Mixing Time Matters: Accelerating Effective Resistance Estimation via Bidirectional Method

Guanyu Cui [1] Hanzhi Wang [2] Zhewei Wei* [1]

[1]Renmin University of China [2]BARC, University of Copenhagen

August 3–7, 2025 KDD2025

Effective Resistance (ER)

- A node proximity metric in undirected graphs.
- Applied in **spectral sparsification**, **clustering**, **network robustness**, **influence maximization**, **GNN rewiring**, etc.
- Given an undirected graph G and nodes s, t, the effective resistance $R(s,t)$ is the resistance between s and t when each edge is a 1-ohm resistor (Figure 1).

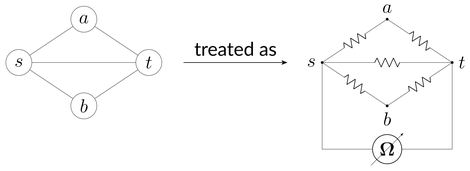


Figure 1. Source of Definition of Effective Resistance

- From physics and graph theory, $R(s,t) = (\boldsymbol{e}_s - \boldsymbol{e}_t)^\top \boldsymbol{L}^\dagger (\boldsymbol{e}_s - \boldsymbol{e}_t)$, where $\boldsymbol{e}_s$ is the one-hot vector of node s, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is the Laplacian, and $\boldsymbol{L}^\dagger$ is its Moore-Penrose pseudoinverse.

Problem Definition

We study single-pair effective resistance (SPER) estimation with an absolute error guarantee:

Single-Pair Effective Resistance (SPER) Estimation with Absolute Error Guarantee

Given a connected undirected graph $G = (V, E)$, nodes $s, t \in V$, absolute error tolerance $\epsilon > 0$, and failure probability $0 < p_f \leq 1$, find an estimator $\hat{R}(s,t)$ such that:

$$\Pr\left(\left|\hat{R}(s,t) - R(s,t)\right| < \epsilon\right) \geq 1 - p_f.$$

Related Works

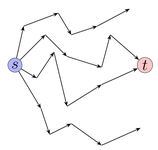
SPER estimation methods can be categorized into four types:

- **Transition-Probabilities-Based**: reformulates $R(s,t)$ as a series of multi-step transition probabilities:

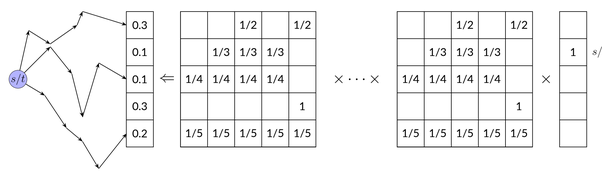
$$R(s,t) = \sum_{\ell=0}^{\infty} \left(\frac{p^{(\ell)}(s,s)}{d(s)} - \frac{p^{(\ell)}(s,t)}{d(t)} - \frac{p^{(\ell)}(t,s)}{d(s)} + \frac{p^{(\ell)}(t,t)}{d(t)} \right),$$

then truncates the series at $L_{\max}$ (denoted as $R_{L_{\max}}(s,t)$) and estimates the probabilities. Representative methods:

1. **EstEff-TranProb** [Peng et al., KDD'21] and **AMC** [Yang et al., SIGMOD'23]: use Monte Carlo to sample a batch of random walks



2. **GEER** [Yang et al., SIGMOD'23]: combines power iteration and random walks



- **Landmark-Based**: reformulates $R(s,t)$ using hitting probabilities. Includes single-landmark methods [Liao et al., SIGMOD'23] and multi-landmark methods [Liao et al., SIGMOD'24]. Cannot set algorithm parameters to achieve an error guarantee.
- **Commute-Time-Based**: estimates commute-time-based formulations of $R(s,t)$. E.g., **EstEff-MC** [Peng et al., KDD'21].
- **Laplacian-Solver-Based**: solves $\boldsymbol{L}\boldsymbol{x} = (\boldsymbol{e}_s - \boldsymbol{e}_t)$ and computes $R(s,t)$. Theoretically sound but challenging to implement in practice.

Related Works (Cont'd)

Table 1 summarizes time complexity of related works.

Table 1. Time complexity of the algorithms.

