Local Algorithms for Estimating Effective Resistance

Pan Peng
Department of Computer Science
University of Sheffield
p.peng@sheffield.ac.uk

Daniel Lopatta
Department of Computer Science
University of Sheffield
dlopatta1@sheffield.ac.uk

Yuichi Yoshida
Principles of Informatics Research Division
National Institute of Informatics
yyoshida@nii.ac.jp

Gramoz Goranci
School of Computing Science
University of Glasgow
gramoz.goranci@glasgow.ac.uk

ABSTRACT

Effective resistance is an important metric that measures the similarity of two vertices in a graph. It has found applications in graph clustering, recommendation systems and network reliability, among others. In spite of the importance of the effective resistances, we still lack efficient algorithms to exactly compute or approximate them on massive graphs.

In this work, we design several *local algorithms* for estimating effective resistances, which are algorithms that only read a small portion of the input while still having provable performance guarantees. To illustrate, our main algorithm approximates the effective resistance between any vertex pair s, t with an arbitrarily small additive error ε in time $O(\text{poly}(\log n/\varepsilon))$, whenever the underlying graph has bounded mixing time. We perform an extensive empirical study on several benchmark datasets, validating the performance of our algorithms.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Theory of computation → Sketching and sampling**.

KEYWORDS

Graph algorithms, Random walks, Effective resistances

1 INTRODUCTION

Metrics that capture the similarity between vertices in a graph have played a pivotal role in the quest for understanding the structure of large-scale networks. Typical examples include personalized PageRank (PPR) [32], Katz similarity [20] and SimRank [19], each of which can be thought of as a random walk-based measure on graphs. These metrics have found applications in recommender systems [21], link prediction [27, 35], etc.

A remarkably important random walk-based metric for measuring vertex similarity is the *effective resistance*. Given a graph G treated as a resistor network, the effective resistance $R_G(s, t)$ between two vertices s, t in G is the energy dissipation in the network when routing one unit of current from s to t. It is well known that the effective resistance is inherently related to the behaviour of random walks on graphs[1]. Concretely, the effective resistance between s and t is proportional to the *commute time* $\kappa(s, t)$, defined as the expected number of steps a random walk starting at s visits vertex t and then goes back to s [9]. Using this interpretation, we can deduce that the smaller $R_G(s, t)$ is, the more similar two vertices s, t are.

Indeed, effective resistance has proven ubiquitous in numerous applications including graph clustering [2, 16], recommender systems [22], measuring robustness of networks [15], spectral sparsification [36], graph convolutional networks [1], location-based advertising [37], among others. Moreover, in the theoretical computer science community, the use of effective resistance has led to a breakthrough line of work for provably speeding up the running time of many flow-based problems in combinatorial optimization [3, 11, 29].

Despite of the importance of effective resistance, we still lack efficient methods to compute or approximate them on massive graphs. For any two vertices s, t and approximation parameter $\varepsilon > 0$, one can $(1 + \varepsilon)$-approximate $R_G(s, t)$ in[2] $\tilde{O}(m \log(1/\varepsilon))$ time [13], where m denotes the number of edges in a graph. There exists an algorithm that $(1+\varepsilon)$-approximates *all-pairs* effective resistances in $\tilde{O}(n^2/\varepsilon)$ time [18]. These results, though theoretically competitive, require access to the entire input graph. Given the rapid growth of modern networks, such polynomial time algorithms (even those running in near linear time in the number of vertices and edges) are prohibitively costly. This motivates the following question:

Can we obtain a competitive estimation to $R_G(s, t)$ while exploring only a small portion of the graph?

We address this question by exploiting the paradigm of *local* or *sub-linear* algorithms. This computational model is particularly desirable in applications where one requires the effective resistances amongst only a few number of vertex pairs. Despite that the effective resistance is a key tool in large-scale graph analytics, designing local algorithms for estimating it is a largely unexplored topic.

In this paper, we provide several local algorithms for estimating pairwise effective resistances with provable performance guarantees. For any specified vertex pair s, t, our algorithms output an estimate of $R_G(s, t)$ with an arbitrarily small constant additive error, while exploring a small portion of the graph. To formally state our results, we utilize the well-known *adjacency list* model [33], which assumes query access to the input graph G and supports the following types of queries in constant time:

[1] We only consider simple random walks in the paper: suppose we are at vertex v, we jump to a neighbor of v with probability $1/\deg(v)$, where $\deg(v)$ is the degree of vertex v.

[2] Throughout the paper, we use $\tilde{O}$ to hide polylogarithmic factors, i.e., $\tilde{O}(f(n)) = O(f(n) \cdot \text{poly}\log f(n))$.

- *degree query*: for any specified vertex v, the algorithm can get the degree $\deg(v)$ of v;
- *neighbor query*: for any specified vertex v and index $i \leq \deg(v)$, the algorithm can get the i-th neighbor of v;
- *uniform sampling*: the algorithm can sample a vertex v of G uniformly at random.

Our main objective is to find a good estimate of the pairwise effective resistance $R_G(s,t)$ for any specified vertex pair s,t by making as few queries as possible to the graph while achieving fast running time.

Our contributions. We give a systemic study of local algorithms for estimating s-t effective resistances for general graphs.

- Theoretically, we provide three types of local algorithms for estimating effective resistances. All of them are based on random walks, but vary from their connections to effective resistances: (i) the first type is based on approximating the pseudo inverse of the Laplacian matrix, (ii) the second type is based on commute times, (iii) the third type is based on the number of spanning trees.
- We empirically demonstrate the competitiveness of our algorithms on popular benchmarks for graph data. In particular, for certain real-world networks, we will see that our algorithms run 10^5 to 10^6 faster than existing polynomial-time methods and estimate effective resistance to within a multiplicative error of 0.1.

To illustrate, our main local algorithm approximates $R_G(s,t)$ with an arbitrarily small additive error ε in time $O(\text{poly}(\log n/\varepsilon))$, whenever the underlying graph has bounded mixing time, which is justified in real-world networks. Previously, the only work on this problem was by Andoni et al. [4], and it achieves $(1+\varepsilon)$-approximation to $R_G(s,t)$ in $O(\frac{1}{\varepsilon^2}\text{poly}\log\frac{1}{\varepsilon})$ time for d-regular expander graphs. Indeed, one of our algorithms for general graphs is based on [4].

Using the fact that the length of shortest paths and effective resistances are exactly the same on tree graphs, we can observe that graphs with large mixing time do not admit efficient local algorithms. Concretely, let us consider a path graph on n vertices. It is known that the path graph has large mixing time, and there is no local algorithm that makes a sub-linear number of queries and approximates the length of shortest paths within a constant multiplicative factor or additive error, thus giving us the same impossibility result for effective resistances. This suggests that our bounded mixing time assumption is necessary to design local algorithms with sublinear number of queries and running time.

2 RELATED WORK

In this section, we discuss some related work.

Hayashi et al. [17] gave an algorithm for approximating the effective resistances of vertex pairs that are endpoints of edges. Their algorithm is based on sampling spanning trees uniformly at random, and it $(1+\varepsilon)$-approximates $R_G(s,t)$ for *every* $(s,t) \in E$ in expected running time $\lceil \log(2m/\delta)/2\varepsilon^2 \rceil \cdot \sum_{u\in V} \pi_G(u)\kappa_G(u,r)$, where $\pi_G(u)$ denotes the stationary probability at a vertex $u \in V$ of a random walk on G, $\kappa_G(u,v)$ denotes the commute time between two vertices $u,v \in V$ and $r \in V$ is some vertex.

There also exist several local algorithms for other random walk based quantities, such as the stationary distribution, PageRank, Personalized PageRank and transition probabilities.

The stationary distribution. Lee et al. [23] and Bressan et al. [8] studied the question of computing the stationary distribution π of a Markov Chain locally. These algorithms take as input any state v, and answer if the stationary probability of v exceeds some $\Delta \in (0,1)$ and/or output an estimate of $\pi(v)$. They only make use of a local neighborhood of v on the graph induced by the Markov chain and run in sublinear time for some families of Markov Chains.

