preliminaries

General matrix notations. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$, we use $\lambda(\mathbf{X})$ to denote its spectrum, $\lambda_i(\mathbf{X})$ represents the i-th eigenvalue of $\mathbf{X}$, $\mathbf{u}_i$ denotes the corresponding eigenvector. For a positive semi-definite (PSD) matrix $\mathbf{X}$, the condition number is defined as $\kappa(\mathbf{X}) = \frac{\lambda_{max}(\mathbf{X})}{\lambda_{min}(\mathbf{X})}$, where $\lambda_{max}(\mathbf{X})$ is the maximum eigenvalue of $\mathbf{X}$, $\lambda_{min}(\mathbf{X})$ is the minimum non-zero eigenvalue of $\mathbf{X}$. Note that for singular matrix, this represents the finite condition number instead of the original definition $\kappa(\mathbf{X}) = \infty$. Let $\mathbf{e}_s \in \mathbb{R}^n$ be the one-hot vector, which takes value 1 at $s \in [n]$ and 0 elsewhere.

Graph notations. Given an undirected, unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we denote $\mathcal{V}$ as the vertex set and $\mathcal{E}$ as the edge set. Let $|\mathcal{V}| = n$ be the number of vertices and $|\mathcal{E}| = m$ be the number of edges of $\mathcal{G}$. For a node $u \in \mathcal{V}$, the set of neighbors is denoted as $\mathcal{N}(u) = \{v|(u, v) \in \mathcal{E}\}$, and the degree of node u is denoted as $d_u = |\mathcal{N}(u)|$. Let $\mathbf{D}$ be the diagonal degree matrix, with diagonal entry $\mathbf{D}_{i,i} = d_i$ (the degree of node i). Denote by $\mathbf{A}$ the adjacency matrix with $\mathbf{A}_{i,j} = 1$ if and only if $(i, j) \in \mathcal{E}$. Then, the Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Let $\mathcal{A} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ be the normalized adjacency matrix, $\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$ be the probability transition matrix, and $\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I}-\mathcal{A}$ be the normalized Laplacian matrix. We denote $0 = \lambda_1(\mathcal{L}) < \lambda_2(\mathcal{L}) \leq ... \leq \lambda_n(\mathcal{L}) \leq 2$ as the eigenvalues of $\mathcal{L}$, and $1 = \lambda_1(\mathcal{A}) > \lambda_2(\mathcal{A}) \geq ... \geq \lambda_n(\mathcal{A}) \geq -1$ as the eigenvalues of $\mathcal{A}$. By definition, we have $\lambda_i(\mathcal{A}) = 1 - \lambda_i(\mathcal{L})$ for each $i \in [n]$. Following Dwaraknath et al. (2024); Cai et al. (2023), we define the expander graphs based on conductance.

Definition 6 *(Expander) We define the conductance of a graph $\mathcal{G}$ as:*

$$\phi_{\mathcal{G}} = \min_{S\subset\mathcal{V}} \frac{|\{(u, v) \in \mathcal{E} : u \in S, v \in \mathcal{V} - S\}|}{\min\{\sum_{u\in S} d_u, \sum_{u\in\mathcal{V}-S} d_u\}}.$$

We refer to the graph $\mathcal{G}$ as a expander if $\phi_{\mathcal{G}} = \tilde{\Omega}(1)$.

By the classical Cheeger's inequality (that is, $\lambda_2(\mathcal{L})/2 \leq \phi_{\mathcal{G}} \leq \sqrt{2\lambda_2(\mathcal{L})}$, see e.g. Spielman (2019) Chapter 21), we have $\phi_{\mathcal{G}} = \tilde{\Omega}(1)$ if and only if $\lambda_2(\mathcal{L}) = \tilde{\Omega}(1)$, where $\lambda_2(\mathcal{L})$ is the second smallest eigenvalue of the normalized Laplacian $\mathcal{L}$ (also the smallest non-zero eigenvalue). Consequently, the definition of an expander graph can be equivalently characterized by $\lambda_2(\mathcal{L}) = \tilde{\Omega}(1)$ or by the condition number $\kappa(\mathcal{L}) = \tilde{O}(1)$.

3. Basic representation

Following Peng et al. (2021); Li and Sachdeva (2023); Dwaraknath et al. (2024), we start by interpreting the ER value through the Taylor expansion of the Laplacian pseudo-inverse (but our interpretation is simpler and slightly differs from the previous works Peng et al. (2021); Li and Sachdeva (2023); Dwaraknath et al. (2024)). We begin by stating the following lemma, which expresses $r_{\mathcal{G}}(s, t)$ as the sum of infinite step of lazy random walks.

Lemma 7 (↓) *The following Equation holds*

$$\begin{aligned} r_{\mathcal{G}}(s,t) &= (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^{\dagger} (\mathbf{e}_s - \mathbf{e}_t) \\ &= \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1} \sum_{l=0}^{+\infty} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l (\mathbf{e}_s - \mathbf{e}_t) \end{aligned}$$

with $\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$ *denotes the probability transition matrix.*

By the Taylor expansion of $r_{\mathcal{G}}(s,t)$ from Lemma 7, we define the L-step truncated ER value $r_{\mathcal{G},L}(s,t)$ as follows.

Definition 8 *We define*

$$r_{\mathcal{G},L}(s,t) = \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1} \sum_{l=0}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l (\mathbf{e}_s - \mathbf{e}_t)$$

as the L step truncation ER value.

Next, we aim to ensure that $r_{\mathcal{G},L}(s,t)$ serves as an ϵ-approximation of $r_{\mathcal{G}}(s,t)$ by selecting a sufficiently large truncation step L. However, for expander graphs, we will prove that $L = \tilde{O}(1)$ is sufficient to achieve this approximation.

Lemma 9 (↓) *$r_{\mathcal{G},L}(s,t)$ is the ϵ-approximation of $r_{\mathcal{G}}(s,t)$ when setting $L \geq 2\kappa(\mathcal{L}) \log \frac{n}{\epsilon}$.*

By Lemma 9, the truncation step is only required to be $L = \tilde{O}(1)$, since $\kappa(\mathcal{L}) = \tilde{O}(1)$ for expander graphs. Inspired by Lemma 9, we define the following vector.

Definition 10 $\mathbf{p}_{L,u} = \frac{1}{2}\sum_{l=0}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l \mathbf{e}_u$ *for any given $u \in \mathcal{V}$.*

From Definition 8 and Definition 10, we have the following representation of $r_{\mathcal{G}}(s,t)$:

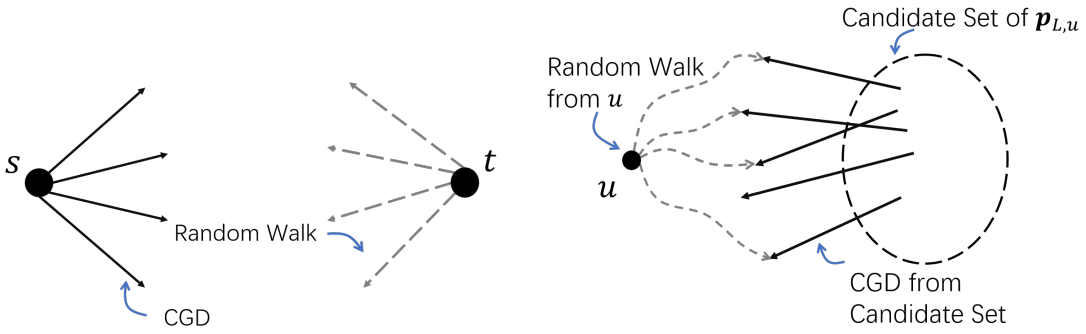
$$\begin{aligned} r_{\mathcal{G},L}(s,t) &= \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1} \sum_{l=0}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l (\mathbf{e}_s - \mathbf{e}_t) \\ &= \mathbf{e}_s^T \mathbf{D}^{-1} \mathbf{p}_{L,s} - \mathbf{e}_t^T \mathbf{D}^{-1} \mathbf{p}_{L,s} - \mathbf{e}_s^T \mathbf{D}^{-1} \mathbf{p}_{L,t} + \mathbf{e}_t^T \mathbf{D}^{-1} \mathbf{p}_{L,t} \\ &= \frac{\mathbf{p}_{L,s}(s)}{d_s} - \frac{\mathbf{p}_{L,s}(t)}{d_t} - \frac{\mathbf{p}_{L,t}(s)}{d_s} + \frac{\mathbf{p}_{L,t}(t)}{d_t}. \end{aligned} \tag{1}$$

Below, we give some basic properties of $\mathbf{p}_{L,u}$ and $r_{\mathcal{G}}(s,t)$ which will be frequently used in our analysis later.

Fact 1 (↓) $\|\mathbf{p}_{L,u}\|_1 \leq \frac{L}{2}$ *for any $u \in \mathcal{V}$.*

Fact 2 (↓) $\frac{\mathbf{p}_{L,u}(v)}{d_v} = \frac{\mathbf{p}_{L,v}(u)}{d_u}$ *for any $u, v \in \mathcal{V}$.*

Fact 3 (↓) $r_{\mathcal{G}}(s,t) \geq \frac{1}{2}\left(\frac{1}{d_s} + \frac{1}{d_t}\right)$.



(*a*) Our online-computation algorithm for computing $\mathbf{p}_{L,s}(t)$ (*b*) Our index-based algorithm for computing $\mathbf{p}_{L,u}$

Figure 1: Illustration of our algorithms for ER computation

4. Online ER Computation Algorithms

In this section, we present our online-computation algorithm for single-pair ER computation and prove Theorem 2. By combining Eq. (1) with Lemma 9, we only need to find the approximation $\hat{\mathbf{p}}_{L,u}(v)$ for each $u, v \in \{s,t\}$ to approximate the ER value $r_{\mathcal{G}}(s,t)$. The approximation is given by $\hat{r}_{\mathcal{G}}(s,t) = \frac{\hat{\mathbf{p}}_{L,s}(s)}{d_s} - \frac{\hat{\mathbf{p}}_{L,s}(t)}{d_t} - \frac{\hat{\mathbf{p}}_{L,t}(s)}{d_s} + \frac{\hat{\mathbf{p}}_{L,t}(t)}{d_t}$. Formally, we prove the following Theorem.

Theorem 11 (↓) *There exists an Algorithm (i.e. our Algorithm 1) outputs the approximation* $\hat{\mathbf{p}}_{L,u}(v)$ *of* $\mathbf{p}_{L,u}(v)$ *such that* $|\mathbf{p}_{L,u}(v) - \hat{\mathbf{p}}_{L,u}(v)|/d_v \leq \frac{\epsilon}{d}$ *for each* $u, v \in \{s,t\}$ *w.h.p., where* $d = \min\{d_s, d_t\}$. *Furthermore, the time complexity of this Algorithm can be bounded by* $\tilde{O}(\frac{\sqrt{d}}{\epsilon})$.

Next, we describe our algorithm to find the approximation $\hat{\mathbf{p}}_{L,u}(v)$ of $\mathbf{p}_{L,u}(v)$. The pseudo-code is outlined in Algorithm 1. Algorithm 1 consists of two stages:

- Stage I: we perform a deterministic search, called CGD, to obtain a coarse approximation of the vector $\mathbf{p}_{L,u}$, denote as $\mathbf{q}_{L,u}$. We output the residuals $\mathbf{r}_{i,u}$ for every $i \in [L]$.
- Stage II: we perform random walks to obtain the precise approximation of $\mathbf{p}_{L,s}(s)$ (resp., $\mathbf{p}_{L,s}(t), \mathbf{p}_{L,t}(s), \mathbf{p}_{L,t}(t)$), denote as $\hat{\mathbf{p}}_{L,s}(s)$ (resp., $\hat{\mathbf{p}}_{L,s}(t), \hat{\mathbf{p}}_{L,t}(s), \hat{\mathbf{p}}_{L,t}(t)$). Thus we obtain the ϵ-approximation of $r_{\mathcal{G}}(s,t)$.