Method	Query Time
EstEff-TranProb [Peng et al., KDD'21]	$\tilde{O}\left(\frac{L_{\max}^4}{\epsilon^2}\right)$
AMC / GEER [Yang et al., SIGMOD'23]	$\tilde{O}\left(\frac{L_{\max}^3}{\epsilon^2 d^2}\right)$
EstEff-MC [Peng et al., KDD'21]	$\tilde{O}\left(\frac{m}{(1-\lambda_2)^2 \epsilon^2 d}\right)$
Laplacian Solvers	$\tilde{O}(m)$
BiSPER **(Ours)**	$\tilde{O}\left(\min\left\{\frac{L_{\max}^{7/3}}{\epsilon^{2/3}}, \frac{L_{\max}^3}{\epsilon^2 d^2}, mL_{\max}\right\}\right)$

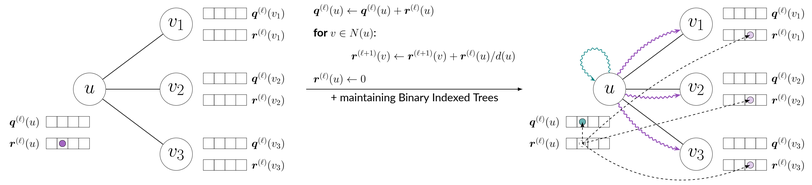
Algorithm

Overview

- **Forward Push with Binary Indexed Trees (BITs) + Backward Adaptive Monte Carlo**.
- For each node $u \in V$ and step $0 \leq \ell \leq L_{\max}$, maintain two quantities: reserves $\boldsymbol{q}_s^{(\ell)}(u), \boldsymbol{q}_t^{(\ell)}(u)$, which capture accumulated probability mass, and residues $\boldsymbol{r}_s^{(\ell)}(u), \boldsymbol{r}_t^{(\ell)}(u)$, which represent the remaining probability to be propagated to the next layer.

Forward Push with Binary Indexed Trees

- Attach a Binary Indexed Tree (BIT) to the reserve and residue vectors of each node to dynamically maintain their prefix sums.
- First, set $\boldsymbol{r}_s^{(0)}(s) = 1$ and $\boldsymbol{r}_t^{(0)}(t) = 1$, with all other values initialized to zero. Then, for each layer $\ell = 0, 1, \cdots, L_{\max}$, and for any node u whose degree-normalized residue $\boldsymbol{r}_s^{(\ell)}(u)/d(u)$ or $\boldsymbol{r}_t^{(\ell)}(u)/d(u)$ exceeds a threshold $r_{\max}$, invoke the following Forward Push procedure:



Backward Adaptive Monte Carlo

- Sample N $L_{\max}$-step random walks from s and t and construct an estimator.
- For the i-th random walk, let the sampled nodes be $s = v_{s,i}^{(0)}, v_{s,i}^{(1)}, \cdots, v_{s,i}^{(L_{\max})}$, $t = v_{t,i}^{(0)}, v_{t,i}^{(1)}, \cdots, v_{t,i}^{(L_{\max})}$, and let

$$\hat{R}_{L_{\max}}(s,t) = \sum_{\ell=0}^{L_{\max}} \left(\frac{\boldsymbol{q}_s^{(\ell)}(s)}{d(s)} - \frac{\boldsymbol{q}_s^{(\ell)}(t)}{d(t)} \right) + \sum_{\ell=0}^{L_{\max}} \left(\frac{\boldsymbol{q}_t^{(\ell)}(t)}{d(t)} - \frac{\boldsymbol{q}_t^{(\ell)}(s)}{d(s)} \right)$$
$$+ \frac{1}{N} \sum_{i=1}^{N} \sum_{\ell=0}^{L_{\max}} \underbrace{\left(\sum_{k=0}^{L_{\max}-\ell} \frac{\boldsymbol{r}_s^{(k)}(v_{s,i}^{(\ell)})}{d(v_{s,i}^{(\ell)})} - \sum_{k=0}^{L_{\max}-\ell} \frac{\boldsymbol{r}_t^{(k)}(v_{t,i}^{(\ell)})}{d(v_{t,i}^{(\ell)})} \right)}_{\text{query BIT}} + \frac{1}{N} \sum_{i=1}^{N} \sum_{\ell=0}^{L_{\max}} \underbrace{\left(\sum_{k=0}^{L_{\max}-\ell} \frac{\boldsymbol{r}_t^{(k)}(v_{t,i}^{(\ell)})}{d(v_{t,i}^{(\ell)})} - \sum_{k=0}^{L_{\max}-\ell} \frac{\boldsymbol{r}_s^{(k)}(v_{s,i}^{(\ell)})}{d(v_{s,i}^{(\ell)})} \right)}_{\text{query BIT}}.$$