PageRank. Borgs et al. presented a method for identifying all vertices whose PageRank is larger than some threshold [6]. Specifically, for a threshold value $\Delta \geq 1$ and a constant $c > 3$, with high probability, their algorithm returns a set $S \subseteq V$ such that S contains all vertices with PageRank at least Δ and no vertex with PageRank at least Δ/c. The algorithm runs in $\tilde{O}(\frac{n}{\Delta})$ time.

Bressan et al. developed a sub-linear time algorithm that employs local graph exploration [7]. Their algorithm $(1+\varepsilon)$-approximates the PageRank of a vertex on a directed graph. For constant $\varepsilon > 0$, the algorithm runs in $\tilde{O}(\min(m^{3/4}\Delta^{1/4}d^{-3/4}, m^{6/7}d^{-5/7}))$, where Δ and d are respectively the maximum and average outdegree.

Personalized PageRank (PPR). The PPR $\pi_s(t)$ of a start vertex s and target vertex t measures the frequency of visiting t via short random-walks from s. For a given threshold δ such that $\pi_s(t) > \delta$, Lofgren et al. solved this with small relative error and an expected running time of $O(\sqrt{d/\delta})$ [25], where d is the average in-degree of the graph. Their algorithm is based on a bi-directional search technique and an improved implementation was presented in [24].

Transition probabilities. Another problem related to effective resistance is estimating transition probabilities in a Markov chain. Specifically, given transition matrix P, initial source distribution σ, target state t, and a fixed length ℓ, the goal is to estimate the probability p that an ℓ-step random walk starting from distribution σ ends at t. Banerjee and Lofgren developed an algorithm that can estimate such a probability with respect to a minimum threshold δ such that $p > \delta$ by employing a bi-directional approach [5]. Specifically, their algorithms returns an estimator $\hat{p}$ of p such that with high probability $|\hat{p} - p| < \max\{\varepsilon p, \delta\}$ for any $\varepsilon > 0$.

3 PRELIMINARIES

Let $G = (V,E)$ be an undirected graph. For any $v \in V$, we let $\deg(v)$ denote the degree of v. The *volume* of a set S of vertices, denoted $\text{vol}(S)$, is the sum of their degrees. Furthermore, for a set $S \subseteq V$, the *conductance* of S, denoted $\phi_G(S)$, is the number of edges with one endpoint in S and the other in $V \setminus S$ divided by $\text{vol}(S)$. The *conductance* of G, denoted $\phi(G)$, is defined to be $\min_{S\subseteq V, 0<\text{vol}(S)\leq \frac{\text{vol}(V)}{2}} \phi_G(S)$. A graph G is called an *expander* if $\phi(G) \geq \phi$ for some universal constant $\phi \in (0,1)$.

Let $\mathbf{A}$ denote its adjacency matrix and let $\mathbf{D}$ denote the degree diagonal matrix. Let $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denote the *Laplacian* matrix of G. Let $\mathbf{L}^\dagger$ denote the Moore-Penrose pseudo-inverse of the Laplacian of G. Let $\mathbf{1}_u \in \mathbb{R}^V$ denote the (row) indicator vector of vertex u such that $\mathbf{1}_u(v) = 1$ if $v = u$ and 0 otherwise. Let $\chi_{s,t} = \mathbf{1}_s - \mathbf{1}_t$.

Definition 3.1. *Given any two vertices $u, v \in V$, the s-t* effective resistance *is defined as*

$$R_G(s,t) := \chi_{s,t}\mathbf{L}^{\dagger}\chi_{s,t}^{\top} = \mathbf{L}^{\dagger}_{s,s} - 2\mathbf{L}^{\dagger}_{s,t} + \mathbf{L}^{\dagger}_{t,t}.$$

Random walks. Given a graph G, we consider the simple random walk on G: suppose we are currently at v, then we jump to a neighbor u with probability $\frac{1}{\deg(v)}$. We use $\mathbf{P} := \mathbf{D}^{-1}\mathbf{A}$ to denote the random walk transition matrix. Let $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$, where λ_i is the i-th largest eigenvalue of the matrix $\mathbf{P}$.

Definition 3.2. *The commute time $\kappa(s,t)$ between vertices s, t is the expected number of steps in a random walk that starts at vertex s visits vertex t and then comes back to s.*

Random walks on graphs are a type of Markov Chain. A Markov chain is said to be *positive recurrent* if, starting in any state i, the expected time until the process returns to state i is finite. A Markov chain is said to be *aperiodic* if for any state i there are no restrictions on when it is possible for the process to enter state i.

Definition 3.3. *A Markov chain is said to be ergodic if it is aperiodic and positive recurrent.*

Informally, the *mixing time* of the graph G refers to the number of steps needed before a random walk on G converges to its stationary distribution. We refer to [34] for a formal definition. It is known that the spectral gap $1-\lambda$ is intimately related to the mixing time of G. That is, the larger $1-\lambda$ is, the smaller mixing time is, and vice versa.

4 THE LOCAL ALGORITHMS

4.1 Algorithms based on approximating Laplacian inverse

In this section, we provide local algorithms for effective resistances by approximating the Laplacian pseudo-inverse $\mathbf{L}^{\dagger}$ of the graph.

High-level idea. Our algorithm works for general graphs and is based on the aforementioned sublinear-time algorithm for d-regular graphs [4]. The basic idea is as follows. Recall that by definition of effective resistance, $R_G(s,t) = \chi_{s,t}\mathbf{L}^{\dagger}\chi_{s,t}^{\top}$ and $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ is the random walk transition matrix. Using the Neumann series of the matrix $\mathbf{L}^{\dagger}$ (see Lemma 4.3), we can write

$$\begin{aligned} R_G(s,t) &= \chi_{s,t}\sum_{i=0}^{\infty}\mathbf{P}^i\mathbf{D}^{-1}\chi_{s,t}^{\top} \\ &= \chi_{s,t}\sum_{i=0}^{\ell-1}\mathbf{P}^i\mathbf{D}^{-1}\chi_{s,t}^{\top} + \chi_{s,t}\sum_{i\geq\ell}^{\infty}\mathbf{P}^i\mathbf{D}^{-1}\chi_{s,t}^{\top}. \end{aligned}$$

for any $\ell > 0$. For graphs with large spectral gap (i.e., expander graphs or graphs with low random walk mixing time), we can show that for any additive error ε, we can choose ℓ appropriately such that the second term is at most $\varepsilon/2$, and the first term can be approximated within additive error $\varepsilon/2$. For the latter, we use a simple Monte Carlo approach (i.e., to use the empirical distribution of the endpoints of a small number of random walks) to approximate the quantity $\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^{\top}$, the (transition) probability that a length-i random walk starting from s ends at t, for any $i \geq 1$.

Now we introduce one assumption, building upon which we present and analyze two local algorithms.

Assumption 4.1. *Let G be a connected graph with minimum vertex degree at least* 1*. Further, assume that the Markov chain corresponding to the random walk on G is ergodic.*

The first algorithm: EstEff-TranProb. We first present an algorithm that uses the above idea. Recall that $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$, where λ_i is the i-th largest eigenvalue of the matrix $\mathbf{P}$.

Theorem 4.2. *Under Assumption 4.1, there is an algorithm EstEff-TranProb(G, ε, s, t) (see Algorithm 1) that outputs an estimate $\hat{\delta}_{s,t}$ such that with probability at least* 9/10*, it holds that*

$$|R_G(s,t) - \hat{\delta}_{s,t}| \leq \varepsilon.$$

The running time and query complexity of the algorithm are $O(\ell^4(\log\ell)/\varepsilon^2)$ for $\ell = \frac{\log(4/(\varepsilon-\varepsilon\lambda))}{\log(1/\lambda)}$.

The above algorithm is very efficient, if the graph has small λ, or has low mixing time, a property that is satisfied by many real networks. Now we present the algorithm EstEff-TranProb.