As we discussed in Section 1, the high-level idea of our Algorithm is to use the deterministic search to reduce the variance produced by random walks, see Fig. 1 (a) as an illustration. More precisely, for Stage I we use a variant of the classical coordinate gradient descent algorithm to approximate $\mathbf{p}_{L,u}$. We denote $\mathbf{x}_{l,u} = \frac{1}{2}(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P})^l \mathbf{e}_u$, therefore $\mathbf{p}_{L,u} = \frac{1}{2}\sum_{l=0}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l \mathbf{e}_u = \sum_{l=0}^{L}\mathbf{x}_{l,u}$. Notice that $\mathbf{x}_{l+1,u} = (\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P})\mathbf{x}_{l,u}$ for each $l < L$, so computing $\mathbf{p}_{L,u}$ is equivalent to solving the following linear system:

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\frac{1}{2}\mathbf{I} - \frac{1}{2}\mathbf{P} & \mathbf{I} & & & \\ & \ddots & \ddots & & \\ & & & \mathbf{I} & \\ & & & -\frac{1}{2}\mathbf{I} - \frac{1}{2}\mathbf{P} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{0,u} \\ \mathbf{x}_{1,u} \\ \mathbf{x}_{2,u} \\ \vdots \\ \mathbf{x}_{L,u} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{e}_u \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Algorithm 1 Our algorithm for single pair ER estimation

Input: $\mathcal{G}, s, t, L, \epsilon$

1: $d \leftarrow \min\{d_s, d_t\}$; $r_{max} \leftarrow \epsilon/\sqrt{d}$ ▷ Stage I: deterministic search
2: $\mathbf{q}_{L,s}, \mathbf{r}_{i,s}$ for $i \in [L] \leftarrow \mathsf{CGD}\ (\mathcal{G}, s, L, r_{max})$
3: $\mathbf{q}_{L,t}, \mathbf{r}_{i,t}$ for $i \in [L] \leftarrow \mathsf{CGD}\ (\mathcal{G}, t, L, r_{max})$
4: $n_r \leftarrow \frac{4L^2}{\epsilon^2} r_{max} d \log n$ ▷ Stage II: random walk sampling
5: **for** j from 1 to n_r **do**
6: Perform lazy random walk of length L start from s. For each step k with $k \leq L$ it arrive at a node w, we update $\hat{\mathbf{p}}_{L,t}(s) \leftarrow \hat{\mathbf{p}}_{L,t}(s) + \frac{d_s}{n_r} \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,t}(w)}{d_w}$ and $\hat{\mathbf{p}}_{L,s}(s) \leftarrow \hat{\mathbf{p}}_{L,s}(s) + \frac{d_s}{n_r} \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,s}(w)}{d_w}$
7: Perform lazy random walk of length L start from t. For each step k with $k \leq L$ it arrive at a node w, we update $\hat{\mathbf{p}}_{L,s}(t) \leftarrow \hat{\mathbf{p}}_{L,s}(t) + \frac{d_t}{n_r} \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,s}(w)}{d_w}$ and $\hat{\mathbf{p}}_{L,t}(t) \leftarrow \hat{\mathbf{p}}_{L,t}(t) + \frac{d_t}{n_r} \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,t}(w)}{d_w}$
8: **end for**

Output: $\hat{r}_{\mathcal{G}}(s,t) = \frac{\hat{\mathbf{p}}_{L,s}(s)}{d_s} - \frac{\hat{\mathbf{p}}_{L,s}(t)}{d_t} - \frac{\hat{\mathbf{p}}_{L,t}(s)}{d_s} + \frac{\hat{\mathbf{p}}_{L,t}(t)}{d_t}$ as the approximation of $r_{\mathcal{G}}(s,t)$

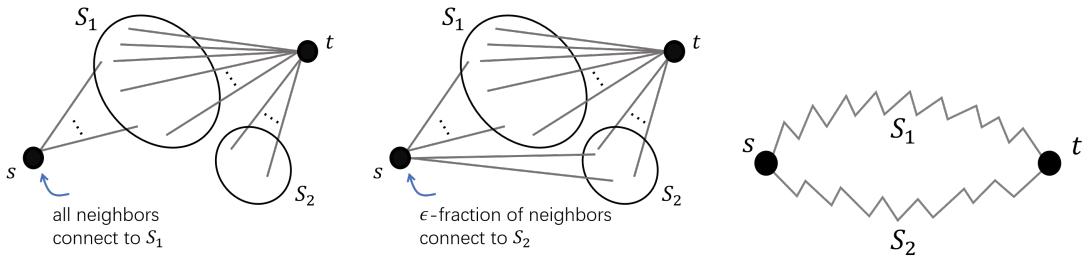
Algorithm 2 Coordinate Gradient Descent (CGD)

Input: $\mathcal{G}, u, L, r_{max}$

1: The approximation $\mathbf{q}_{L,u}(w) = 0$ for each $w \in \mathcal{V}$
2: $\mathbf{r}_{i,u}(w) = 0$ for each $w \in \mathcal{V}$, $i \in [L+1]$, and $\mathbf{r}_{0,u} = \frac{1}{2}\mathbf{e}_s$
3: **for** $i = 0, 1, 2, ..., L$ **do**
4: **while** $\exists w \in \mathcal{V}$ such that $\mathbf{r}_{i,u}(w) > \frac{r_{max}}{L^2} d_w$ **do**
5: $\mathbf{q}_{L,u}(w) \leftarrow \mathbf{q}_{L,u}(w) + \mathbf{r}_{i,u}(w)$
6: **for** $v \in \mathcal{N}(w)$ **do**
7: $\mathbf{r}_{i+1,u}(v) \leftarrow \mathbf{r}_{i+1,u}(v) + \frac{1}{2d_w}\mathbf{r}_{i,u}(w)$
8: **end for**
9: $\mathbf{r}_{i+1,u}(w) \leftarrow \mathbf{r}_{i+1,u}(w) + \frac{1}{2}\mathbf{r}_{i,u}(w)$
10: $\mathbf{r}_{i,u}(w) \leftarrow 0$
11: **end while**
12: **end for**

Output: $\mathbf{q}_{L,u}$ as the coarse approximation of $\mathbf{p}_{L,u}$, residuals $\mathbf{r}_{i,u}$ for each $i \in [L]$

For this linear system, we compute the approximation $\hat{\mathbf{x}}_{l,u}$ for each $l \in [L]$ and thus obtain the approximation $\mathbf{q}_{L,u} = \sum_{l=0}^{L} \hat{\mathbf{x}}_{l,u}$. For a given source node u, the Coordinate Gradient Descent (CGD) Algorithm outputs the approximation $\mathbf{q}_{L,u}$ and residuals $\mathbf{r}_{i,u}$ for every $i \in [L]$ (without explicitly outputs each approximation $\hat{\mathbf{x}}_{l,u}$). Initially, we define the residual vector $\mathbf{r}_{i,u} = 0$ for $i \in [1, ..., L]$ and $\mathbf{r}_{0,u} = \frac{1}{2}\mathbf{e}_u$ and the approximation $\mathbf{q}_{L,u} = 0$. Next, for each $i \in [0, ..., L]$, for each $w \in \mathcal{V}$ with $\mathbf{r}_{i,u}(w) > \frac{r_{max}}{L^2} d_w$, we perform the following three steps (Line 4-11 of Algorithm 2): (i) add the score $\mathbf{q}_{L,u}(w)$ by $\mathbf{r}_{i,u}(w)$; (ii) uniformly distribute $\frac{1}{2}\mathbf{r}_{i,u}(w)$ to the neighbor of w, namely $\mathbf{r}_{i+1,u}(v) \leftarrow \mathbf{r}_{i+1,u}(v) + \frac{1}{2d_w}\mathbf{r}_{i,u}(w)$ for $v \in \mathcal{N}(w)$ and $\mathbf{r}_{i+1,u}(w) \leftarrow \mathbf{r}_{i+1,u}(w) + \frac{1}{2}\mathbf{r}_{i,u}(w)$; (iii) set $\mathbf{r}_{i,u}(w)$ to 0. When the iteration stops, Algorithm 2 outputs $\mathbf{q}_{L,u}$ as the coarse approximation of $\mathbf{p}_{L,u}$, and L residual vectors $\mathbf{r}_{i,u}$ for each $i \in [L]$. For Stage II, we sample n_r lazy random walks from the source node s, each random walk has length L to refine the approximation (Line 6 of Algorithm 1). Specifically, a lazy random walk of length L from s denotes a sequence $(v_0 = s, v_1, ..., v_L)$ such that for every $1 \leq k \leq L$, v_k is a random neighbor of v_{k-1} with probability $\frac{1}{2}$, and $v_{k+1} = v_k$ with probability $\frac{1}{2}$. For each step $k \leq L$ when the lazy random walk arrives at a node $w = v_k$, we update $\hat{\mathbf{p}}_{L,t}(s) \leftarrow \hat{\mathbf{p}}_{L,t}(s) + \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,t}(w)}{d_w}$ and $\hat{\mathbf{p}}_{L,s}(s) \leftarrow \hat{\mathbf{p}}_{L,s}(s) + \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,s}(w)}{d_w}$. We perform similar steps for the sink node t (Line 7 of Algorithm 1).



(*a*) graph $\mathcal{G}_1$ (*b*) graph $\mathcal{G}_2$ (*c*) parallel resistor illustration

Figure 2: illustration of the construction of our lower bound

5. Lower Bound for Online Single-Pair ER Computation

In this section, we prove a lower bound for online single-pair ER computation on graphs. Below, we first describe our construction. Given the parameter $\epsilon \in (0,1)$, we consider two vertex sets S_1 and S_2, with $|S_1| = n_1$ and $|S_2| = n_2$. We define a d-regular expander graph on S_1. Note that this can be easily constructed because a random d-regular graph is an expander graph with high probability when $d \geq \log^2 n_1$, by Friedman (2003). On S_2, we define an empty graph with only n_2 isolated vertices and $n_2 \ll n_1$. Then, we add an additional (source) node s with d_s neighbors, such that $d_s \gg 1/\epsilon$. We randomly choose x vertices in S_1, denote as $\mathcal{N}_1 = \{v_1, ..., v_x\} \subset S_1$, and we randomly choose $(d_s - x)$ vertices in S_2, denote as $\mathcal{N}_2 = \{v_{x+1}, ..., v_{d_s}\} \subset S_2$. We set the neighbors of s to be $\mathcal{N}(s) = \{v_1, ..., v_{d_s}\} = \mathcal{N}_1 \cup \mathcal{N}_2$. For graph $\mathcal{G}_1$ we directly choose $x = d_s$, and for graph $\mathcal{G}_2$ we choose $x = (1-\epsilon)d_s$ for the given parameter $\epsilon \in (0,1)$. Finally, we add a (sink) node t that connect to all vertices in S_1 and S_2. We choose d such that $d \geq \max\{d_s \log n_1, \log^2 n_1\}$ and $n_1 \geq 2d^3$. See Fig. 2 (a),(b) as an illustration. Clearly, by our construction, any subset of $\mathcal{G}_1$ and $\mathcal{G}_2$ has $\Omega(1)$ expansion, so they are expanders. Then, we prove the following Theorem, which immediately implies Theorem 3.

Theorem 12 $|r_{\mathcal{G}_1}(s,t) - r_{\mathcal{G}_2}(s,t)| > \frac{\epsilon}{4} r_{\mathcal{G}_1}(s,t)$, *but any randomized local algorithm requires* $\Omega(\frac{1}{\epsilon})$ *time to distinguish* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *with probability* $\geq 2/3$.

First, to get intuitions for why $|r_{\mathcal{G}_1}(s,t) - r_{\mathcal{G}_2}(s,t)| > \frac{\epsilon}{4} r_{\mathcal{G}_1}(s,t)$, we use the representation of parallel resistors, see Fig. 2 (c) as an illustration. By our construction, we connect x neighbors of s to S_1 and connect $(d_s - x)$ neighbors of s to S_2. Then, we connect the two parts of the resistor S_1, S_2 in parallel. For $\mathcal{G}_1$ all the neighbors of s is in S_1; for $\mathcal{G}_2$ we choose $x = (1-\epsilon)d_s$, thereby there are $(1-\epsilon)$ fraction of the neighbors of s in S_1 and ϵ fraction of the neighbors in S_2. We prove the s,t resistance from S_1 is $\frac{1}{x} + o(\frac{1}{x})$; while the s,t resistance from S_2 is $\frac{2}{d_s - x}$. As a result, the ER value $r_{\mathcal{G}_1}(s,t)$ and $r_{\mathcal{G}_2}(s,t)$ will differ by $\frac{\epsilon}{4}$ in terms of relative error. To reach this end, we first prove the following Lemma.

Lemma 13 (↓) $r_{S_1}(s,t) = \frac{1}{x} + O(\frac{1}{dx}\log n_1) = \frac{1}{x} + O(\frac{1}{d_s x})$, *and* $r_{S_2}(s,t) = \frac{2}{d_s - x}$. *Where* $r_{S_1}(s,t)$ *denotes the* s,t *resistance only consider the edges between* $S_1 \cup \{s,t\}$ *and* $r_{S_2}(s,t)$ *denotes the* s,t *resistance only consider the edges between* $S_2 \cup \{s,t\}$.

Armed with Lemma 13 and the parallel resistance formula, we prove that $r_{\mathcal{G}_1}(s,t)$ and $r_{\mathcal{G}_2}(s,t)$ differs by $\frac{\epsilon}{4}$ in terms of relative error.

Lemma 14 (↓) $|r_{\mathcal{G}_1}(s,t) - r_{\mathcal{G}_2}(s,t)| > \frac{\epsilon}{4} r_{\mathcal{G}_1}(s,t)$.

Finally, we prove that any local algorithms to distinguish $\mathcal{G}_1$ and $\mathcal{G}_2$ requires at least $\Omega(\frac{1}{\epsilon})$ queries.

Lemma 15 (↓) *Any local algorithm that distinguish $\mathcal{G}_1$ and $\mathcal{G}_2$ with probability $\geq 2/3$ requires at least $\Omega(\frac{1}{\epsilon})$ queries.*

Combining Lemma 14 and Lemma 15, we immediately prove Theorem 12, which implies Theorem 3.