Theoretical Results

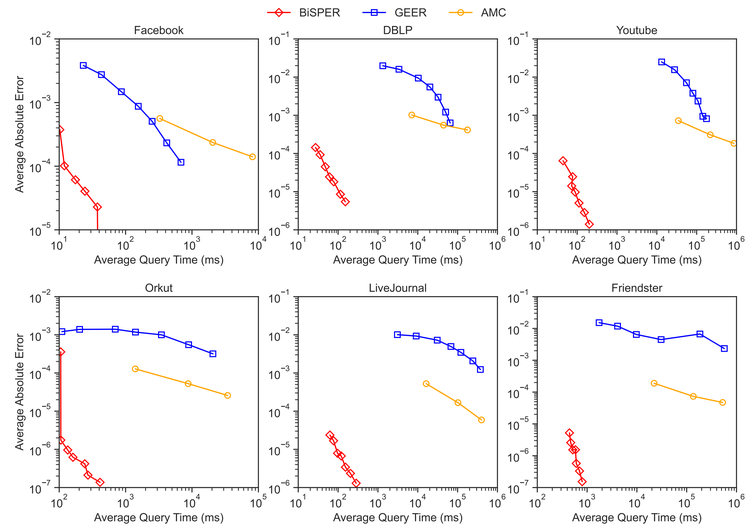
- We derive the following two theorems through refined analysis:
- **Theorem 1 (Correctness).** The estimator $\hat{R}_{L_{\max}}(s,t)$ is an unbiased estimator of $R_{L_{\max}}(s,t)$ and satisfies the absolute error guarantee.
- **Theorem 2 (Time Complexity).** The **worst-case** time complexity of our algorithm is $\tilde{O}\left(\min\left\{\frac{L_{\max}^{7/3}}{\epsilon^{2/3}}, \frac{L_{\max}^3}{\epsilon^2 d^2}, mL_{\max}\right\}\right)$.

Experiments

We randomly fix 100 source-target node pairs from each dataset and evaluate the performance of each algorithm on these pairs.

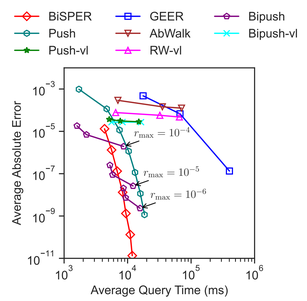
Experiment I: Query Efficiency for $R_{L_{\max}}(s,t)$ on Real-World Graphs

- Set $L_{\max} = 100$, and evaluate each algorithm's performance in estimating $R_{L_{\max}}(s,t)$;
- This reflects the need for a large $L_{\max}$ to accurately approximate SPER. Since computing ground-truth on large graphs via Power Iteration is infeasible, we rely on this setting.



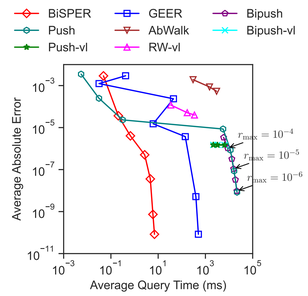
Experiment II: Query Efficiency for $R(s,t)$ on Real-World Graphs

- For small graphs, $L_{\max}$ can be set sufficiently large to closely approximate $R(s,t)$.



Experiment III: Query Efficiency for $R(s,t)$ on Synthetic Graphs

- Conduct Experiment II on a synthetic Erdős-Rényi random graph with parameters $(n, p) = (5000, 0.005)$.



*Zhewei Wei is the corresponding author. Contact us: cuiguanyu@ruc.edu.cn; zhewei@ruc.edu.cn