Algorithm 1: EstEff-TranProb(G, ε, s, t)

$\ell = \frac{\log(4/(\varepsilon-\varepsilon\lambda))}{\log(1/\lambda)}$
$r \leftarrow 40\ell^2(\log(80\ell))/\varepsilon^2$
for $i := 0, 1, \ldots, \ell-1$ **do**
  Perform r independent random walks of length i starting at s, and let $X_{i,s}$ (resp., $X_{i,t}$) be the number of walks that end at s (resp., t).
  Perform r independent random walks of length i starting at t, and let $Y_{i,s}$ (resp., $Y_{i,t}$) be the number of walks that end at s (resp., t)
  Set $\hat{\delta}^{(i)}_{s,t} = \frac{X_{i,s}}{r\deg(s)} - \frac{X_{i,t}}{r\deg(t)} - \frac{Y_{i,s}}{r\deg(s)} + \frac{Y_{i,t}}{r\deg(t)}$
return $\hat{\delta}_{s,t} = \sum_{i=0}^{\ell-1}\hat{\delta}^{(i)}_{s,t}$

Proof of Theorem 4.2 We first note that the running time and query complexity of the algorithm are $O(r\ell^2) = O(\ell^4(\log\ell)/\varepsilon^2)$.

In the following, we prove the correctness of the algorithm. We first present a basic property of effective resistance. Let $\mathbf{Q} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Recall that $\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{D}^{1/2}(I - \mathbf{Q})\mathbf{D}^{1/2}$ and that $R_G(s,t) = \chi_{s,t}\mathbf{L}^{\dagger}\chi_{s,t}^{\top}$. Note that $\mathbf{Q} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2}$ is symmetric and is similar to $\mathbf{P}$ (as the diagonal matrix $\mathbf{D}$ is invertible, which in turn follows from Assumption 4.1). We let $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathbf{Q}$ (and also $\mathbf{P}$ by the similarity of $\mathbf{P}$ and $\mathbf{Q}$), with corresponding (row) orthonormal eigenvectors $w_1, w_2, w_3, \ldots, w_n$, i.e,. $w_j\mathbf{Q} = \lambda_j w_j$. It is known that $\lambda_1 = 1$ and $w_1 = \frac{\mathbf{1}_V D^{1/2}}{\sqrt{2m}}$.

Lemma 4.3. *It holds that*

$$R_G(s,t) = \chi_{s,t}\sum_{i=0}^{\infty}\mathbf{P}^i\mathbf{D}^{-1}\chi_{s,t}^{\top}.$$

Proof. By the spectral decomposition of $\mathbf{Q}$, we have that for any integer $i \geq 0$, $\mathbf{Q}^i = \sum_{j=1}^n \lambda_j^i w_j^{\top} w_j = w_1^{\top}w_1 + \sum_{j=2}^n \lambda_j^i w_j^{\top} w_j$.

Since $\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{D}^{1/2}(I - \mathbf{Q})\mathbf{D}^{1/2}$, we have that

$$\mathbf{L}^{\dagger} = \mathbf{D}^{-1/2}(I - \mathbf{Q})^{\dagger}\mathbf{D}^{-1/2}$$

$$= \mathbf{D}^{-1/2} \sum_{j=2}^{n} \frac{1}{1-\lambda_j} w_j^\top w_j \mathbf{D}^{-1/2} = \mathbf{D}^{-1/2} \sum_{j=2}^{n} \sum_{i=0}^{\infty} \lambda_j^i w_j^\top w_j \mathbf{D}^{-1/2}$$
$$= \mathbf{D}^{-1/2} \sum_{i=0}^{\infty} \sum_{j=2}^{n} \lambda_j^i w_j^\top w_j \mathbf{D}^{-1/2} = \mathbf{D}^{-1/2} \sum_{i=0}^{\infty} \left(\mathbf{Q}^i - w_1^\top w_1\right) \mathbf{D}^{-1/2}.$$

Now we write $\chi_{s,t}\mathbf{D}^{-1/2} = \sum_{j=1}^{n} \alpha_j w_j$. We note that $\alpha_1 = \chi_{s,t}\mathbf{D}^{-1/2} w_1^\top = \chi_{s,t}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{1}_V^\top/\sqrt{2m} = 0$. Thus $\chi_{s,t}\mathbf{D}^{-1/2} = \sum_{j=2}^{n} \alpha_j w_j$. Then $\chi_{s,t}\mathbf{D}^{-1/2}\mathbf{Q}^i\mathbf{D}^{-1/2}\chi_{s,t}^\top = \sum_{j=1}^{n} \alpha_j^2 \lambda_j^i$.

Note that $\mathbf{P}^i\mathbf{D}^{-1} = (\mathbf{D}^{-1}\mathbf{A})^i\mathbf{D}^{-1} = \mathbf{D}^{-1/2}(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^i\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{Q}^i\mathbf{D}^{-1/2}$. Thus

$$R_G(s,t) = \chi_{s,t}\mathbf{L}^\dagger\chi_{s,t}^\top = \chi_{s,t}\mathbf{D}^{-1/2} \sum_{i=0}^{\infty} (\mathbf{Q}^i - w_1^\top w_1)\mathbf{D}^{-1/2}\chi_{s,t}^\top$$
$$= \sum_{i=0}^{\infty} \chi_{s,t}\mathbf{D}^{-1/2}\mathbf{Q}^i\mathbf{D}^{-1/2}\chi_{s,t}^\top = \sum_{i=0}^{\infty} \chi_{s,t}\mathbf{P}^i\mathbf{D}^{-1}\chi_{s,t}^\top. \quad \blacksquare$$

By Assumption 4.1, the Markov chain corresponding to the random walk on G is ergodic. Then $\lambda_2 < 1$. Recall that $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$ and that $\chi_{s,t}\mathbf{D}^{-1/2} = \sum_{j=1}^{n} \alpha_j w_j$, where $\alpha_1 = 0$. Furthermore, by the assumption that each vertex has degree at least 1, we have $\sum_{j=2}^{n} \alpha_j^2 = \|\chi_{s,t}\mathbf{D}^{-1/2}\|_2^2 \le \|\chi_{s,t}\|_2^2 = 2$. Now we prove the following two claims.

Claim 4.4. *It holds that* $|R_G(s,t) - \sum_{i=0}^{\ell-1} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top| \le \frac{\varepsilon}{2}$.

Proof. It holds that

$$\left|R_G(s,t) - \sum_{i=0}^{\ell-1} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top\right|$$
$$= \left|\sum_{i=0}^{\infty} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top - \sum_{i=0}^{\ell-1} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top\right|$$
$$= \left|\sum_{i=\ell}^{\infty} \chi_{s,t}\mathbf{D}^{-1/2}\mathbf{Q}^i\mathbf{D}^{-1/2}\chi_{s,t}^\top\right| = \left|\sum_{i=\ell}^{\infty} \sum_{j=2}^{n} \alpha_j^2 \lambda_j^i\right|$$
$$\le \sum_{i=\ell}^{\infty} \lambda^i \sum_{j=2}^{n} \alpha_j^2 \le \frac{2\lambda^\ell}{1-\lambda} \le \frac{\varepsilon}{2},$$

where the last inequality follows from $\ell = \frac{\log(4/(\varepsilon-\varepsilon\lambda))}{\log(1/\lambda)}$. $\blacksquare$

Claim 4.5. *With probability at least* $9/10$,

$$\left|\hat{\delta}_{s,t} - \sum_{i=0}^{\ell-1} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top\right| \le \frac{\varepsilon}{2}.$$

Proof. We observe that for any $i \ge 0$,

$$\chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top = (\mathbf{1}_s - \mathbf{1}_t)P^i \cdot \mathbf{D}^{-1}(\mathbf{1}_s - \mathbf{1}_t)^\top$$
$$= \frac{\mathbf{1}_s P^i \mathbf{1}_s^\top}{\deg(s)} - \frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)} - \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_s^\top}{\deg(s)} + \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)}.$$