6. Index-based Algorithms for ER Computation

In this section, we present a more efficient index-based ER computation algorithm and prove Theorem 5. Note that by Theorem 2, if we directly use Algorithm 1 to compute the all pair ER values, the time complexity is bounded by:

$$\begin{aligned}
\tilde{O}\left(\sum_{u\in\mathcal{V}}\sum_{v\in\mathcal{V},d_v\leq d_u}\frac{\sqrt{d_v}}{\epsilon}\right) &\leq \tilde{O}\left(\sum_{u\in\mathcal{V}}\sum_{v\in\mathcal{V}}\frac{\sqrt{d_v}}{\epsilon}\right)\\
(Cauchy-Schwarz) &\leq \tilde{O}\left(\frac{1}{\epsilon}\sum_{u\in\mathcal{V}}\left(\left(\sum_{v\in\mathcal{V}}1^2\right)^{1/2}\left(\sum_{v\in\mathcal{V}}(\sqrt{d_v})^2\right)^{1/2}\right)\right)\\
&= \tilde{O}\left(\frac{1}{\epsilon}\sum_{u\in\mathcal{V}}\sqrt{nm}\right) = \tilde{O}\left(\frac{1}{\epsilon}n\sqrt{nm}\right).
\end{aligned}$$

In this section, we will prove that $\tilde{O}(\sqrt{nm}/\epsilon)$ processing time is enough for the ER sketch algorithms. Recall that by Eq. (1), given any vertex pair (s,t), one can approximate $r_{\mathcal{G}}(s,t)$ by the linear combination of $\mathbf{p}_{L,s}(s)$, $\mathbf{p}_{L,s}(t)$, $\mathbf{p}_{L,t}(s)$ and $\mathbf{p}_{L,t}(t)$. Therefore, if one can approximates these n vectors $\mathbf{p}_{L,u}$ for all $u\in\mathcal{V}$, one can approximate $r_{\mathcal{G}}(s,t)$ for any given vertex pair (s,t). We prove the following Theorem.

Theorem 16 (↓) *There exists an algorithm (i.e. Algorithm 3) that outputs the approximation $\hat{\mathbf{p}}_{L,u}$ for every $u\in\mathcal{V}$, such that $|\hat{\mathbf{p}}_{L,u}(v) - \mathbf{p}_{L,u}(v)| \leq \epsilon$ for every v with $d_v \leq d_u$. In addition, the time complexity of this algorithm is $\tilde{O}(\sqrt{nm}/\epsilon)$, the space complexity is $\tilde{O}(n/\epsilon)$.*

The key observation here is that we can approximate the vector $\mathbf{p}_{L,u}$ within almost the same time as just approximate $\mathbf{p}_{L,u}(v)$ for a vertex v. The pseudo-code of our ER sketch algorithm is outlined in Algorithm 3. To compute the approximation vector $\hat{\mathbf{p}}_{L,u}$, Algorithm 3 consists of three stages, see Fig. 1 (b) as an illustration.

- Stage I: perform $n_r^{(1)}$ random walks to find a candidate set, denote as $S'_{\epsilon,u} = \{w\in\mathcal{V} : \mathbf{p}'_{L,u}(w) \geq \epsilon/2\}$;
- Stage II: perform CGD operations from each $v\in S'_{\epsilon,u}$;
- Stage III: perform random walks start from u to derive the final approximation $\hat{\mathbf{p}}_{L,u}$.

Algorithm 3 Our algorithm for building the index

Input: $\mathcal{G}, s, t, L, \epsilon$

1: **for** $u \in \mathcal{V}$ **do**
2: $n_r^{(1)} \leftarrow \frac{L^2 \log n}{\epsilon}$, $\mathbf{p}'_{L,u} \leftarrow 0$ ▷ Stage I: find the ϵ-contributing set
3: Perform $n_r^{(1)}$ lazy random walks of length L start from u. For each lazy random walk, for each step k with $k \leq L$ it arrive at a node w, we update $\mathbf{p}'_{L,u}(w) \leftarrow \mathbf{p}'_{L,u}(w) + \frac{1}{2n_r^{(1)}}$
4: Define $S'_{\epsilon,u} = \{w \in \mathcal{V} : \mathbf{p}'_{L,u}(w) > \epsilon/2\}$ as the candidate set
5: $r_{max} \leftarrow L\epsilon\sqrt{\frac{n}{m}}$ ▷ Stage II: CGD from candidate set $S'_{\epsilon,u}$
6: **for** $v \in S'_{\epsilon,u}$ **do**
7: $\mathbf{q}_{L,v}, \mathbf{r}_{i,v}$ for $i \in [L] \leftarrow \mathsf{CGD}\ (\mathcal{G}, w, L, \frac{r_{max}}{2\mathbf{p}'_{L,u}(v)})$
8: **end for**
9: $n_r^{(3)} \leftarrow \frac{L}{\epsilon^2} r_{max} d_u \log n$ ▷ Stage III: random walk sampling from u
10: Perform $n_r^{(3)}$ lazy random walks of length L start from u, denote $N^{(3)}_{u,w,k}$ as the number of walks arrive at w at step k with $k \leq L$.
11: **for** $v \in S'_{\epsilon,u}$ with $d_v \leq d_u$ **do**
12: $\hat{\mathbf{p}}_{L,u}(v) \leftarrow \frac{\mathbf{q}_{L,v}(u) d_v}{d_u} + \frac{d_v}{n_r^{(3)}} \sum_{k=0}^{L} \sum_{w \in \mathcal{V}} \left(\sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,u}(w)}{d_w} \right) N^{(3)}_{u,w,k}$
13: **end for**
14: **end for**

Output: $\hat{\mathbf{p}}_{L,u}$ for every $u \in \mathcal{V}$

Algorithm 4 Our query algorithm for s, t-ER value

Input: $s, t, \hat{\mathbf{p}}_{L,u}$ for all $u \in \mathcal{V}$

1: **if** $d_s \leq d_t$ **then**
2: $\hat{r}_{\mathcal{G}}(s,t) = \frac{\hat{\mathbf{p}}_{L,s}(s)}{d_s} - 2\frac{\hat{\mathbf{p}}_{L,t}(s)}{d_s} + \frac{\hat{\mathbf{p}}_{L,t}(t)}{d_t}$
3: **else**
4: $\hat{r}_{\mathcal{G}}(s,t) = \frac{\hat{\mathbf{p}}_{L,s}(s)}{d_s} - 2\frac{\hat{\mathbf{p}}_{L,s}(t)}{d_t} + \frac{\hat{\mathbf{p}}_{L,t}(t)}{d_t}$
5: **end if**

Output: $\hat{r}_{\mathcal{G}}(s,t)$

Specifically, our algorithm is implemented as follows. First, we perform a traversal for all $u \in \mathcal{V}$. For each u, we compute the approximation $\hat{\mathbf{p}}_{L,u}$. For Stage I, we perform $n_r^{(1)}$ random walks from the source node u to compute a very coarse approximation $\mathbf{p}'_{L,u}$ (Lines 2-4 in Algorithm 3). We define the candidate set $S'_{\epsilon,u} = \{w \in \mathcal{V} : \mathbf{p}'_{L,u}(w) > \epsilon/2\}$ and next we focus on the approximation on the candidate set $S'_{\epsilon,u}$ (the size of the candidate set $S'_{\epsilon,u}$ is only $\tilde{O}(\frac{1}{\epsilon})$, independent of n). For Stage II, we perform CGD operations from each $v \in S'_{\epsilon,u}$ with the threshold $\frac{r_{max}}{2\mathbf{p}'_{L,u}(v)}$ (Lines 5-8 in Algorithm 3). For Stage III, we perform $n_r^{(3)}$ random walks from the source node u (Lines 9-10 in Algorithm 3). The key mechanism here is that the random walks can share the computation of CGD for different $\mathbf{p}_{L,u}(v_1)$ and $\mathbf{p}_{L,u}(v_2)$ with $v_1 \neq v_2$. Therefore, to approximate the vector $\mathbf{p}_{L,u}$ we only need to perform random walks from the source node u without the traversal for $w \in \mathcal{V}$. Finally, we update the result and get the final approximation $\hat{\mathbf{p}}_{L,u}(v)$ for every $v \in S'_{\epsilon,u}$ with $d_v \leq d_u$ (Lines 11-13 in Algorithm 3). For this step, the operation time is at most the same as Stage II, this enables the local computation of the vector $\mathbf{p}_{L,u}$.

7. Conclusion

In this paper, we propose several new algorithms for approximating Effective Resistance (ER) with reduced dependency on the error parameter ϵ. For online ER computation algorithms, we integrate deterministic search with random walk sampling to develop an $\tilde{O}(\sqrt{d}/\epsilon)$-time algorithm that ϵ-approximates the single-pair ER value in expander graphs. Additionally, we establish that $\Omega(1/\epsilon)$ represents the lower bound for local ER approximation, even in the case of expander graphs. For index-based ER computation algorithms, we extend our techniques and propose an ER sketch algorithm that advances the state-of-the-art. For the open problems, beyond the natural challenge of improving both upper and lower bounds, it would particularly be interested in exploring the relationship between $\kappa(\mathcal{L})$ and ϵ in the context of the lower bound for local ER computation, so as to unify the analysis for both expander and non-expander graphs.

Acknowledgments

This work is supported by the Funds of the National Natural Science Foundation of China (NFSC) No.U2241211 and U24A20255. Rong-Hua Li is the corresponding author of this paper.

References

Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. In *9th Innovations in Theoretical Computer Science Conference (ITCS)*, 2018.

Alexandr Andoni, Robert Krauthgamer, and Yosef Pogrow. On solving linear systems in sublinear time. In *10th Innovations in Theoretical Computer Science Conference (ITCS)*, 2019.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning (ICML)*, 2023.

Marco Bressan, Enoch Peserico, and Luca Pretto. Sublinear algorithms for local graph centrality estimation. In *59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2018.

Dongrun Cai, Xue Chen, and Pan Peng. Effective resistances in non-expander graphs. *31st Annual European Symposium on Algorithms (ESA)*, 2023.

Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. Graph sparsification, spectral sketches, and faster resistance computation via short cycle decompositions. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.

David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining (KDD)*, 2018.

David Durfee, Rasmus Kyng, John Peebles, Anup B. Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

Rajat Vadiraj Dwaraknath, Ishani Karmarkar, and Aaron Sidford. Towards optimal effective resistance estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kimon Fountoulakis, Farbod Roosta-Khorasani, Julian Shun, Xiang Cheng, and Michael W Mahoney. Variational perspective on local graph clustering. *Mathematical Programming*, 2019.

Kimon Fountoulakis, Di Wang, and Shenghao Yang. P-norm flow diffusion for local graph clustering. In *International Conference on Machine Learning (ICML)*, 2020.

Kimon Fountoulakis, Meng Liu, David F Gleich, and Michael W Mahoney. Flow-based algorithms for improving clusters: A unifying framework, software, and performance. *SIAM Review*, 2023.

Joel Friedman. A proof of alon's second eigenvalue conjecture. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC)*, 2003.

Arun Jambulapati and Aaron Sidford. Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the laplacian and its pseudoinverse. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018.

Yujia Jin, Christopher Musco, Aaron Sidford, and Apoorv Vikram Singh. Moments, random walks, and limits for spectrum approximation. In *The Thirty Sixth Annual Conference on Learning Theory (COLT)*, 2023.

Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016.

Lawrence Li and Sushant Sachdeva. A new approach to estimating effective resistances and counting spanning trees in expander graphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023.

Meihao Liao, Rong-Hua Li, Qiangqiang Dai, Hongyang Chen, Hongchao Qin, and Guoren Wang. Efficient resistance distance computation: The power of landmark-based approaches. *Proceedings of the ACM on Management of Data (SIGMOD)*, 2023.

Meng Liu and David F Gleich. Strongly local p-norm-cut algorithms for semi-supervised learning and local graph clustering. *Advances in neural information processing systems (NeurIPS)*, 2020.

Peter Lofgren and Ashish Goel. Personalized pagerank to a target node. *arXiv preprint arXiv:1304.4658*, 2013.

Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. Personalized pagerank estimation and search: A bidirectional approach. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*, 2016.

Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016.

Pan Peng, Daniel Lopatta, Yuichi Yoshida, and Gramoz Goranci. Local algorithms for estimating effective resistance. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, 2021.

Dana Ron. Sublinear-time algorithms for approximating graph parameters. In *Computing and Software Science: State of the Art and Perspectives*. 2019.

Sushant Sachdeva, Nisheeth K Vishnoi, et al. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 2014.

Shota Saito and Mark Herbster. Multi-class graph clustering via approximated effective p-resistance. In *International Conference on Machine Learning (ICML)*, 2023.

Ali Kemal Sinop, Lisa Fawcett, Sreenivas Gollapudi, and Kostas Kollias. Robust routing using electrical flows. *ACM Transactions on Spatial Algorithms and Systems*, 2023.

Spielman. *Spectral and Algebraic Graph Theory.* Unpublished Manuscript, 2019.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, 2008.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations (ICLR)*, 2022.

Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2022.

Hanzhi Wang, Zhewei Wei, Ji-Rong Wen, and Mingji Yang. Revisiting local computation of pagerank: Simple and optimal. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, 2024.

Zhewei Wei, Ji-Rong Wen, and Mingji Yang. Approximating single-source personalized pagerank with absolute error guarantees. In *27th International Conference on Database Theory (ICDT)*, 2024.

Renchi Yang and Jing Tang. Efficient estimation of pairwise effective resistance. *Proceedings of the ACM on Management of Data (SIGMOD)*, 2023.