Note that for any $0 \le i \le \ell - 1$, in the algorithm, we perform r random walks of length i from s. Since $X_{i,s}$ is the number of walks that end at s and $\mathbf{1}_s\mathbf{P}^i\mathbf{1}_s^\top$ is exactly the probability of a random walk of length i from s ends at s, we have that

$$\mathbb{E}X_{i,s} = r \cdot \mathbf{1}_s\mathbf{P}^i\mathbf{1}_s^\top.$$

Furthermore, by the Chernoff-Hoeffding bound,

$$\mathbb{P}\left[\left|\frac{X_{i,s}}{r\deg(s)} - \frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_s^\top}{\deg(s)}\right| \ge \frac{\varepsilon}{8\ell}\right] = \mathbb{P}\left[\left|\frac{X_{i,s}}{r\deg(s)} - \frac{\mathbb{E}[X_{i,s}]}{r\deg(s)}\right| \ge \frac{\varepsilon}{8\ell}\right]$$
$$= \mathbb{P}\left[\left|X_{i,s} - \mathbb{E}[X_{i,s}]\right| \ge \frac{r\deg(s)\varepsilon}{8\ell}\right]$$
$$\le 2\exp(-2\deg(s)^2\varepsilon^2 r^2/(64\ell^2 r)) \le 2\exp(-\varepsilon^2 r/(32\ell^2)) \le \frac{1}{40\ell},$$

where the last inequality follows from $r = 40\ell^2(\log(80\ell))/\varepsilon^2$. Similarly,

$$\mathbb{P}\left[\left|\frac{X_{i,t}}{r\deg(t)} - \frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)}\right| \ge \frac{\varepsilon}{8\ell}\right] \le \frac{1}{40\ell},$$
$$\mathbb{P}\left[\left|\frac{Y_{i,s}}{r\deg(s)} - \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_s^\top}{\deg(s)}\right| \ge \frac{\varepsilon}{8\ell}\right] \le \frac{1}{40\ell},$$
$$\mathbb{P}\left[\left|\frac{Y_{i,t}}{r\deg(t)} - \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)}\right| \ge \frac{\varepsilon}{8\ell}\right] \le \frac{1}{40\ell}.$$

Thus by a union bound, it holds that

$$\left|\hat{\delta}_{s,t} - \sum_{i=0}^{\ell-1} \chi_{s,t}\mathbf{P}^i \cdot \mathbf{D}^{-1}\chi_{s,t}^\top\right|$$
$$= \left|\sum_{i=0}^{\ell-1}\left(\frac{X_{i,s}}{r\deg(s)} - \frac{X_{i,t}}{r\deg(t)} - \frac{Y_{i,s}}{r\deg(s)} + \frac{Y_{i,t}}{r\deg(t)}\right)\right.$$
$$\left. - \sum_{i=0}^{\ell-1}\left(\frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_s^\top}{\deg(s)} - \frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)} - \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_s^\top}{\deg(s)} + \frac{\mathbf{1}_t\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)}\right)\right|$$
$$\le \frac{\varepsilon}{8\ell} \cdot \ell \cdot 4 = \frac{\varepsilon}{2}$$

with probability $1 - 4 \cdot \ell \cdot \frac{1}{40\ell} = \frac{9}{10}$. $\blacksquare$

Therefore, with probability at least $9/10$, it holds that

$$|R_G(s,t) - \hat{\delta}_{s,t}| \le \varepsilon.$$

This finishes the proof of Theorem 4.2.

The second algorithm: EstEff-TranProb-Collision. In the previous algorithm, we used the simple Monte Carlo approach to approximate the transition probabilities (which correspond to Line 4 and 5 in Algorithm 1). Now we give a more efficient procedure to estimate the transition probability $\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top$. Such an algorithm is based on the idea of treating the term $\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top$ (roughly) as a collision probability of two random walks of length $i/2$, starting from s and t, respectively. In particular, if $p = \mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top$, then for typical vertices s, t, we can approximates the probability p in $O(1/\sqrt{p})$ time, in contrast to the $O(1/p)$ time from the Monte Carlo approach. This idea of approximating transition probability has been given in [5]. We use this idea to present a new algorithm whose performance guarantee is given in the following theorem.

Theorem 4.6. *Suppose that Assumption 4.1 holds. Suppose further that for any* $i \le \ell$,

$$\|\mathbf{1}_s\mathbf{P}^i\mathbf{D}^{-1/2}\|_2^2, \|\mathbf{1}_t\mathbf{P}^i\mathbf{D}^{-1/2}\|_2^2 \le \beta_i,$$

for some parameters β_i*'s. The Algorithm 2 (i.e.,* EstEff-TranProb-Collision*(*G, s, t*)) outputs an estimate* $\hat{\delta}_{s,t}$ *such that with probability*

at least $9/10$*, it holds that*

$$|R_G(s,t) - \hat{\delta}_{s,t}| \le \varepsilon.$$

The running time and query complexity of the algorithm are $O(\sum_{i=0}^{\ell-1} r_i) = O(\frac{\ell^{3/2}}{\varepsilon}\sum_{i=0}^{\ell-1}\sqrt{\beta_i} + \frac{\ell^3}{\varepsilon^2}\sum_{i=0}^{\ell-1}\beta_i^{3/2})$.

On the choice of β_i*:* Note that the algorithm is parametrized by β_i's. We note that for expander graphs or graphs with low mixing time, it holds that β_i is a number that exponentially decreases in terms of i, i.e,. $\beta_i \le c^i$ for some constant $c < 1$, as long as ℓ is not too large. The reason is that in an expander graph G with $\phi(G) \ge \phi$ for some constant ϕ, it holds that $||\mathbf{1}_s\mathbf{P}^i\mathbf{D}^{-1/2}||_2^2 \le \frac{1}{\mathrm{vol}(V_G)} + (1-\frac{\phi^2}{4})^{2i}$ for any starting vertex s (see e.g., [12]). Therefore, in this case, the running time in Theorem 4.6 will be dominated by $O(\frac{\ell^3}{\varepsilon^2})$, which is faster than Algorithm 1.

Algorithm 2: EstEff-TranProb-Collision(G, ε, s, t)

$\ell \leftarrow \frac{\log(4/\varepsilon(1-\lambda))}{\log(1/\lambda)}$
for $i := 0, 1, \ldots, \ell-1$ **do**
  $r_i \leftarrow 20000(\sqrt{\frac{\ell^3\beta_i}{\varepsilon^2}} + \frac{\ell^3\beta_i^{3/2}}{\varepsilon^2})$
  Perform r_i independent random walks of length $i_1 := \lceil i/2 \rceil$ starting at s (resp., t), and let $\overrightarrow{X}_s \in \mathbb{R}^V$ (resp., $\overrightarrow{X}_t \in \mathbb{R}^V$) be a *row vector* whose v'th component is the fraction of random walks from s (resp., t) that end up at v, divided by $\sqrt{\deg(v)}$
  Perform r independent random walks of length $i_2 := \lfloor i/2 \rfloor$ starting at s (resp., t), and let $\overrightarrow{Y}_s \in \mathbb{R}^V$ (resp., $\overrightarrow{Y}_t \in \mathbb{R}^V$) be a *row vector* whose v'th component is the fraction of random walks from s (resp., t) that end up at v, divided by $\sqrt{\deg(v)}$
  Set $\hat{\delta}_{s,t}^{(i)} = \overrightarrow{X}_s \cdot \overrightarrow{Y}_s^\top - \overrightarrow{X}_s \cdot \overrightarrow{Y}_t^\top - \overrightarrow{X}_t \cdot \overrightarrow{Y}_s^\top + \overrightarrow{X}_t \cdot \overrightarrow{Y}_t^\top$
return $\hat{\delta}_{s,t} = \sum_{i=0}^{\ell-1} \hat{\delta}_{s,t}^{(i)}$