Appendix A. Guideline of the Appendix

In Appendix B, we provide some explainations for the parallel resistance formula. In Appendix C, we provide the theoretical analysis of Algorithm 1 (i.e. our query processing algorithm for ER computation of a single pair) and prove Theorem 2. In Appendix D, we provide the theoretical analysis of Algorithm 3 (i.e. our index based algorithm for ER computation) and prove Theorem 5. In Appendix E, we provide the other omitting proofs of this paper, including the omitting proofs in section 3 and section 5.

Appendix B. Parallel Resistance Formula

For the proof of our lower bound, we provide some details of the parallel resistance formula. We begin by the definition of Schur complement.

Definition 17 *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, The Schur complement of a subset $S \subset \mathcal{V}$ is a weighted graph with the corresponding Laplacian matrix $\mathbf{SC}(\mathcal{G}, S) = \mathbf{L}_{[S,S]} - \mathbf{L}_{[S,\bar{S}]}\mathbf{L}^{-1}_{[\bar{S},\bar{S}]}\mathbf{L}_{[\bar{S},S]}$, where $\bar{S} = \mathcal{V} - S$ and $\mathbf{L}_{[S,S]}$ denotes the submatrix of $\mathbf{L}$ with row and column indexed by S.*

There is a well known result that the s,t-ER value on graph $\mathcal{G}$ is equal to the s,t-ER value on the Schur complement.

Theorem 18 *For a subset $S \subset \mathcal{V}$ such that $s,t \in S$, we have $r_{\mathcal{G}}(s,t) = (\mathbf{e}_s - \mathbf{e}_t)^T\mathbf{L}^{\dagger}(\mathbf{e}_s - \mathbf{e}_t) = (\mathbf{e}_s - \mathbf{e}_t)^T\mathbf{SC}(\mathcal{G}, S)^{\dagger}(\mathbf{e}_s - \mathbf{e}_t)$.*

Now we consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, such that $\mathcal{V} = \{s,t\} \sqcup S_1 \sqcup S_2$, and there are no edges between S_1 and S_2, i.e. $\mathcal{E}(S_1, S_2) = \emptyset$, and $(s,t) \notin \mathcal{E}$. We denote $\mathcal{G}_{S_1}$ as the induced subgraph on $\{s,t\} \sqcup S_1$, and $\mathcal{G}_{S_2}$ as the induced subgraph on $\{s,t\} \sqcup S_2$. We denote $r_{\mathcal{G}}(s,t)$ as the s,t-ER value on $\mathcal{G}$, $r_{S_1}(s,t)$ as the s,t-ER value on subgraph $\mathcal{G}_{S_1}$, and $r_{S_2}(s,t)$ as the s,t-ER value on subgraph $\mathcal{G}_{S_2}$, respectively. Then we provide the following parallel resistance formula.

Theorem 19 $\frac{1}{r_{\mathcal{G}}(s,t)} = \frac{1}{r_{S_1}(s,t)} + \frac{1}{r_{S_2}(s,t)}$.

Proof We prove this result by linear algebra argument instead of physics intuition. First, we consider the Schur complement of $\mathcal{G}_{S_1}$ on $\{s,t\}$, $\mathbf{SC}(\mathcal{G}_{S_1}, \{s,t\})$. We notice that this is a graph with single (weighted) edge between s,t. So we denote $\mathbf{SC}(\mathcal{G}_{S_1}, \{s,t\}) = w_{s,t}(\mathbf{e}_s - \mathbf{e}_t)(\mathbf{e}_s - \mathbf{e}_t)^T$. Similarly, we denote $\mathbf{SC}(\mathcal{G}_{S_2}, \{s,t\}) = w'_{s,t}(\mathbf{e}_s - \mathbf{e}_t)(\mathbf{e}_s - \mathbf{e}_t)^T$. Clearly, by definition $w_{s,t} = 1/r_{S_1}(s,t)$ and $w'_{s,t} = 1/r_{S_2}(s,t)$. Next, we consider the Schur complement of $\mathcal{G}$ on $\{s,t\}$, $\mathbf{SC}(\mathcal{G}, \{s,t\})$. By $\mathcal{E}(S_1, S_2) = \emptyset$ and the fact that taking the Schur complement is equivalent to the partial Gaussian elimination procedure Kyng and Sachdeva (2016), $\mathbf{SC}(\mathcal{G}, \{s,t\})$ is a graph with single edge between s,t with edge weight $w_{s,t} + w'_{s,t}$ (because the process of eliminating S_1 and eliminating S_2 are independent). By definition $w_{s,t} + w'_{s,t} = 1/r_{\mathcal{G}}(s,t)$. This finishes the proof. ■

By the same argument, we can generalize the parallel resistance formula to the k parallel resistance case. That is, when we consider $\mathcal{V} = \{s,t\} \sqcup S_1 \sqcup S_2 \sqcup \ldots \sqcup S_k$ with $\mathcal{E}(S_i, S_j) = \emptyset$, $i,j \in [k]$. Then $\frac{1}{r_{\mathcal{G}}(s,t)} = \sum_{i=1}^{k} \frac{1}{r_{S_i}(s,t)} + \mathbb{I}_{(s,t)\in\mathcal{E}}$, where $\mathbb{I}_{(s,t)\in\mathcal{E}}$ is the indicator function. Since the proof is almost same as Theorem 19, we omit it here.

Appendix C. Theoretical analysis of Algorithm 1

In this section, we prove the theoretical guarantee output by Algorithm 1. To this end, we begin by analyzing the output guarantee by CGD (Algorithm 2). The following Lemma shows the relationship between the approximation vector $\mathbf{q}_{L,u}$ output by CGD and the accurate vector $\mathbf{p}_{L,u}$.

Lemma 20 *The following equation holds at any iteration of Algorithm 2:*

$$\mathbf{p}_{L,u} = \mathbf{q}_{L,u} + \sum_{i=0}^{L}\sum_{k=i}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{r}_{i,u}. \tag{2}$$

Proof of Lemma 20. The proof is by induction. For the initial step, $\mathbf{r}_{0,u} = \frac{1}{2}\mathbf{e}_s$ and $\mathbf{r}_{i,u} = 0$ for each $i \in [1, 2, ..., L]$. Therefore the right hand side of Equ. (2) equals $\sum_{i=0}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i} \mathbf{r}_{0,u} = \mathbf{p}_{L,u}$. Next we assume that after j CGD operations, the approximation vector $\mathbf{q}_{L,u}$ and the residuals $\mathbf{r}_{0,u}, \mathbf{r}_{1,u}, ..., \mathbf{r}_{L,u}$ satisfy Equ. (2). For the $(j+1)^{th}$ CGD iteration, we consider the operation on (w, i). That is, we perform following three operations: (i) $\mathbf{q}_{L,u}(w) \leftarrow \mathbf{q}_{L,u}(w) + \mathbf{r}_{i,u}(w)$; (ii) $\mathbf{r}_{i+1,u}(v) \leftarrow \mathbf{r}_{i+1,u}(v) + \frac{1}{2d_w}\mathbf{r}_{i,u}(w)$ for each $v \in \mathcal{N}(w)$ and $\mathbf{r}_{i+1,u}(w) \leftarrow \mathbf{r}_{i+1,u}(w) + \frac{1}{2}\mathbf{r}_{i,u}(w)$; (iii) $\mathbf{r}_{i,u}(w) \leftarrow 0$. We define the resulting vector $\hat{\mathbf{q}}_{L,u}$ and $\hat{\mathbf{r}}_{0,u}, \hat{\mathbf{r}}_{1,u}, ..., \hat{\mathbf{r}}_{L,u}$ after this CGD operation, and consider the difference $\Delta(w, i)$ of the right hand side of Eq. (2):

$$\Delta(w,i) = \mathbf{q}_{L,u} + \sum_{i=0}^{L}\sum_{k=i}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{r}_{i,u} - \hat{\mathbf{q}}_{L,u} + \sum_{i=0}^{L}\sum_{k=i}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i}\hat{\mathbf{r}}_{i,u}.$$

We note the following three equations hold after the CGD operation on (w, i):

$$\begin{aligned}
\mathbf{q}_{L,u} - \hat{\mathbf{q}}_{L,u} &= \mathbf{r}_{i,u}(w)\mathbf{e}_w; \\
\mathbf{r}_{i,u} - \hat{\mathbf{r}}_{i,u} &= -\mathbf{r}_{i,u}(w)\mathbf{e}_w; \\
\mathbf{r}_{i+1,u} - \hat{\mathbf{r}}_{i+1,u} &= \frac{1}{2}\mathbf{r}_{i,u}(w)\mathbf{e}_w + \sum_{v\in\mathcal{N}(w)}\frac{1}{2d_w}\mathbf{r}_{i,u}(w)\mathbf{e}_v \\
&= \mathbf{r}_{i,u}(w)\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)\mathbf{e}_w.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Delta(w,i) &= \mathbf{r}_{i,u}(w)\mathbf{e}_w - \mathbf{r}_{i,u}(w)\sum_{k=i}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{e}_w \\
&\quad + \mathbf{r}_{i,u}(w)\sum_{k=i+1}^{L}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i}\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)\mathbf{e}_w \\
&= \mathbf{r}_{i,u}(w) - \mathbf{r}_{i,u}(w) = 0.
\end{aligned}$$

Thus Eq. (2) also holds after $(j+1)^{th}$ CGD operation. By induction, Eq. (2) holds at any iteration of Algorithm 2. This finishes the proof. ■

Next, we show that CGD outputs the approximation vector $\mathbf{q}_{L,u}$ satisfies the certain error guarantee.

Lemma 21 CGD *outputs the approximation* $\mathbf{q}_{L,u}$ *such that* $\mathbf{p}_{L,u}(w) - r_{max}d_w \leq \mathbf{q}_{L,u}(w) \leq \mathbf{p}_{L,u}(w)$ *for any* $w \in \mathcal{V}$. *Furthermore, the time complexity of* CGD *is* $O(\frac{L^3}{r_{max}})$.

Proof of Lemma 21. By Lemma 20, Eq. (2) holds at any iteration of CGD. By our implementation, the residuals satisfy $\mathbf{r}_{i,u}(w) \leq \frac{r_{max} d_w}{L^2}$ for any $w \in \mathcal{V}$, $i \in [L]$. Therefore, we have:

$$\begin{aligned}
\|\mathbf{D}^{-1}(\mathbf{p}_{L,u} - \mathbf{q}_{L,u})\|_\infty &= \left\| \mathbf{D}^{-1} \sum_{i=0}^{L} \sum_{k=i}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i} \mathbf{r}_{i,u} \right\|_\infty \\
&\leq \sum_{i=0}^{L} \sum_{k=i}^{L} \left\| \mathbf{D}^{-1} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i} \mathbf{r}_{i,u} \right\|_\infty \\
&= \sum_{i=0}^{L} \sum_{k=i}^{L} \left\| \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}^T\right)^{k-i} \mathbf{D}^{-1}\mathbf{r}_{i,u} \right\|_\infty \\
&\leq \sum_{i=0}^{L} \sum_{k=i}^{L} \left\| \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}^T\right)^{k-i} \right\|_\infty \left\| \mathbf{D}^{-1}\mathbf{r}_{i,u} \right\|_\infty.
\end{aligned}$$

Next, we notice that $\left\| \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}^T\right)^{k-i} \right\|_\infty = 1$ since $\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}^T\right)^{k-i}$ is a row-stochastic matrix and $\|\mathbf{D}^{-1}\mathbf{r}_{i,u}\|_\infty \leq \frac{r_{max}}{L^2}$. Therefore,

$$\|\mathbf{D}^{-1}(\mathbf{p}_{L,u} - \mathbf{q}_{L,u})\|_\infty \leq \sum_{i=0}^{L} \sum_{k=i}^{L} \frac{r_{max}}{L^2} \leq r_{max}.$$

Thus $\mathbf{p}_{L,u}(w) - r_{max} d_w \leq \mathbf{q}_{L,u}(w) \leq \mathbf{p}_{L,u}(w) + r_{max} d_w$ for any $w \in \mathcal{V}$. Furthermore, we notice that $\mathbf{r}_{i,u}(w) \geq 0$ for $\forall w \in \mathcal{V}$, $i \in [L]$. By Eq. (2), we obtain $\mathbf{q}_{L,u}(w) \leq \mathbf{p}_{L,u}(w)$ for $\forall w \in \mathcal{V}$. Putting it together, we have $\mathbf{p}_{L,u}(w) - r_{max} d_w \leq \mathbf{q}_{L,u}(w) \leq \mathbf{p}_{L,u}(w)$.

For the time complexity of CGD, we observe that at the beginning of each iteration (Line 4 in Algorithm 2), $\|\mathbf{r}_{i,u}\|_1 \leq 1$, since $\|\mathbf{r}_{i,u}\|_1$ is non-increasing with i. Next, we only invoke nodes $w \in \mathcal{V}$ with $\mathbf{r}_{i,u}(w) > \frac{r_{max} d_w}{L^2}$, for each w the operation numbers is $O(d_w)$ (Line 5-9 in Algorithm 2). Finally, we set $\mathbf{r}_{i,u}(w) \leftarrow 0$ after the operation on w (Line 10 in Algorithm 2). Thus, one unit operation will decrease $\|\mathbf{r}_{i,u}\|_1$ by at least $\frac{r_{max}}{L^2}$ in average. So the total operations of CGD in each iteration (Line 4-11 in Algorithm 2) can be bounded by $O(\frac{L^2}{r_{max}})$. Since Algorithm 2 has L iterations, the total time complexity is bounded by $O(\frac{L^3}{r_{max}})$. ■

By Lemma 21, when setting $r_{max} = \frac{\epsilon}{d}$, the approximation guarantee by CGD already satisfies Theorem 11 and thus this is an algorithm for ϵ-approximate ER estimation. However, the time complexity of CGD is $O(\frac{L^3}{r_{max}}) = \tilde{O}(\frac{d}{\epsilon})$. Next we will prove that combining with random walk sampling (i.e. Stage II in algorithm 1), this time complexity can be reduced to $\tilde{O}(\frac{\sqrt{d}}{\epsilon})$ while maintaining the same error guarantee. To this end, we consider again the output guarantee of CGD for a given node $v \in \mathcal{V}$:

$$\mathbf{p}_{L,u}(v) = \mathbf{q}_{L,u}(v) + \sum_{i=0}^{L} \sum_{k=i}^{L} \left(\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^{k-i} \mathbf{r}_{i,u} \right)(v).$$

We divide d_v on both sides of the equation:

$$\begin{aligned}
\frac{\mathbf{p}_{L,u}(v)}{d_v} &= \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{i=0}^{L}\sum_{k=i}^{L}\left(\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{r}_{i,u}\right)(v)/d_v \\
&= \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{i=0}^{L}\sum_{k=i}^{L}\sum_{w\in\mathcal{V}}\frac{\mathbf{r}_{i,u}(w)}{d_v}\mathbf{e}_v^T\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{e}_w \\
&= \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{i=0}^{L}\sum_{k=i}^{L}\sum_{w\in\mathcal{V}}\frac{\mathbf{r}_{i,u}(w)}{d_w}\mathbf{e}_w^T\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^{k-i}\mathbf{e}_v \\
&= \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{i=0}^{L}\sum_{k=0}^{L-i}\sum_{w\in\mathcal{V}}\frac{\mathbf{r}_{i,u}(w)}{d_w}\mathbf{e}_w^T\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^{k}\mathbf{e}_v \\
&= \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{k=0}^{L}\sum_{w\in\mathcal{V}}\left(\sum_{i=0}^{L-k}\frac{\mathbf{r}_{i,u}(w)}{d_w}\right)\mathbf{e}_w^T\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^{k}\mathbf{e}_v.
\end{aligned} \tag{3}$$

We observe that $\mathbf{e}_w^T\left(\frac{1}{2}\mathbf{I}+\frac{1}{2}\mathbf{P}\right)^k\mathbf{e}_v$ is the probability that a k-step lazy random walk start from v and ends at w. Thus we define a sequence of random variables output by lazy random walks: $\{X_{v,k,j}\}$ with $v\in\{s,t\}, k\in[L]$ and $j\in[n_r]$. Each $X_{v,k,j}$ represents the j^{th} sample of random walk, start from node v, and the walk length is k. If this k step lazy random walk arrive at w, then $X_{v,k,j}=\sum_{i=0}^{L-k}\frac{\mathbf{r}_{i,u}(w)}{d_w}$. We denote $\bar{X}_{v,k}=\frac{1}{n_r}\sum_{j=1}^{n_r}X_{v,k,j}$ as the average of these random variables. So by Eq. (3), clearly we have:

$$\frac{\mathbf{p}_{L,u}(v)}{d_v} = \frac{\mathbf{q}_{L,u}(v)}{d_v} + \sum_{k=0}^{L}\mathbb{E}[\bar{X}_{v,k}]. \tag{4}$$

This immedeately proves that Algorithm 1 outputs the unbiased approximation of $\mathbf{p}_{L,u}(v)$. The next Lemma proves the concentration of this approximation.

Lemma 22 *When setting $n_r=\tilde{O}(\frac{L^2}{\epsilon^2}r_{max}d)$, we have $\left|\bar{X}_{v,k}-\mathbb{E}[\bar{X}_{v,k}]\right|\le\frac{\epsilon}{Ld}$ with probability at least $1-n^{-1}$, for every $v\in\{s,t\}$ and $k\in[L]$.*

Proof of Lemma 22. We observe that by our implementation of CGD, $\mathbf{r}_{i,u}(w)\le\frac{r_{max}d_w}{L^2}$ for any $w\in\mathcal{V}, i\in[L]$. Thus, for each random variable $|X_{v,k,j}|\le\max_{w,k}\left\{\sum_{i=0}^{L-k}\frac{\mathbf{r}_{i,u}(w)}{d_w}\right\}\le\frac{r_{max}}{L}$. Next, we note that for fixed k,v, the random variables $\{X_{v,k,j}\}$ are i.i.d since we sample random walks independently. Thus,

$$\begin{aligned}
Var[\bar{X}_{v,k}] &= \frac{1}{n_r}Var[X_{v,k,j}] \le \frac{1}{n_r}\mathbb{E}[X_{v,k,j}^2] \\
&\le \frac{r_{max}}{Ln_r}\mathbb{E}[X_{v,k,j}] \le \frac{r_{max}}{Ln_r}\frac{\mathbf{p}_{L,u}(v)}{d_v} \\
&\le \frac{r_{max}}{Ln_r}\frac{\|\mathbf{p}_{L,u}\|_1}{d_v} \le \frac{r_{max}}{n_r}\frac{1}{d_v} \le \frac{\epsilon^2}{4L^2d^2}.
\end{aligned}$$

The final inequality holds when setting $n_r = \frac{4L^2}{\epsilon^2} r_{max} d$, since $d = \min\{d_s, d_t\}$ and $v \in \{s, t\}$. Next, by Chebyshev's inequality,

$$\mathbb{P}\left[\left|\bar{X}_{v,k} - \mathbb{E}[\bar{X}_{v,k}]\right| \geq \frac{\epsilon}{Ld}\right] \leq \frac{1}{2}.$$

Finally, by the standard median of the means estimator, repeat the sampling process $\log n$ times and take the median as the output, we can make the final output satisfies $\left|\bar{X}_{v,k} - \mathbb{E}[\bar{X}_{v,k}]\right| \leq \frac{\epsilon}{Ld}$ with probability at least $1 - n^{-\Omega(1)}$. Putting these things together, the total sampling times $n_r = \frac{4L^2}{\epsilon^2} r_{max} d \log n = \tilde{O}(\frac{L^2}{\epsilon^2} r_{max} d)$ is enough. ■

Armed with Lemma 22, we are now ready to prove Theorem 11.

Proof of Theorem 11. From Lemma 22, for each $k \in [L]$, the inequality $\left|\bar{X}_{v,k} - \mathbb{E}[\bar{X}_{v,k}]\right| \leq \frac{\epsilon}{Ld}$ with probability at least $1 - n^{-1}$. By Eq. (4), the error produced by Algorithm 1 can be bounded by:

$$\left|\frac{\mathbf{p}_{L,u}(v)}{d_v} - \frac{\hat{\mathbf{p}}_{L,u}(v)}{d_v}\right| \leq \sum_{k=0}^{L} \left|\mathbb{E}[\bar{X}_{v,k}] - X_{v,k}\right| \leq \sum_{k=0}^{L} \frac{\epsilon}{Ld} \leq \frac{\epsilon}{d}.$$

with probability at least $1 - Ln^{-1} = 1 - \tilde{O}(n^{-1})$. Next, by Lemma 21, the time complexity for Phase I is $O(\frac{L^3}{r_{max}})$. For the time complexity of Phase II, since we set the number of random walks $n_r = \tilde{O}(\frac{L^2}{\epsilon^2} r_{max} d)$, for each random walk the length is L. So the time complexity of Phase II is $O(n_r L) = \tilde{O}(\frac{L^3 r_{max} d}{\epsilon^2})$. Finally, to balance the time complexity for Phase I and Phase II, we set $r_{max} = \frac{\epsilon}{\sqrt{d}}$. So the total time complexity of Algorithm 1 can be bounded by $\tilde{O}(\frac{L^3\sqrt{d}}{\epsilon}) = \tilde{O}(\frac{\sqrt{d}}{\epsilon})$. This finishes the proof of Theorem 11. ■

Finally, we use Theorem 11 to prove Theorem 2.

Proof of Theorem 2. By Equ. (1) and Theorem 11, the approximation output by Algorithm 1 satisfy: $|\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq \frac{4\epsilon}{d}$ w.h.p. By the fact $r_{\mathcal{G}}(s,t) \geq \frac{1}{2}\left(\frac{1}{d_s} + \frac{1}{d_t}\right) \geq \frac{1}{2d}$, so $|\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq 8\epsilon r_{\mathcal{G}}(s,t)$ w.h.p.. In addition, by Lemma 9, The L step truncation ER satisfy $|r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq \epsilon r_{\mathcal{G}}(s,t)$, so the approximation $\hat{r}_{\mathcal{G}}(s,t)$ safisfy:

$$\begin{aligned}|\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G}}(s,t)| &\leq |\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| + |r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \\ &\leq 8\epsilon r_{\mathcal{G}}(s,t) + \epsilon r_{\mathcal{G}}(s,t) = 9\epsilon r_{\mathcal{G}}(s,t).\end{aligned}$$

This proves that $\hat{r}_{\mathcal{G}}(s,t)$ is the 9ϵ-approximation of $r_{\mathcal{G}}(s,t)$ w.h.p. Finally, by Theorem 11, the time complexity of Algorithm 1 is $\tilde{O}(\frac{\sqrt{d}}{\epsilon})$. ■

Appendix D. Theoretical analysis of Algorithm 3

In this section, we provide the theoretical analysis of Algorithm 3, and prove Theorem 16 and Theorem 5. First, inspired by Wei et al. (2024) and Li and Sachdeva (2023), we define the ϵ-contributing set for $\mathbf{p}_{L,u}$.

Definition 23 *for any $u \in \mathcal{V}$, the ϵ-contributing set of $\mathbf{p}_{L,u}$ is defined as $S_{\epsilon,u} = \{w \in \mathcal{V} : \mathbf{p}_{L,u}(w) > \epsilon\}$.*

Next, we observe that the size of ϵ-contributing set is only $\tilde{O}(\frac{1}{\epsilon})$, independent of n. This immediately follows by Fact 1.

Fact 4 *The size of the ϵ-contributing set $|S_{\epsilon,u}| \leq \frac{L}{2\epsilon} = \tilde{O}(\frac{1}{\epsilon})$.*

Since in Theorem 16 we just need to approximate the vector such that $|\hat{\mathbf{p}}_{L,u}(w) - \mathbf{p}_{L,u}(w)| \leq \epsilon$, so we only focus on the ϵ-contributing set $S_{\epsilon,u}$. This is because if a node $w \notin S_{\epsilon,u}$, by definition $0 \leq \mathbf{p}_{L,u}(w) \leq \epsilon$, so we set $\hat{\mathbf{p}}_{L,u}(w) = 0$ is already a reasonable approximation. Next, we will prove Stage I of Algorithm 3 outputs the candidate set $S'_{\epsilon,u}$ that contains $S_{\epsilon,u}$ with high probability.

Lemma 24 *$\mathbf{p}'_{L,u}$ output by Stage I satisfy the following error guarantee:*

- *(i) For any $\mathbf{p}_{L,u}(w) > \epsilon$, we have $|\mathbf{p}'_{L,u}(w) - \mathbf{p}_{L,u}(w)| \leq \frac{1}{4}\mathbf{p}_{L,u}(w)$ with probability at least $1 - n^{-\Omega(1)}$.*
- *(ii) For any $\mathbf{p}_{L,u}(w) \leq \epsilon$, we have $|\mathbf{p}'_{L,u}(w) - \mathbf{p}_{L,u}(w)| \leq \frac{1}{4}\epsilon$ with probability at least $1 - n^{-\Omega(1)}$.*

Furthermore, $S'_{\epsilon,u} \supset S_{\epsilon,u}$ with probability at least $1 - n^{-\Omega(1)}$.