Proof of Theorem 4.6. W.l.o.g. we consider the case that the length i of the random walk is even. Note that for any s, t,

$$\begin{aligned}
\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top &= \mathbf{1}_s(\mathbf{D}^{-1}\mathbf{A})^i\mathbf{1}_t^\top = \mathbf{1}_s(\mathbf{D}^{-1}\mathbf{A})^{i/2}\mathbf{D}^{-1}\Big((\mathbf{D}^{-1}\mathbf{A})^\top\Big)^{i/2}\mathbf{D}\mathbf{1}_t^\top \\
&= \mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1}(\mathbf{P}^\top)^{i/2}\mathbf{D}\mathbf{1}_t^\top = \mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1}(\mathbf{P}^\top)^{i/2}\mathbf{D}\mathbf{1}_t^\top \\
&= \langle \mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1/2}, \mathbf{1}_t\mathbf{D}\mathbf{P}^{i/2}\mathbf{D}^{-1/2}\rangle \\
&= \deg(t)\cdot\langle \mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1/2}, \mathbf{1}_t\mathbf{P}^{i/2}\mathbf{D}^{-1/2}\rangle.
\end{aligned}$$

Thus,

$$\frac{\mathbf{1}_s\mathbf{P}^i\mathbf{1}_t^\top}{\deg(t)} = \langle \mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1/2}, \mathbf{1}_t\mathbf{P}^{i/2}\mathbf{D}^{-1/2}\rangle.$$

Note that for any vertex v, the quantity $[\mathbf{1}_s\mathbf{P}^{i/2}\mathbf{D}^{-1}](v)$ is the probability of a length-$(i/2)$ random walk that starts from s and ends at vertex v, divided by $\sqrt{\deg(v)}$; and the quantity $[\mathbf{1}_t\mathbf{P}^{i/2}](v)$ is the probability of a length-$(i/2)$ random walk that starts from s and ends at vertex v, divided by $\sqrt{\deg(v)}$.

Now we use the argument in the proof of Lemma 19 in [10]. Specifically, let $Z_{s,t} = \overrightarrow{X}_s \cdot \overrightarrow{Y}_t^\top$, where $\overrightarrow{X}_s \cdot \overrightarrow{Y}_t^\top$ are defined in Algorithm 2. Then $\mathbb{E}(Z_{s,t}) = (\mathbf{D}^{-1/2}\mathbf{P}^i\mathbf{1}_a)^\top(\mathbf{D}^{-1/2}P^i\mathbf{1}_a)$. By Chebyshev's inequality and Lemma 19 in [10], we get $\mathbb{P}[|Z_{s,t} - \mathbb{E}(Z_{s,t})| > \frac{\varepsilon}{8\ell}] < (\frac{8\ell}{\varepsilon})^2(\frac{\beta_i}{r_i^2} + \frac{2\beta_i^{3/2}}{r_i}) \le \frac{1}{40\ell}$, as we have chosen $r_i = 20000(\sqrt{\frac{\ell^3\beta_i}{\varepsilon^2}} + \frac{\ell^3\beta_i^{3/2}}{\varepsilon^2})$ in the algorithm. Then the statement of the theorem follows by analogous argument from the proof of Theorem 4.2. ∎

Finally, we remark that the success probabilities of both algorithms EstEff-TranProb and EstEff-TranProb-Collision can be boosted to $1 - \frac{1}{\mathrm{poly}(n)}$ by standard median trick, i.e., repeatedly run these algorithms $O(\log n)$ times and output the median. On graphs with bounded mixing time, which correspond to graphs such that $1-\lambda \ge \frac{1}{\mathrm{poly}(\log n)}$, the algorithms run in $O(\mathrm{poly}(\log n/\varepsilon))$ time.

4.2 Algorithms based on commute times of random walks

In this section, we provide two algorithms based on the well known connections between effective resistances and commute time/visiting probability in random walks. Let $\gamma > 0$ be a threshold parameter.

The first algorithm: EstEff-MC. We can use the commute time $\kappa(s,t)$ to approximate $R_G(s,t)$. We make use of the following results.

Lemma 4.7 ([9, 30])**.** *It holds that* $\kappa(s,t) = 2mR_G(s,t)$.

Lemma 4.8 (Proposition 2.3 in [26])**.** *The probability that a random walk starting at* s *visits* t *before returning to* s *is* $1/(\kappa(s,t)\cdot\pi(s))$*, where* $\pi(s) = \frac{\deg(s)}{2m}$ *is the stationary probability of* s.

We obtain the following corollary by Lemmas 4.7 and 4.8.

Corollary 4.9. *The probability* $p(s,t)$ *that a random walk starting at* s *visits* t *before returning to* s *is* $\frac{1}{R_G(s,t)\cdot\deg(s)}$*. In particular, if* $R_G(s,t) \le \gamma$*, then* $p(s,t) \ge \frac{1}{\gamma\cdot\deg(s)}$.

The corollary above suggests the Monte Carlo algorithm below. The algorithm performs a number of random walks, starting at vertex s. Then it essentially count how many times the random walk traverses from s to t and back.

Algorithm 3: EstEff-MC$(G, s, t, \gamma, \varepsilon)$

W.l.o.g. suppose that $\deg(s) \le \deg(t)$
$N_0 \leftarrow \frac{3\ln 6\cdot\gamma\cdot\deg(s)}{\varepsilon^2}$, $X \leftarrow 0$
for $i = 1, \ldots, N_0$ **do**
  Perform a random walk from s, and stop the walk
    (1) if the walk has visited t and then returns to s.
    (2) or if the walk has return to s before visiting t.
  If the item (1) occurs, $X \leftarrow X + 1$
return $\frac{N_0}{\deg(s)\cdot X}$

Theorem 4.10. *Assume that that $R_G(s,t) \le \gamma$. Let $N_0 = \frac{3\ln 6 \cdot \gamma \cdot \deg(s)}{\varepsilon^2}$. Then with probability 2/3, Algorithm 3 (i.e., EstEff-MC) returns an $(1+\varepsilon)$-approximation for $R_G(s,t)$. The running time of the algorithm is $O(\frac{m \cdot \deg(s) \cdot \gamma^2}{\varepsilon^2})$.*

We remark that the above algorithm runs in sublinear time if $\gamma = o_n(1)$, i.e., $R_G(s,t)$ is small enough. In other words, when the two vertices s, t are "similar" enough, our algorithm will be fast.

Proof of Theorem 4.10. In Algorithm 3, let X_i be the indicator variable that denotes the i-th random walk to be successful (where we do not abort the walk because of its length). Then $\mathbb{P}(X_i = 1) = p(s,t)$ where $p(s,t)$ is as defined in Corollary 4.9. Furthermore, let $X = \sum_{i=1}^{N_0} X_i$. Observe that $\mathbb{E}(X) = N_0 \cdot p(s,t) = \frac{N_0}{R_{\text{eff}}(s,t) \cdot \deg(s)}$, where we have used that $p(s,t) = \frac{1}{R_{\text{eff}}(s,t) \cdot \deg(s)}$.

Next, assume that $R_{\text{eff}}(s,t) \le \gamma$; let $N_0 = \frac{\ln(1/\delta) \cdot 3 \cdot \gamma \cdot \deg(s)}{\varepsilon^2} \ge \frac{\ln(1/\delta) \cdot 3 \cdot R_{\text{eff}}(s,t) \cdot \deg(s)}{\varepsilon^2}$, where $\delta > 0$ is a parameter that will be specified later. Using Chernoff and union bounds we find that $\mathbb{P}[|X - \mathbb{E}(X)| > \varepsilon' \cdot \mathbb{E}(X)] < 2 \cdot e^{-\frac{\varepsilon'^2 \cdot N_0}{3 \cdot R_{\text{eff}}(s,t) \cdot \deg(u)}} \le 2 \cdot \delta$ for any $\varepsilon' > 0$. Thus, we find that with probability at least $1 - 2\delta$, $(1 - 2\varepsilon') R_{\text{eff}}(s,t) \le \frac{N_0}{\deg(u) \cdot X} \le (1 + 2\varepsilon') R_{\text{eff}}(s,t)$. Now, choosing $\varepsilon' = \varepsilon/2$ yields the desired approximation ratio.

As a second step, we will show that each of the random walks in Algorithm 3 is expected to terminate within at most $2m\gamma$ steps. Consider the two cases in which the walks terminates. Let γ_i, $i \in \{1,2\}$ denote the number of steps taken in the random walk in some iteration, such that $i = 1$ if the first termination criterion of the loop is fulfilled and $i = 2$ in the other case. Then clearly, the number of steps taken in a random walk is $\min\{\gamma_1, \gamma_2\}$. Furthermore, it holds that $\min\{\gamma_1, \gamma_2\} \le \gamma_1$. Note that $\mathbb{E}(\gamma_1)$ is the commute time $\kappa(s,t)$. Then we find that $\mathbb{E}(\min\{\gamma_1, \gamma_2\}) \le \mathbb{E}(\gamma_1) = \kappa(s,t)$.