Proof of Lemma 24. We define a sequence of random variables output by lazy random walks: $\{X^{(1)}_{u,w,k,j}\}$ with source node u and target node w, and $k \in [L]$ and $j \in [n_r^{(1)}]$. Each $X^{(1)}_{u,w,k,j}$ represents the j^{th} sample of random walk, start from node u with k steps. If this k step random walk reach w, then $X^{(1)}_{u,w,k,j} = 1$, else $X^{(1)}_{u,w,k,j} = 0$. We denote $\bar{X}^{(1)}_{u,w,k} = \frac{1}{n_r}\sum_{j=1}^{n_r^{(1)}} X^{(1)}_{u,w,k,j}$ as the average of these random variables. Thus $\mathbb{E}[\bar{X}^{(1)}_{u,w,k}] = \mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^k \mathbf{e}_u$ and $Var[\bar{X}^{(1)}_{u,w,k}] \leq \frac{1}{n_r^{(1)}} Var[X^{(1)}_{u,w,k,j}] \leq \frac{1}{n_r^{(1)}}\mathbb{E}[(X^{(1)}_{u,w,k,j})^2] = \frac{1}{n_r^{(1)}}\mathbb{E}[X^{(1)}_{u,w,k,j}]$. We set $n_r^{(1)} = \frac{128L^2}{\epsilon}$. For $\mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^k \mathbf{e}_u > \frac{\epsilon}{L}$, $k \in [L]$, by the Chebyshev's inequality,

$$\mathbb{P}\left[\left|\bar{X}^{(1)}_{u,w,k} - \mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right]\right| \geq \frac{1}{8}\mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right]\right] \leq \frac{1}{2}.$$

For $\mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^k \mathbf{e}_u \leq \frac{\epsilon}{L}$, $k \in [L]$, by the Chebyshev's inequality,

$$\mathbb{P}\left[\left|\bar{X}^{(1)}_{u,w,k} - \mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right]\right| \geq \frac{\epsilon}{8L}\right] \leq \frac{1}{2}.$$

Again, by the standard median of means estimator, repeating the sampling process $O(\log n)$ times and take the median, this probability can be improved to less than $n^{-\Omega(1)}$. Putting it together, the total sampling times $n_r^{(1)} = O(\frac{L^2 \log n}{\epsilon})$ is enough. Finally, since $\mathbf{p}_{L,u}(w) = \frac{1}{2}\sum_{k=0}^{L} \mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^k \mathbf{e}_u = \frac{1}{2}\sum_{k=0}^{L} \mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right]$, we define the approximation $\mathbf{p}'_{L,u}(w) = \frac{1}{2}\sum_{k=0}^{L} \bar{X}^{(1)}_{u,w,k}$ (This is equivalent to the process of Line 4 in Algorithm 3). Thus, the approximation $\hat{\mathbf{p}}_{L,u}$ satisfy the following two conditions:

- (i) For any $\mathbf{p}_{L,u}(w) > \epsilon$, we have

$$|\mathbf{p}'_{L,u}(w) - \mathbf{p}_{L,u}(w)| \leq \sum_{k=0}^{L} \left|\bar{X}^{(1)}_{u,w,k} - \mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right]\right| \leq \frac{1}{8}\mathbf{p}_{L,u}(w) + \frac{\epsilon}{8} \leq \frac{1}{4}\mathbf{p}_{L,u}(w)$$

  with probability at least $1 - Ln^{-\Omega(1)} = 1 - n^{-\Omega(1)}$.

- (ii) For any $\mathbf{p}_{L,u}(w) \leq \epsilon$, we have

$$|\mathbf{p}'_{L,u}(w) - \mathbf{p}_{L,u}(w)| \leq \sum_{k=0}^{L} \left| \bar{X}^{(1)}_{u,w,k} - \mathbb{E}\left[\bar{X}^{(1)}_{u,w,k}\right] \right| \leq \frac{1}{8}\mathbf{p}_{L,u}(w) + \frac{\epsilon}{8} \leq \frac{1}{4}\epsilon$$

with probability at least $1 - Ln^{-\Omega(1)} = 1 - n^{-\Omega(1)}$.

As a result, $S'_{\epsilon,u} = \{w \in \mathcal{V} : \mathbf{p}'_{L,u}(w) > \epsilon/2\}$ contains $S_{\epsilon,u}$ with probability at least $1 - n^{-\Omega(1)}$. ■

Next, we perform CGD operations for each node $v \in S'_{\epsilon,u}$. For each v, we set the threshold for CGD to be $\frac{r_{max}}{2\mathbf{p}'_{L,u}(v)}$. By Lemma 20 and Eq. (3), we have the following equation:

$$\frac{\mathbf{p}_{L,v}(u)}{d_u} = \frac{\mathbf{q}_{L,v}(u)}{d_u} + \sum_{k=0}^{L}\sum_{w\in\mathcal{V}} \left(\sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,v}(w)}{d_w} \right) \mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^k \mathbf{e}_u. \tag{5}$$

By the fact $\frac{\mathbf{p}_{L,v}(u)}{d_u} = \frac{\mathbf{p}_{L,u}(v)}{d_v}$, this is equivalent to

$$\mathbf{p}_{L,u}(v) = \frac{\mathbf{q}_{L,v}(u)d_v}{d_u} + d_v \sum_{k=0}^{L}\sum_{w\in\mathcal{V}} \left(\sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,v}(w)}{d_w} \right) \mathbf{e}_w^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^k \mathbf{e}_u. \tag{6}$$

This enables us to approximate $\mathbf{p}_{L,u}(v)$ for any $v \in S'_{\epsilon,u}$ by random walks from just a source node u, see Fig. 1 (b) as an illustration. Similar to the proof for Lemma 22 and Lemma 24, we define a sequence of random variables: $\{X^{(3)}_{u,k,j}\}$. Each $X^{(3)}_{u,k,j}$ represents the j^{th} sample of random walk, start from node u with k steps. If this k step random walk reach w, then we set $X^{(3)}_{u,k,j} = \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,v}(w)}{d_w}$. We denote $\bar{X}^{(3)}_{u,k} = \frac{1}{n_r^{(3)}} \sum_{j=1}^{n_r^{(3)}} X^{(3)}_{u,k,j}$ as the average of these random variables. For each $v \in S'_{\epsilon,u}$, clearly we have

$$\mathbf{p}_{L,u}(v) = \frac{\mathbf{q}_{L,v}(u)d_v}{d_u} + d_v \sum_{k=0}^{L} \mathbb{E}[\bar{X}^{(3)}_{u,k}]. \tag{7}$$

We define the approximation as

$$\hat{\mathbf{p}}_{L,u}(v) = \frac{\mathbf{q}_{L,v}(u)d_v}{d_u} + d_v \sum_{k=0}^{L} \bar{X}^{(3)}_{u,k}. \tag{8}$$

This is equivalent to the expression of Line 12 in Algorithm 3. Next, we prove the concentration result of $\hat{\mathbf{p}}_{L,u}$.

Lemma 25 *When setting* $n_r^{(3)} = \tilde{O}(\frac{L}{\epsilon^2} r_{max} d_u)$, *the approximation* $\hat{\mathbf{p}}_{L,u}$ *satisfy:* $|\hat{\mathbf{p}}_{L,u}(v) - \mathbf{p}_{L,u}(v)| \leq \epsilon$ *with probability at least* $1 - n^{-\Omega(1)}$ *for any* $v \in S'_{\epsilon,u}$ *with* $d_v \leq d_u$.

Proof of Lemma 25. we set the threshold for CGD $\frac{r_{max}}{2\mathbf{p}'_{L,u}(v)}$, thus $r_{i,v}(w) \leq \frac{r_{max}d_w}{2L^2\mathbf{p}'_{L,u}(v)}$ for any $w \in \mathcal{V}$ and $i \leq L$. For each random variable $X^{(3)}_{u,k,j}$, we have $|X^{(3)}_{u,k,j}| \leq \max_{w,k} \left\{ \sum_{i=0}^{L-k} \frac{\mathbf{r}_{i,v}(w)}{d_w} \right\} \leq$

$\frac{r_{max}}{2L\mathbf{p}'_{L,u}(v)}$. Therefore,

$$\begin{aligned} Var[\bar{X}^{(3)}_{u,k}] &= \frac{1}{n_r} Var[X^{(3)}_{u,k,j}] \leq \frac{1}{n_r}\mathbb{E}[(X^{(3)}_{u,k,j})^2] \\ &\leq \frac{r_{max}}{2Ln_r\mathbf{p}'_{L,u}(v)}\mathbb{E}[X^{(3)}_{u,k,j}] \leq \frac{r_{max}}{2Ln_r\mathbf{p}'_{L,u}(v)}\frac{\mathbf{p}_{L,u}(v)}{d_v} \\ &\leq \frac{r_{max}}{Ln_r}\frac{1}{d_v} \leq \frac{\epsilon^2}{4L^2 d_u d_v}, \end{aligned}$$

where the final inequality holds when setting $n_r^{(3)} = \frac{4L}{\epsilon^2} r_{max} d_u$. By $d_v \leq d_u$, we obtain $Var[\bar{X}^{(3)}_{u,k}] \leq \frac{\epsilon^2}{4L^2 d_v^2}$, which is equivalent to $Var[d_v \bar{X}^{(3)}_{u,k}] \leq \frac{\epsilon^2}{4L^2}$. By Chebyshev's inequality,

$$\mathbb{P}[|d_v \bar{X}^{(3)}_{u,k} - \mathbb{E}[d_v \bar{X}^{(3)}_{u,k}]| \geq \frac{\epsilon}{L}] \leq \frac{1}{2}. \tag{9}$$

Again, by the standard median of means estimator, we repeat the sampling process $O(\log n)$ times to improve this probability to less than $n^{-\Omega(1)}$. The total sampling times is $n_r^{(3)} = O(\frac{L}{\epsilon^2} r_{max} d_u \log n) = \tilde{O}(\frac{L}{\epsilon^2} r_{max} d_u)$. Since the approximation is defined as $\hat{\mathbf{p}}_{L,u}(v) = \frac{\mathbf{q}_{L,v}(u) d_v}{d_u} + d_v \sum_{k=0}^{L} \bar{X}^{(3)}_{u,k}$, by Equ. (9),

$$|\hat{\mathbf{p}}_{L,u}(v) - \mathbf{p}_{L,u}(v)| \leq \sum_{k=0}^{L} |d_v \bar{X}^{(3)}_{u,k} - \mathbb{E}[d_v \bar{X}^{(3)}_{u,k}]| \leq \sum_{k=0}^{L} \frac{\epsilon}{L} \leq \epsilon$$

with probability at least $1 - Ln^{-\Omega(1)} = 1 - n^{-\Omega(1)}$. This finishes the proof. ■

Finally, we use Lemma 25 to prove Theorem 16.

Proof of Theorem 16. First, the error bound of Algorithm 3 directly follows by Lemma 25. For the time complexity of Algorithm 3, for each $u \in \mathcal{V}$, in Phase I we perform $n_r^{(1)}$ random walks, each length L, so the time complexity is $O(n_r^{(1)} L) = O(\frac{4L^3 \log n}{\epsilon}) = \tilde{O}(\frac{L^3}{\epsilon})$. In Phase II we perform CGD from the candidate set $S'_{\epsilon,u}$ with threshold $\frac{r_{max}}{2\mathbf{p}'_{L,u}}$, the time complexity is bounded by $O(\sum_{v \in S'_{\epsilon,u}} \frac{L^3 \mathbf{p}'_{L,u}(v)}{r_{max}}) = O(\sum_{v \in S'_{\epsilon,u}} \frac{L^3 \mathbf{p}_{L,u}(v)}{r_{max}})$. By the fact that $\|\mathbf{p}_{L,u}\|_1 \leq \frac{L}{2}$, this is further bounded by $O(\frac{L^4}{r_{max}})$. In phase III we perform $n_r^{(3)}$ random walks from u, each length L, so the time complexity is $O(n_r^{(3)} L) = O(\frac{L^2}{\epsilon^2} r_{max} d_u \log n) = \tilde{O}(\frac{L^2}{\epsilon^2} r_{max} d_u)$. We sum over all $u \in \mathcal{V}$, and setting $r_{max} = L\epsilon\sqrt{\frac{n}{m}}$, the total time complexity is:

$$\tilde{O}\left(n\frac{L^3}{\epsilon} + n\frac{L^4}{r_{max}} + m\frac{L^2}{\epsilon^2} r_{max}\right) = \tilde{O}\left(n\frac{L^3}{\epsilon} + \sqrt{nm}\frac{L^3}{\epsilon}\right) = \tilde{O}\left(\frac{\sqrt{nm}}{\epsilon}\right).$$

For the space complexity, by Fact 4 and Lemma 24, the size of the candidate set $S'_{\epsilon,u}$ can be bounded by $|S'_{\epsilon,u}| \leq \tilde{O}(\frac{1}{\epsilon})$, so the total space complexity is $\tilde{O}(\sum_{u \in \mathcal{V}} |S'_{\epsilon,u}|) = \tilde{O}(\frac{n}{\epsilon})$. This proves Theorem 16. ■

Arming with the query algorithm (Algorithm 4) and Theorem 16, we are ready to prove Theorem 5.

Proof of Theorem 5. Given any vertex pair (s,t), without loss of generality we assume $d_s \leq d_t$. By Equ. (1) and the fact $\frac{\mathbf{p}_{L,s}(t)}{d_t} = \frac{\mathbf{p}_{L,t}(s)}{d_s}$, we can express the ER value by

$$r_{\mathcal{G},L}(s,t) = \frac{\mathbf{p}_{L,s}(s)}{d_s} - 2\frac{\mathbf{p}_{L,t}(s)}{d_s} + \frac{\mathbf{p}_{L,t}(t)}{d_t}.$$

By Theorem 16, Algorithm 3 outputs the approximation such that $|\mathbf{p}_{L,s}(s) - \hat{\mathbf{p}}_{L,s}(s)| \leq \epsilon$, $|\mathbf{p}_{L,t}(s) - \hat{\mathbf{p}}_{L,t}(s)| \leq \epsilon$ and $|\mathbf{p}_{L,t}(t) - \hat{\mathbf{p}}_{L,t}(t)| \leq \epsilon$. Thus the approximation $\hat{r}_{\mathcal{G}}(s,t)$ output by Algorithm 4 satisfy: $|\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq \frac{4\epsilon}{d_s} \leq 8\epsilon r_{\mathcal{G}}(s,t)$. By Lemma 9, $|r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq \epsilon r_{\mathcal{G}}(s,t)$, thus $|\hat{r}_{\mathcal{G}}(s,t) - r_{\mathcal{G}}(s,t)| \leq 9\epsilon r_{\mathcal{G}}(s,t)$. In addition, since the query time of Algorithm 4 is $O(1)$ for a given vertex pair (s,t), so our algorithm is an $(\tilde{O}(\frac{\sqrt{nm}}{\epsilon}), O(1), \tilde{O}(\frac{n}{\epsilon}))$-ER sketch algorithm. Finally, we combine our results with a previous resistance sparsifier result from Chu et al. (2018) .

Theorem 26 *Chu et al. (2018) Given graph $\mathcal{G}$ with m edges and n vertices, there exist an algorithm that runs in $\tilde{O}(m)$ time and produce a graph $\mathcal{H}$ with $\tilde{O}(\frac{n}{\epsilon})$ edges, such that for $\forall s,t \in \mathcal{V}$, $r_{\mathcal{G}}(s,t) \approx_\epsilon r_{\mathcal{H}}(s,t)$.*

Therefore, our algorithm can be also processed in $\tilde{O}(m + n/\epsilon^{1.5})$ time. Putting it together, our algorithm is an $(\tilde{O}(\min\{m + n/\epsilon^{1.5}, \sqrt{nm}/\epsilon\}), O(1), \tilde{O}(n/\epsilon))$-ER sketch algorithm. ■

Remark We remark that as a corollary, our $\tilde{O}(\frac{\sqrt{nm}}{\epsilon})$ time algorithm provides the first sublinear algorithm for the multiple pair ER computation problem. Besides, we remark that if we simply use CGD to approximate $\mathbf{p}_{L,u}$ with threshold $\frac{\epsilon}{d_u}$ for each $u \in \mathcal{V}$, by Lemma 21, this also satisfies the error guarantee of Theorem 16. In addition, by Lemma 21, the total time complexity of this procedure is bounded by $O(\sum_{u\in\mathcal{V}} \frac{L^3 d_u}{\epsilon}) = \tilde{O}(\frac{m}{\epsilon})$, which is asymptotically the same processing time as the Algorithm in Dwaraknath et al. (2024) for expander graphs. This is an interesting corollary because CGD is a deterministic algorithm that achieves the same complexity as the previous state-of-the-art randomized algorithms.

Appendix E. Other Omitting Proofs

Proof of Lemma 7. We use the similar argument as Lemma 4.3 in Peng et al. (2021). First, we note that $\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{D}^{1/2}(\mathbf{I} - \mathcal{A})\mathbf{D}^{1/2}$, where $\mathcal{A}$ is the normalized adjacency matrix. Therefore, we have:

$$\begin{aligned}
\mathbf{L}^\dagger &= \mathbf{D}^{-1/2}(\mathbf{I} - \mathcal{A})^\dagger \mathbf{D}^{-1/2} = \mathbf{D}^{-1/2} \sum_{j=2}^{n} \frac{1}{1-\lambda_j(\mathcal{A})} \mathbf{u}_j \mathbf{u}_j^T \mathbf{D}^{-1/2} \\
&= \frac{1}{2}\mathbf{D}^{-1/2} \sum_{j=2}^{n} \sum_{l=0}^{\infty} \left(\frac{1}{2} + \frac{1}{2}\lambda_j(\mathcal{A})\right)^l \mathbf{u}_j \mathbf{u}_j^T \mathbf{D}^{-1/2} \\
&= \frac{1}{2}\mathbf{D}^{-1/2} \sum_{l=0}^{\infty} \sum_{j=2}^{n} \left(\frac{1}{2} + \frac{1}{2}\lambda_j(\mathcal{A})\right)^l \mathbf{u}_j \mathbf{u}_j^T \mathbf{D}^{-1/2} \\
&= \frac{1}{2}\mathbf{D}^{-1/2} \sum_{l=0}^{\infty} \left(\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathcal{A}\right)^l - \mathbf{u}_1 \mathbf{u}_1^T \right) \mathbf{D}^{-1/2}.
\end{aligned}$$

Where $\mathbf{u}_j$ denotes the eigenvector of $\mathcal{A}$ corresponding to the eigenvalue $\lambda_j(\mathcal{A})$. Next, we notice the fact that $\mathbf{u}_1 = \mathbf{D}^{1/2}\mathbf{1} \perp \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t)$ for any $s, t \in \mathcal{V}$, where $\mathbf{1}$ is the all-one vector. Therefore,

$$\begin{aligned} r_{\mathcal{G}}(s,t) &= (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^\dagger (\mathbf{e}_s - \mathbf{e}_t) \\ &= \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1/2} \sum_{l=0}^{\infty} \left(\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathcal{A} \right)^l - \mathbf{u}_1 \mathbf{u}_1^T \right) \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t) \\ &= \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1/2} \sum_{l=0}^{+\infty} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathcal{A} \right)^l \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t) \\ &= \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1} \sum_{l=0}^{+\infty} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l (\mathbf{e}_s - \mathbf{e}_t). \end{aligned}$$

This finishes the proof. ■

Proof of Lemma 9. By the definition of $r_{\mathcal{G},L}(s,t)$, we have that

$$\begin{aligned} &|r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \\ &= \left| \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1} \sum_{l=L+1}^{\infty} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l (\mathbf{e}_s - \mathbf{e}_t) \right| \\ &= \left| \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1/2} \sum_{l=L+1}^{\infty} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathcal{A} \right)^l \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t) \right| \\ &= \left| \frac{1}{2}(\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{D}^{-1/2} \sum_{l=L+1}^{\infty} \sum_{j=2}^{n} \left(\frac{1}{2} + \frac{1}{2}\lambda_j(\mathcal{A}) \right)^l \mathbf{u}_j \mathbf{u}_j^T \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t) \right|. \end{aligned}$$

Next, we denote $\alpha_j = \mathbf{u}_j^T \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t)$, and $\alpha = (\alpha_1, ..., \alpha_n)^T = (\mathbf{u}_1, ..., \mathbf{u}_n)^T \mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t)$. Therefore, we can bound

$$\|\alpha\|_2 = \|(\mathbf{u}_1, ..., \mathbf{u}_n)\mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t)\|_2 \leq \|\mathbf{D}^{-1/2}(\mathbf{e}_s - \mathbf{e}_t)\|_2 \leq \sqrt{2}.$$

And thus $\sum_{j=1}^{n} \alpha_j^2 \leq 2$. Therefore,

$$|r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| = \left| \frac{1}{2} \sum_{l=L+1}^{\infty} \sum_{j=2}^{n} \left(\frac{1}{2} + \frac{1}{2}\lambda_j(\mathcal{A}) \right)^l \alpha_j^2 \right| \leq \sum_{l=L+1}^{\infty} \left(\frac{1}{2} + \frac{1}{2}\lambda_2(\mathcal{A}) \right)^l,$$

where $\lambda_2(\mathcal{A})$ is the second largest eigenvalue of $\mathcal{A}$. By the property of the eigenvalues of normalized Laplacian matrix, $\lambda_2(\mathcal{L}) = 1 - \lambda_2(\mathcal{A})$. Therefore,

$$\begin{aligned} |r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| &\leq \sum_{l=L+1}^{\infty} \left(1 - \frac{1}{2}\lambda_2(\mathcal{L}) \right)^l \leq \left(1 - \frac{1}{2}\lambda_2(\mathcal{L}) \right)^L \frac{2}{\lambda_2(\mathcal{L})} \\ &\leq \left(1 - \frac{1}{2}\lambda_2(\mathcal{L}) \right)^L \kappa(\mathcal{L}) \leq \frac{\epsilon}{n}. \end{aligned}$$

The final inequality holds when setting $L = \log \frac{\epsilon}{n\kappa(\mathcal{L})} / \log\left(1 - \frac{1}{2}\lambda_2(\mathcal{L})\right)$. We notice the fact $\log\left(1 - \frac{1}{2}\lambda_2(\mathcal{L})\right) \leq -\frac{1}{2}\lambda_2(\mathcal{L})$ and $\kappa(\mathcal{L}) \leq \frac{2}{\lambda_2(\mathcal{L})}$, so we set $L \geq 2\kappa(\mathcal{L}) \log \frac{n}{\epsilon}$ is enough. Finally, by the fact $r_{\mathcal{G}}(s,t) \geq \frac{1}{d_s} + \frac{1}{d_t} \geq \frac{1}{n}$, we have $|r_{\mathcal{G}}(s,t) - r_{\mathcal{G},L}(s,t)| \leq \frac{\epsilon}{n} \leq \epsilon r_{\mathcal{G}}(s,t)$, thus $r_{\mathcal{G},L}(s,t)$ is the ϵ-approximation of $r_{\mathcal{G}}(s,t)$. ■

Proof of Fact 1. By Definition 10,

$$\|\mathbf{p}_{L,u}\|_1 = \left\| \frac{1}{2} \sum_{l=0}^{L} \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l \mathbf{e}_u \right\|_1 \leq \frac{1}{2} \sum_{l=0}^{L} \left\| \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l \mathbf{e}_u \right\|_1 \leq \frac{L}{2}.$$

The final inequality holds because $\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P}\right)^l$ is a column stochastic matrix. ■

Proof of Fact 2. By Definition 10,

$$\begin{aligned} \frac{\mathbf{p}_{L,u}(v)}{d_v} &= \frac{1}{2} \sum_{l=0}^{L} \frac{1}{d_v} \left(\left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l \mathbf{e}_u \right)(v) = \frac{1}{2} \sum_{l=0}^{L} \frac{1}{d_v} \mathbf{e}_v^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l \mathbf{e}_u \\ &= \frac{1}{2} \sum_{l=0}^{L} \frac{1}{d_u} \mathbf{e}_u^T \left(\frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{P} \right)^l \mathbf{e}_v = \frac{\mathbf{p}_{L,v}(u)}{d_u}. \end{aligned}$$

■

Proof of Fact 3. We notice the fact that $\mathbf{L} = \mathbf{D} - \mathbf{A} \preceq 2\mathbf{D}$. Thus, for any $\mathbf{x} \perp \mathbf{1}$, $\mathbf{x}^T\mathbf{L}^\dagger\mathbf{x} \geq \mathbf{x}^T \frac{1}{2}\mathbf{D}^{-1}\mathbf{x}$. Therefore,

$$\begin{aligned} r_{\mathcal{G}}(s,t) &= (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^\dagger (\mathbf{e}_s - \mathbf{e}_t) \\ &\geq (\mathbf{e}_s - \mathbf{e}_t)^T \frac{1}{2}\mathbf{D}^{-1} (\mathbf{e}_s - \mathbf{e}_t) = \frac{1}{2}\left(\frac{1}{d_s} + \frac{1}{d_t} \right). \end{aligned}$$

■

Proof of Lemma 13. To prove Lemma 13, we need the following two Propositions. The first Proposition is the submatrix representation of ER value, and the second Proposition is the classic block matrix inverse formula.

Proposition 27 *(Corollary 1 in Liao et al. (2023))* $r_{\mathcal{G}}(s,t) = (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^\dagger (\mathbf{e}_s - \mathbf{e}_t) = (\mathbf{L}_s^{-1})_{tt} = (\mathbf{L}_t^{-1})_{ss}$ *for any graph* $\mathcal{G}$*. Where* $\mathbf{L}$ *denotes the Laplacian matrix of* $\mathcal{G}$*, and* $\mathbf{L}_s$ *(resp.,* $\mathbf{L}_t$*) denotes the submatrix of* $\mathbf{L}$*, deleting the row and column indexed by node* s *(resp.,* t*).*

Proposition 28 *Let* $S = D - CA^{-1}B$*, we have:*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{bmatrix}$$

Now we use Proposition 27 and Proposition 28 to prove Lemma 13. We observe that the resistance $r_{S_2}(s,t)$ is simply the the parallel connection of $(d_s - x)$ resistors with resistance value 2. Thus, by the parallel resistance formula, $r_{S_2}(s,t) = \frac{2}{d_s - x}$. For $r_{S_1}(s,t)$, we define $\mathbf{L}_{S_1 \cup \{s,t\}} \in \mathbb{R}^{(n_1+2)\times(n_1+2)}$ as the Laplacian matrix constraint on the edges between $S_1 \cup \{s,t\}$, $\mathbf{L}_{S_1} \in \mathbb{R}^{n_1 \times n_1}$

as the Laplacian matrix constraint on the edges between S_1. Let $\mathbf{e}_{\mathcal{N}_1}$ denotes the indicating vector that only takes value 1 in $\mathcal{N}_1$ and 0 elsewhere, $\mathbf{I}_{\mathcal{N}_1}$ denotes the diagonal matrix that only takes value 1 at diagonal entries indexed by $\mathcal{N}_1$ and 0 elsewhere, where $\mathcal{N}_1 = \{v_1, ..., v_x\}$ is the set of the neighbors of s in S_1. By the definition of ER and the submatrix representation (Proposition 27), we have:

$$r_{S_1}(s,t) = (\mathbf{e}_s - \mathbf{e}_t)^T \mathbf{L}^{\dagger}_{S_1 \cup \{s,t\}} (\mathbf{e}_s - \mathbf{e}_t) = \begin{bmatrix} x & -\mathbf{e}_{\mathcal{N}_1}^T \\ -\mathbf{e}_{\mathcal{N}_1} & \mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} \end{bmatrix}^{-1}_{s,s}$$