Finally, let $\delta = 1/3$. Then we find that Algorithm 3

- runs in expected time $R_{\text{eff}}(s,t) \cdot N_0 \in O(\frac{m \cdot \deg(s) \cdot \gamma^2}{\varepsilon^2})$ and
- with probability at least $1 - \delta = 2/3$, $\frac{N_0}{\deg(u) \cdot X}$ is an $(1+\varepsilon)$-approximation of $R_{\text{eff}}(s,t)$.

This concludes the proof. ∎

The second algorithm: EstEff-MC2. For the special case that there is an edge (s,t) between the two specified vertices s, t, we can also make use of the following probabilistic interpretation of effective resistance.

Lemma 4.11 ([31])**.** *Consider an edge (s,t). Then $R_G(s,t)$ is the probability that a random walk from s visits t for the first time using (s,t).*

This suggests the following Monte Carlo algorithm.

Theorem 4.12. *For $R_G(s,t) > \gamma$, Algorithm 4 (i.e., EstEff-MC2) returns with probability (1-δ) a $(1+\varepsilon)$-approximation of $R_G(s,t)$.*

The proof of the above theorem is deferred to Appendix B. Note that in contrast to Algorithm 3, the random walks in Algorithm 4 stop as soon as we have reached the destination vertex t. Hence, one can expect that Algorithm 4 runs faster than Algorithm 3. Experimental comparisons of the running times can be found in Section 5.1.

Algorithm 4: EstEff-MC2$(G, s, t, \varepsilon, \gamma, \delta)$

```
W.l.o.g. suppose that deg(s) ≤ deg(t)
M_0 ← ln(1/δ)·3 / (ε²·γ), X ← 0
for i = 1, ..., M_0 do
    Perform a random walk from s, and stop the walk
        (1) if the walk visits t for the first time using the edge (s,t)
        (2) or if the walk visits t for the first time using any other edge.
    If the item (1) occurs, X ← X + 1
return X / M_0
```

4.3 An algorithm based on estimating the number of spanning trees

Now we present a local algorithm based on a connection to the number of spanning trees of a graph. Let $T(G)$ denote the number of spanning trees of G.

Lemma 4.13 (Corollary 4.2 in [26])**.** *Let G be a graph and $s, t \in V$. Let G' be the graph obtained by identifying s and t. Then*

$$R_G(s,t) = \frac{T(G')}{T(G)}$$

Lyons and Oveis Gharan gave a local algorithm for estimating the number of spanning trees [28].

Lemma 4.14 (Corollary 1.2 in [28])**.** *Let $G = (V, E)$ be a graph. In the adjacent list model, together with knowledge of n and $|E|$, there exists a randomized algorithm that for any given $\varepsilon, \delta > 0$, outputs an estimate Z that approximates $\frac{\log T(G)}{|V|}$ within an additive error of ε, i.e., $\left|\frac{\log T(G)}{|V|} - Z\right| \le \varepsilon$ with probability at least $1 - \delta$, by using only $\tilde{O}(\varepsilon^{-5} + \varepsilon^{-2} \log^2 n) \log \delta^{-1}$ number of queries.*

This suggests the following algorithm based on estimating the number of spanning trees. As remarked in [28], the assumption of having the knowledge of $|E|$ in Algorithm 6 might not even be necessary.

Algorithm 5: AppNumST(G, ε, δ) [Algorithm 2 in [28]]

```
r ← ⌈90³ ε⁻³⌉
s ← Σ_{1≤t<2r} 1/t
N ← ⌈8 log(4/δ) s² / ε²⌉
for i = 1 ← N do
    Let x be a randomly chosen vertex of G.
    Sample 1 ≤ t < 2r with probability 1/st.
    Run a t-step lazy simple random walk from x, and let
      Y_i ← 𝕀[X_t = x]
Sample ⌈256 log(1/δ)(log n)²/ε²⌉ random vertices of G, and
  let W̃ be the average of the logarithm of twice the degree
  of sampled vertices.
return Z := −n⁻¹ log(4|E|) + W̃ − s(Σ_{i=1}^{N} Y_i)/N + s/n
```

Algorithm 6: EstEff-SpanTree$(G, \varepsilon, \delta, u, v)$

1 $a \leftarrow$ AppNumST$(G, \frac{\varepsilon}{2}, \frac{\delta}{2})$
2 $b \leftarrow$ AppNumST$(G, \frac{\varepsilon}{2}, \frac{\delta}{2})$
3 **return** $\frac{e^{a(n-1)}}{e^{bn}}$

Theorem 4.15. *Algorithm 6 returns with probability at least* $1 - \delta$ *an estimator* X *such that*

$$e^{-\varepsilon n} R_G(s,t) \leq X \leq e^{\varepsilon n} R_G(s,t).$$

The algorithm uses $\tilde{O}(\varepsilon^{-5} + \varepsilon^{-2} \log^2 n) \log \delta^{-1}$ *queries.*

We give the proof of Theorem 4.15 in Appendix B and remark that the above algorithm seems of theoretical interest only, as it does not perform well in practice.

5 EXPERIMENTS

In this section, we show our experimental results. The experiments were conducted on a Linux server with Intel Xeon E5-2643 (3.4GHz) and 768GB of main memory, and all the programs were implemented in C++ and compiled with g++ 4.8.4. The graphs used in the experiments are taken from SNAP[3] and basic information about the graphs is given in Table 1. We generated query pairs by randomly sampling edges 1,000 times with replacements.

Table 1: Datasets

	n	m
Facebook	4,039	88,233
DBLP	317,080	1,049,869
YouTube	1,134,891	2,987,627

We implemented the following algorithms:

- Exact: This method first applies the QR decomposition to the Laplacian as preprocessing and computes effective resistance according to its definition, i.e., $R_G(s,t) := \chi_{s,t} \mathbf{L}^{\dagger} \chi_{s,t}^{\top}$.
- HAY [17]: This method computes effective resistances of all the edges at once by sampling spanning trees and it is still. state-of-the-art for this problem. We fixed the number of sampled spanning trees to 10,000.
- TP: Implementation of Algorithm 1. We set $\varepsilon = \lambda = 0.1$.
- TP-C: Implementation of Algorithm 2. We set $\varepsilon = \lambda = 0.1$.
- MC: Implementation of Algorithm 3. We set $\varepsilon = \gamma = 0.1$.
- MC2: Implementation of Algorithm 4. We set $\varepsilon = \gamma = 0.1$.
- ST: Implementation of Algorithm 6. We set $\varepsilon = 0.1$.

To implement each algorithm, we used the same number of random walks as the one given in the corresponding pseudocode.

5.1 Running time

Figure 1 shows the running time of each method. For our methods, we plotted the running time for the 1,000 queries in increasing order. For Exact and HAY, we plotted their preprocessing time. We do not show the running time of Exact on DBLP and YouTube because it did not terminate in 8 hours.

We can first observe that MC and ST on Facebook are as slow as previous (polynomial-time) algorithms, and hence we do not consider those algorithms for other graphs.

We can observe that TP, TP-C, and MC2 are much faster than the existing methods. Note that the running time of MC2 depends on the queried edge (s,t) because it runs until the random walk starting at s reaches t. In contrast, the running time of TP and TP-C is almost independent of the queried edge, which is preferable. A reason that TP-C is slower than TP is that we need to compute inner products in TP-C (Line 6 of Algorithm 2).

5.2 Accuracy

Figure 2 shows the accuracy of existing and our methods. For each method, we computed the relative error as $|R - \tilde{R}|/R$ for each query, where R is the exact effective resistance for Facebook using Exact and the one estimated by HAY for DBLP and YouTube, and $\tilde{R}$ is the estimated effective resistance. Then, we plotted the 1,000 relative errors after sorting them in increasing order. Except for ST, the relative error of our methods are within 0.1 for most of the queries, as expected from the choice $\varepsilon = 0.1$. Also, the results for Facebook justifies the use of HAY on DBLP and YouTube as the baseline method. In Figure 2(a), the results of TP and TP-C are very close such that their lines overlap. The fact that the lines change concavity twice appears to be a universal phenomenon for probabilistic distributions.

In Figure 3, each blue point represents $(R, \tilde{R})$ for a query, where R and $\tilde{R}$ are as we defined in the paragraph above. MC2 shows the best accuracy on DBLP and YouTube. Accuracy of TP-C is comparable to that of TP. Recalling that TP runs faster than TP-C, we can conclude that TP is superior to TP-C.

6 CONCLUSION

In this paper, we developed a number of local algorithms for estimating the pairwise effective resistances, a fundamental metric for measuring the similarity of vertices in a graph. Our algorithms explore only a small portion of the graph while provides a good approximation to $R_G(s,t)$ for any specified s, t. Our algorithms are desirable in applications where the effective resistances of a small number of vertex pairs are needed. Our experiments on benchmark datasets validate the performance of these local algorithms.

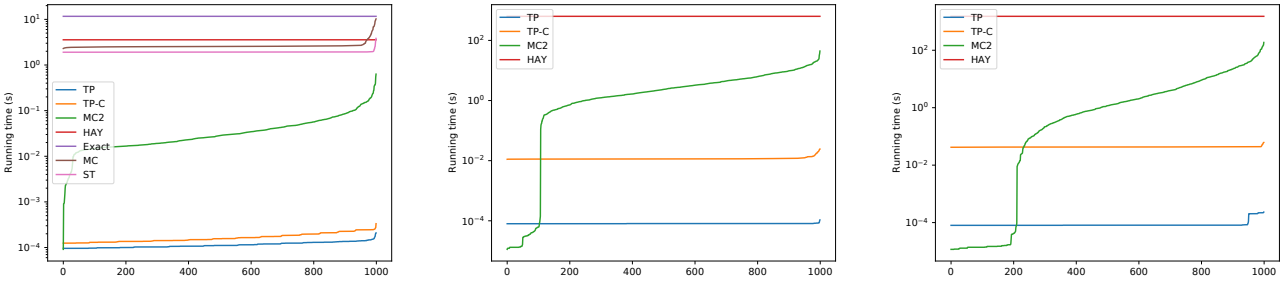
ACKNOWLEDGMENTS

Y.Y. is supported in part by JSPS KAKENHI Grant Number 17H04676, 18H05291, and 20H05965.

REFERENCES

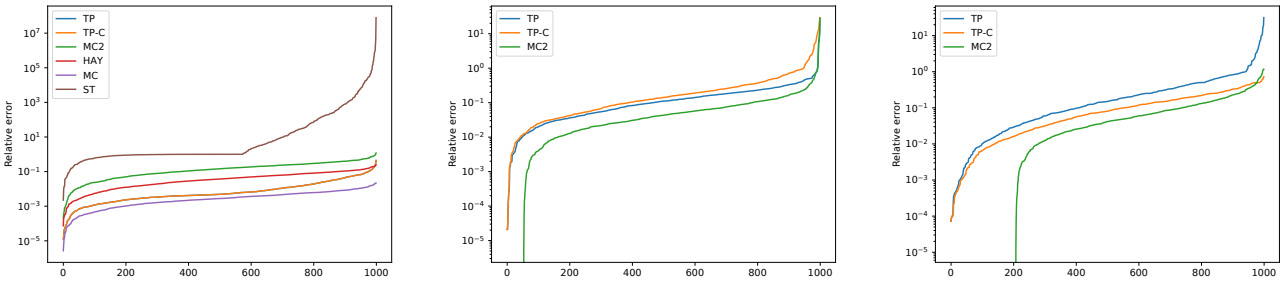
[1] Tasweer Ahmad, Lianwen Jin, Luojun Lin, and GuoZhi Tang. 2021. Skeleton-based action recognition using sparse spatio-temporal GCN with edge effective resistance. *Neurocomputing* 423 (2021), 389 – 398.
[2] Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. 2018. Graph Clustering using Effective Resistance. In *9th Innovations in Theoretical Computer Science Conference (ITCS)*, Vol. 94. 41:1–41:16.
[3] Nima Anari and Shayan Oveis Gharan. 2015. Effective-Resistance-Reducing Flows, Spectrally Thin Trees, and Asymmetric TSP. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*.

[3] https://snap.stanford.edu



(a) Facebook **(b) DBLP** **(c) YouTube**

Figure 1: Running time



(a) Facebook **(b) DBLP** **(c) YouTube**

Figure 2: Relative error

[4] Alexandr Andoni, Robert Krauthgamer, and Yosef Pogrow. 2018. On Solving Linear Systems in Sublinear Time. In *10th Innovations in Theoretical Computer Science Conference (ITCS)*.

[5] Siddhartha Banerjee and Peter Lofgren. 2015. Fast Bidirectional Probability Estimation in Markov Models. In *Advances in Neural Information Processing Systems (NIPS)*. 1423–1431.

[6] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Shang-Hua Teng. 2012. A sublinear time algorithm for pagerank computations. In *International Workshop on Algorithms and Models for the Web-Graph (WAW)*. Springer, 41–53.

[7] Marco Bressan, Enoch Peserico, and Luca Pretto. 2018. Sublinear algorithms for local graph centrality estimation. In *Proceedings of the IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 709–718.

[8] Marco Bressan, Enoch Peserico, and Luca Pretto. 2019. On approximating the stationary distribution of time-reversible Markov chains. *Theory of Computing Systems* (2019), 1–23.

[9] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. 1996. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity* 6, 4 (1996), 312–340.

[10] Ashish Chiplunkar, Michael Kapralov, Sanjeev Khanna, Aida Mousavifar, and Yuval Peres. 2018. Testing Graph Clusterability: Algorithms and Lower Bounds. (2018). arXiv:1808.04807

[11] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. 2011. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing (STOC)*. 273–282.

[12] Fan R. K Chung. 1997. *Spectral Graph Theory*. American Mathematical Society.

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

[14] Devdatt P Dubhashi and Alessandro Panconesi. 2009. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press.

[15] Wendy Ellens, FM Spieksma, P Van Mieghem, A Jamakovic, and RE Kooij. 2011. Effective graph resistance. *Linear algebra and its applications* 435, 10 (2011), 2491–2506.

[16] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[17] Takanori Hayashi, Takuya Akiba, and Yuichi Yoshida. 2016. Efficient algorithms for spanning tree centrality. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. 3733–3739.

[18] Arun Jambulapati and Aaron Sidford. 2018. Efficient $\tilde{O}(n/\varepsilon)$ Spectral Sketches for the Laplacian and its Pseudoinverse. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2487–2503.

[19] Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 538–543.

[20] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[21] Heung-Nam Kim and Abdulmotaleb El Saddik. 2011. Personalized pagerank vectors for tag recommendations: inside folkrank. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*. 45–52.

[22] Jérôme Kunegis and Stephan Schmidt. 2007. Collaborative filtering using electrical resistance network models. In *Industrial Conference on Data Mining*. Springer, 269–282.

[23] Christina E Lee, Asuman Ozdaglar, and Devavrat Shah. 2013. Computing the stationary distribution locally. In *Advances in Neural Information Processing Systems*. 1376–1384.

[24] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. 2016. Personalized pagerank estimation and search: A bidirectional approach. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 163–172.