Next, we use the block matrix inverse formula (Proposition 28), we have:

$$\begin{bmatrix} x & -\mathbf{e}_{\mathcal{N}_1}^T \\ -\mathbf{e}_{\mathcal{N}_1} & \mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} \end{bmatrix}^{-1}_{s,s} = \frac{1}{x} + \frac{1}{x^2} \mathbf{e}_{\mathcal{N}_1}^T \left[\mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} - \frac{1}{x} \mathbf{e}_{\mathcal{N}_1} \mathbf{e}_{\mathcal{N}_1}^T \right]^{-1} \mathbf{e}_{\mathcal{N}_1} \tag{10}$$

Now we prove the second term of the right hand side of Eq. (10) is at most $O(\frac{\log n_1}{dx})$. To this end, we notice that $|\mathcal{N}_1| = x$ and $\mathbf{I}_{\mathcal{N}_1} - \frac{1}{x} \mathbf{e}_{\mathcal{N}_1} \mathbf{e}_{\mathcal{N}_1}^T \succeq 0$, thus

$$\mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} - \frac{1}{x} \mathbf{e}_{\mathcal{N}_1} \mathbf{e}_{\mathcal{N}_1}^T \succeq \mathbf{L}_{S_1} + \mathbf{I}.$$

Next, we define $\mathbf{A}_{S_1}, \mathbf{P}_{S_1} \in \mathbb{R}^{n_1 \times n_1}$ to be the adjacency matrix and probability transition matrix constraint on the edges between S_1, respectively. By our construction, we have

$$\begin{aligned} \mathbf{L}_{S_1} + \mathbf{I} &= (d+1)\mathbf{I} - \mathbf{A}_{S_1} = (d+1)\mathbf{I} - d\mathbf{P}_{S_1} \\ &= (d+1) \left[\mathbf{I} - \left(1 - \frac{1}{d+1} \right) \mathbf{P}_{S_1} \right]. \end{aligned}$$

Therefore, we can bound:

$$\begin{aligned} 0 \leq \mathbf{e}_{\mathcal{N}_1}^T \left[\mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} - \frac{1}{x} \mathbf{e}_{\mathcal{N}_1} \mathbf{e}_{\mathcal{N}_1}^T \right]^{-1} \mathbf{e}_{\mathcal{N}_1} &\leq \frac{1}{d+1} \mathbf{e}_{\mathcal{N}_1}^T \left[\mathbf{I} - \left(1 - \frac{1}{d+1} \right) \mathbf{P}_{S_1} \right]^{-1} \mathbf{e}_{\mathcal{N}_1} \\ &= \frac{1}{d+1} \sum_{k=0}^{\infty} \left(1 - \frac{1}{d+1} \right)^k \mathbf{e}_{\mathcal{N}_1}^T \mathbf{P}_{S_1}^k \mathbf{e}_{\mathcal{N}_1}. \end{aligned} \tag{11}$$

Next we use the classical random walk mixing result. For expander graph S_1, if we start a random walk with initial probability $\pi_{\mathcal{N}_1} = \frac{1}{|\mathcal{N}_1|} \mathbf{e}_{\mathcal{N}_1}$, after $K = O(\log n_1)$ steps, $\left\| \mathbf{P}_{S_1}^K \pi_{\mathcal{N}_1} - \frac{\mathbf{1}}{n} \right\|_2 \leq \frac{1}{n^2}$. Combing this result with Eq. (11), we have:

$$\begin{aligned} &\mathbf{e}_{\mathcal{N}_1}^T \left[\mathbf{L}_{S_1} + \mathbf{I} + \mathbf{I}_{\mathcal{N}_1} - \frac{1}{x} \mathbf{e}_{\mathcal{N}_1} \mathbf{e}_{\mathcal{N}_1}^T \right]^{-1} \mathbf{e}_{\mathcal{N}_1} \\ &\leq \frac{1}{d+1} \left[\sum_{k=0}^{K} \left(1 - \frac{1}{d+1} \right)^k \mathbf{e}_{\mathcal{N}_1}^T \mathbf{P}_{S_1}^k \mathbf{e}_{\mathcal{N}_1} + \sum_{k=K}^{\infty} \left(1 - \frac{1}{d+1} \right)^k \mathbf{e}_{\mathcal{N}_1}^T \mathbf{P}_{S_1}^k \mathbf{e}_{\mathcal{N}_1} \right] \\ &\leq \frac{1}{d+1} \left[K|\mathcal{N}_1| + \frac{\left(1 - \frac{1}{d+1} \right)^{K+1}}{1 - \left(1 - \frac{1}{d+1} \right)} |\mathcal{N}_1|^2 \frac{2}{n_1} \right] \\ &\leq \frac{x}{d+1} O(\log n_1) + \frac{2x^2}{n_1}. \end{aligned}$$

Finally, we set $n_1 \geq 2d^3 \geq 2dx^2$, the above inequality is bounded by $O(\frac{x \log n_1}{d})$. Combining this result with Eq. (10), we obtain $r_{S_1}(s,t) = \frac{1}{x} + O(\frac{1}{dx}\log n_1)$. This finishes the proof. ■

Proof of Lemma 14. Recall that by our construction, for $\mathcal{G}_1$ all neighbors of s is in S_1. Thus, $r_{\mathcal{G}_1}(s,t) = r_{S_1}(s,t) = \frac{1}{d_s} + O(\frac{\log n_1}{dd_s}) = \frac{1}{d_s} + O(\frac{1}{d_s^2})$. This is equivalently saying $d_s r_{\mathcal{G}_1}(s,t) = 1 + O(\frac{1}{d_s})$. On the other hand, for $\mathcal{G}_2$, the neighbors of s are $x = (1-\epsilon)d_s$ in S_1 and ϵd_s in S_2. By the parallel resistance formula and Lemma 13,

$$\begin{aligned} r_{\mathcal{G}_2}(s,t) &= \frac{r_{S_1}(s,t) r_{S_2}(s,t)}{r_{S_1}(s,t) + r_{S_2}(s,t)} = \frac{\frac{2}{d_s - x}\left(\frac{1}{x} + O(\frac{1}{d_s x})\right)}{\frac{2}{d_s - x} + \frac{1}{x} + O(\frac{1}{d_s x})} \\ &= \frac{2 + O(\frac{1}{d_s})}{d + x + O(\frac{d_s - x}{d_s})} = \frac{2 + O(\frac{1}{d_s})}{(2-\epsilon)d_s + O(\epsilon)}. \end{aligned}$$

Thus, we have

$$\begin{aligned} d_s r_{\mathcal{G}_2}(s,t) &= \frac{2d_s\left(1 + O(\frac{1}{d_s})\right)}{(2-\epsilon)d_s + O(\epsilon)} = \frac{2}{2-\epsilon}\frac{1 + O(\frac{1}{d_s})}{1 + O(\frac{\epsilon}{d_s})} \\ &= \frac{2}{2-\epsilon}\left(1 + O(\frac{1}{d_s})\right). \end{aligned}$$

Finally, we set $d_s \gg \frac{1}{\epsilon}$, thus $d_s r_{\mathcal{G}_2}(s,t) = \frac{2}{2-\epsilon}(1 + o(\epsilon))$. Therefore,

$$\begin{aligned} |r_{\mathcal{G}_1}(s,t) - r_{\mathcal{G}_2}(s,t)| &= \frac{1}{d_s}\left[\frac{2}{2-\epsilon} - 1 + o(\epsilon)\right] = \frac{1}{d_s}\left[\frac{\epsilon}{2-\epsilon} + o(\epsilon)\right] \\ &\geq \frac{\epsilon}{2}\frac{1}{d_s} \geq \frac{\epsilon}{4} r_{\mathcal{G}_1}(s,t). \end{aligned}$$

The final inequality holds because $r_{\mathcal{G}_1}(s,t) = \frac{1}{d_s} + O(\frac{1}{d_s^2}) \leq \frac{2}{d_s}$. This finishes the proof. ■

Proof of Lemma 15. Through our construction, we observe that the only way to distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$ is to perform a neighbor query on node s. For $\mathcal{G}_1$, we have all the neighbors of s, $\mathcal{N}_{\mathcal{G}_1}(s) \subset S_1$. For $\mathcal{G}_2$, we have $\mathcal{N}_1 \subset S_1$ with $|\mathcal{N}_1| = (1-\epsilon)d_s$, $\mathcal{N}_2 \subset S_2$ with $|\mathcal{N}_2| = \epsilon d_s$ and $\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}_{\mathcal{G}_2}(s)$ is the neighbor set of s. Thus, distinguishing $\mathcal{G}_1$ and $\mathcal{G}_2$ via a neighbor query on s reduces to determining whether the elements of $\mathcal{N}_{\mathcal{G}_1}(s), \mathcal{N}_{\mathcal{G}_2}(s)$ follow the same distribution. We prove this result using the classical Yao's Minimax Principle.

Theorem 29 *(Yao's Minimax Principle, see e.g. Lemma D.1 from Cai et al. (2023)) Let $\mathcal{X}$ denotes the set of inputs of a problem, $\Delta(\mathcal{X})$ denotes the set of all distributions over $\mathcal{X}$. Let $\mathcal{A}_\mu$ denotes minimum cost among all deterministic algorithms that solves the problem with probability $\geq 2/3$, with respect to the distribution $\mu \in \Delta(\mathcal{X})$. Let $\mathcal{R}$ denotes the minimum cost among all randomized algorithms that solves the problem for all $x \in \mathcal{X}$. Then, $\mathcal{R} \geq \max_{\mu \in \mathcal{X}} \mathcal{A}_\mu$.*

In our case, we choose the problem inputs $\mathcal{X} = \{\mathcal{N}(s)\}$, and we consider the following two events $C_1, C_2 \subset \mathcal{X}$:

$$\begin{aligned} C_1 &= \{\mathcal{N}(s) : \forall v \in \mathcal{N}(s), v \in S_1\} \\ C_2 &= \{\mathcal{N}(s) : \mathcal{N}(s) = \mathcal{N}_1 \cup \mathcal{N}_2, \\ &\qquad \mathcal{N}_1 \subset S_1, \mathcal{N}_2 \subset S_2, \text{ and } |\mathcal{N}_1| = (1-\epsilon)|\mathcal{N}(s)|\} \end{aligned}$$

Next, we consider the distribution μ over $\mathcal{X}$, such that: (i) C_1 happens with probability $\frac{1}{2}$ and $\forall x \in C_1 \subset \mathcal{X}$ happens with equal probability; (ii) C_2 happens with probability $\frac{1}{2}$ and $\forall x \in C_2 \subset \mathcal{X}$ happens with equal probability. We consider any property testing deterministic algorithm A, such that A should answer YES if there are $\geq \epsilon|\mathcal{N}(s)|$ number of nodes $v \in \mathcal{N}(s)$ in S_2; A should answer NO if there are $< \epsilon|\mathcal{N}(s)|$ number of nodes $v \in \mathcal{N}(s)$ in S_2. We denote $A(C_1)$, $A(C_2)$ as the random variable, which represents the answer of A on some $x \in C_1$ uniformly randomly (resp., $x \in C_2$ uniformly randomly). We consider when A succeeds with probability $\geq 2/3$:

$$\frac{2}{3} \leq \mathbb{P}_\mu[A \ \ succeeds] = \frac{1}{2}\mathbb{P}[A(C_1) = NO] + \frac{1}{2}\mathbb{P}[A(C_2) = YES].$$

Since C_1 is fixed, we only need to consider the second probability $\mathbb{P}[A(C_2) = YES]$. We assume A makes q queries on C_2. Then,

$$\begin{aligned}
\mathbb{P}[A(C_2) = YES] &= \mathbb{P}[A(C_2) = YES | A(C_1) = A(C_2)]\mathbb{P}[A(C_1) = A(C_2)] \\
&\quad + \mathbb{P}[A(C_2) = YES | A(C_1) \neq A(C_2)]\mathbb{P}[A(C_1) \neq A(C_2)] \\
&\leq \mathbb{P}[A(C_1) = YES] + \mathbb{P}[A(C_1) \neq A(C_2)] \qquad (12) \\
&= \mathbb{P}[A(C_1) = YES] + \mathbb{P}[one\ query\ returns\ v \in S_2] \\
&\leq 1 - \mathbb{P}[A(C_1) = NO] + \epsilon q.
\end{aligned}$$

The final inequality holds because there are only $\epsilon|\mathcal{N}(s)|$ randomly chosen elements in S_2. For each fixed query, the probability that we find some $v \in S_2$ is at most ϵ. Arming with Eq. (12), we have:

$$\frac{2}{3} \leq \frac{1}{2}\mathbb{P}[A(C_1) = NO] + \frac{1}{2}\mathbb{P}[A(C_2) = YES] \leq \frac{1}{2} + \frac{1}{2}\epsilon q.$$

As a result, $q \geq \frac{1}{3\epsilon}$. This implies A makes at least $\Omega(\frac{1}{\epsilon})$ queries on C_2. By Yao's Minimax Principle, any randomized algorithm with success probability $\geq 2/3$ that distinguish $\mathcal{N}_{\mathcal{G}_1}(s)$ and $\mathcal{N}_{\mathcal{G}_2}(s)$ requires $\Omega(\frac{1}{\epsilon})$ queries. This finishes the proof of Theorem 12. ■