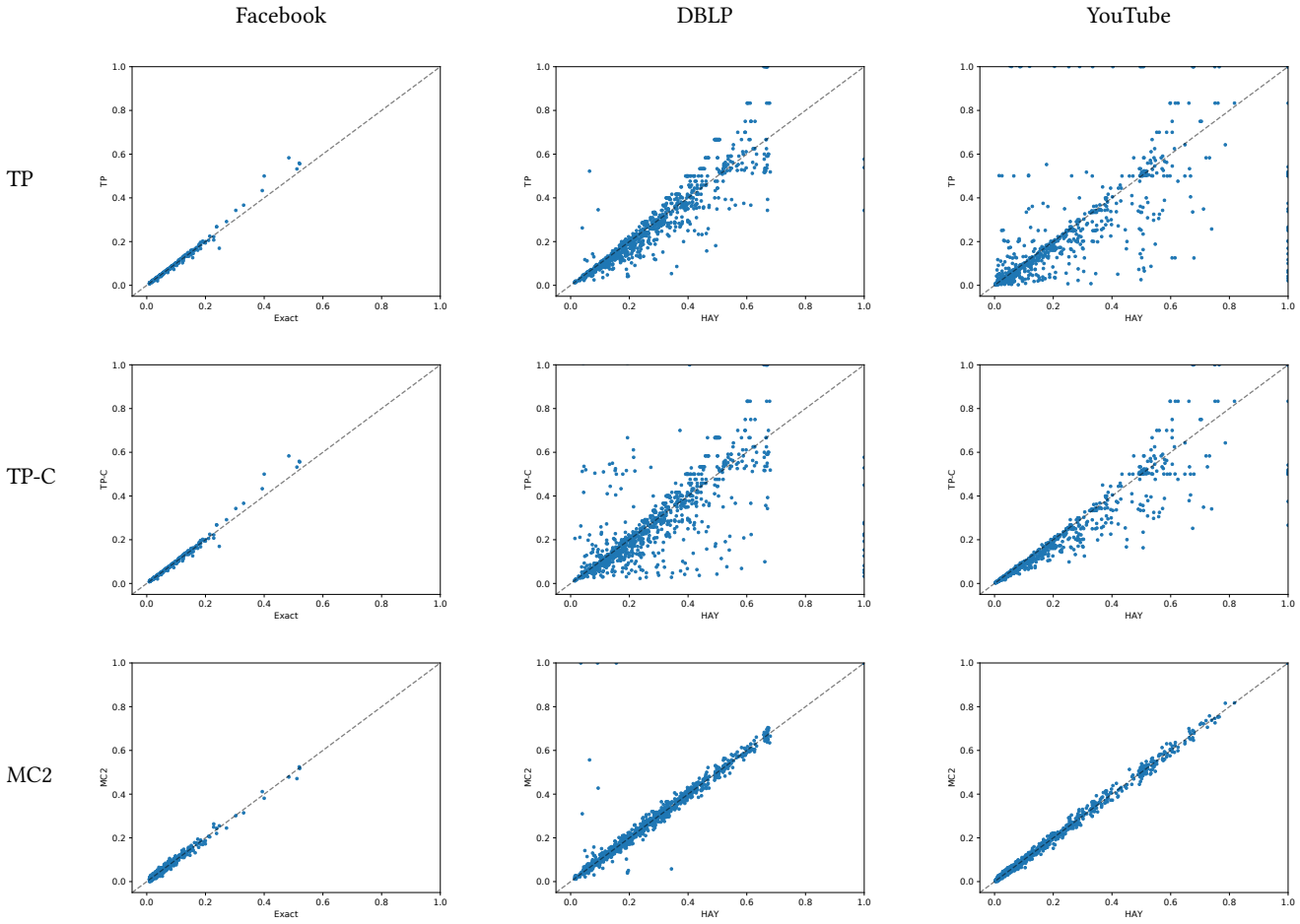


Figure 3: Quality of estimated effective resistance

[25] Peter A Lofgren, Siddhartha Banerjee, Ashish Goel, and C Seshadhri. 2014. FAST-PPR: scaling personalized pagerank estimation for large graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1436–1445.

[26] László Lovász et al. 1993. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty* 2, 1 (1993), 1–46.

[27] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[28] Russell Lyons and Shayan Oveis Gharan. 2017. Sharp bounds on random walk eigenvalues via spectral embedding. *International Mathematics Research Notices* 2018, 24 (2017), 7555–7605.

[29] Aleksander Madry, Damian Straszak, and Jakub Tarnawski. 2014. Fast Generation of Random Spanning Trees and the Effective Resistance Metric. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[30] C St JA Nash-Williams. 1959. Random walk and electric currents in networks. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 55. Cambridge University Press, 181–194.

[31] Shayan Oveis Gharan. 2015. Lecture 4-5: Effective Resistance and Simple Random Walks. *https://homes.cs.washington.edu/~shayan/courses/approx/adv-approx-4.pdf* (2015).

[32] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[33] Dana Ron. 2019. Sublinear-time algorithms for approximating graph parameters. In *Computing and Software Science*. Springer, 105–122.

[34] Alistair Sinclair. 1992. Improved bounds for mixing rates of Markov chains and multicommodity flow. In *Latin American Symposium on Theoretical Informatics (LATIN)*. 474–487.

[35] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. 2009. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC)*. 322–335.

[36] Daniel A Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.

[37] Haoran Yu, Ermin Wei, and Randall A. Berry. 2019. Analyzing Location-Based Advertising for Vehicle Service Providers Using Effective Resistances. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1, Article 6 (2019), 35 pages.

A THE CHERNOFF-HOEFFDING BOUND

We make use of the following Chernoff-Hoeffding bound (see Theorem 1.1 in [14]).

Theorem A.1 (The Chernoff-Hoeffding bound). *Let $s \geq 1$. Let $X := \sum_{1 \leq i \leq s} X_i$, where X_i, $1 \leq i \leq s$, are independently distributed in $[0, 1]$. Then for all $t > 0$,*

$$\mathbb{P}[|X - \mathbb{E}[X]| > t] \leq e^{-2t^2/s}.$$

B MISSING PROOFS OF SECTION 4

We present here the proofs of two theorems in Section 4.

Proof of Theorem 4.12. In Algorithm 4, let X_i be the indicator variable that denotes the i-th random walk to be successful (regardless of its length). Then $\mathbb{P}(X_i = 1) = R_G(s,t)$. Furthermore, let $X = \sum_1^{M_0} X_i$. Observe that $\mathbb{E}(X) = M_0 \cdot R_G(s,t)$.

Next, assume that $R_G(s,t) > \gamma$; let $M_0 = \frac{\ln(1/\delta')\cdot 3}{\varepsilon^2\cdot\gamma} > \frac{\ln(1/\delta')\cdot 3}{\varepsilon^2\cdot R_G(s,t)}$. Using Chernoff and union bounds we find that $\mathbb{P}[|X - \mathbb{E}(X)| > \varepsilon\cdot\mathbb{E}(X)] < 2\cdot e^{-\frac{\varepsilon^2 M_0 R_G(s,t)}{3}} < 2\cdot\delta'$ for any $\varepsilon, \delta' > 0$. Thus, we find that with probability at least $1 - 2\delta'$, $(1-\varepsilon)R_G(s,t) \le X/M_0 \le (1+\varepsilon)R_G(s,t)$. Now, choosing $\delta' = \frac{\delta}{2}$ yields the desired approximation ratio.

Proof of Theorem 4.15. By Lemma 4.14, with probability $1-\frac{\delta}{2}$, the a returned in line 1 of Algorithm 6 satisfies $|a - \frac{\log(T(G'))}{n-1}| \le \frac{\varepsilon}{2}$. Similarly, with the same probability, $|b - \frac{\log(T(G))}{n}| \le \frac{\varepsilon}{2}$. By the union bound, this implies that with probability $1-\delta$, $a(n-1) - bn \le \frac{\varepsilon}{2}(2n-1) - \log(T(G)) + \log T((G'))$. Thus, we find that $\frac{e^{a(n-1)}}{e^{bn}} \le e^{\varepsilon n}\cdot\frac{T(G')}{T(G)}$. Similarly, it holds that $\frac{e^{a(n-1)}}{e^{bn}} \ge e^{-\varepsilon n}\cdot\frac{T(G')}{T(G)}$.

Let $X = \frac{e^{a(n-1)}}{e^{bn}}$. Then by Lemma 4.13, $e^{-\varepsilon n}R_G(s,t) \le X \le e^{\varepsilon n}R_G(s,t)$. This yields the desired approximation ratio. ∎