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

Introduction: Effective Resistance

- Effective Resistance (ER) originates from the analysis of electric circuits in physics.
- Given an undirected G, two nodes s and t, $R(s,t)$ is defined as the resistance between s and t when each edge is treated as a one-ohm resistor.

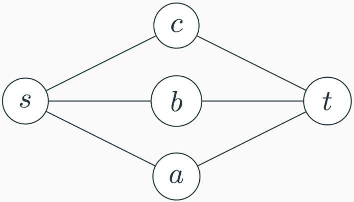
- Based on physical laws and graph theory, ER can be defined as follows:

$$R(s,t) = (\boldsymbol{e}_s - \boldsymbol{e}_t)^\top \boldsymbol{L}^\dagger (\boldsymbol{e}_s - \boldsymbol{e}_t).$$

Introduction: Effective Resistance

- ER serves as a **proximity** metric on **undirected** graphs;
- Intuitively, by the principles of series and parallel circuits, a larger $R(s,t)$ implies **fewer paths** and **weaker connectivity**.

- Conversely, a smaller $R(s,t)$ suggests more paths and stronger connectivity.



Introduction: Applications of Effective Resistance

- ER finds applications across many areas, including:
- **Theoretical Reasearch**: optimal transport [Robertson et al., arXiv'24], maximum flow [Christiano et al., STOC'11] and clustering [Alev et al., ITCS'18];
- **Data Mining**: influence maximization [Hong et al., COMPLEX NETWORKS'23], network robustness analysis [Yamashita et al., ICOIN'21];
- **Graph Machine Learning**, and **Graph Neural Networks**: graph rewiring [Black et al., ICML'23] and added to GNNs to enhance performance [Zhang et al., ICLR'23];

Research Focus

- We focus on estimating the **single-pair effective resistance** (SPER) with an absolute error guarantee:

Definition(SPER Estimation with Absolute Error Guarantee)

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

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

Existing Algorithms

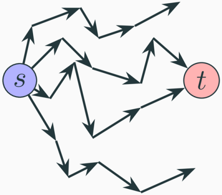
- **Transition-Probabilities-Based**: express $R(s,t)$ as a series of multi-step transition probabilities:

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

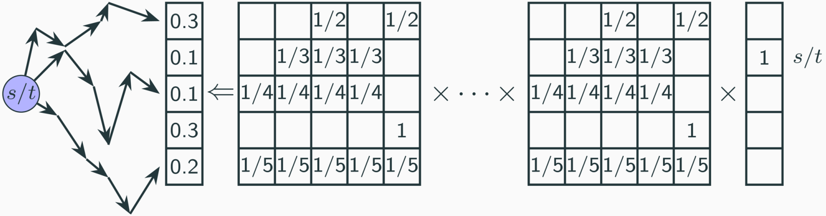
  then truncate the series at some $L_{\max}$ (denoted as $R_{L_{\max}}(s,t)$) and estimate the probabilities.

Existing Algorithms

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



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



Existing Algorithms

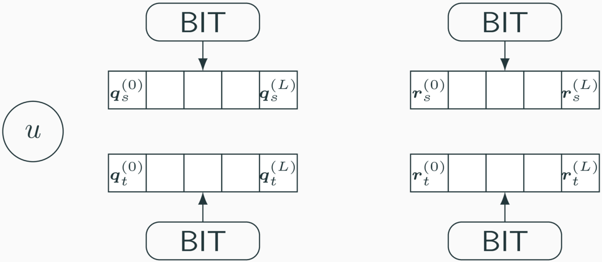
- **Landmark-Based**: reformulates $R(s,t)$ using hitting probabilities.
  - Includes **single-landmark** methods [Liao et al., SIGMOD'23] and **multi-landmark** methods [Liao et al., SIGMOD'24].
  - **Limitation:** Cannot set parameters to achieve an error guarantee.
- **Commute-Time-Based**: estimates 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 ER. Theoretically sound but challenging to implement in practice.

Our Algorithm

- Our approach: improves upon Transition-Probabilities-Based algorithms;
- Core idea: combine **Forward Push using Binary Indexed Trees** (BITs) with a **Backward Adaptive Monte Carlo** phase;

Algorithm: Preparation

- For each node $u \in V$ and step $0 \leq \ell \leq L_{\max}$, maintain two types of quantities:
  - **reserves** $\boldsymbol{q}_s^{(\ell)}(u), \boldsymbol{q}_t^{(\ell)}(u)$: accumulated probability mass;
  - **residues** $\boldsymbol{r}_s^{(\ell)}(u), \boldsymbol{r}_t^{(\ell)}(u)$, probability mass yet to be propagated to the next layer.
- Attach a **Binary Indexed Tree** (BIT) to each node's reserve and residue vectors;
  - BIT is a data structure that dynamically maintains prefix sums of an array.



Algorithm: Forward Push Phase

- Initialize $\boldsymbol{r}_s^{(0)}(s) = 1$ and $\boldsymbol{r}_t^{(0)}(t) = 1$, set the rest to zero.
- Then, for layer $\ell = 0, 1, \cdots, L_{\max}$, 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:

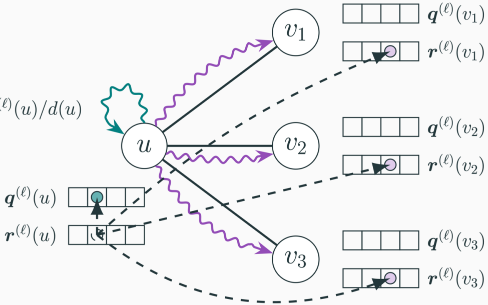
$\boldsymbol{q}^{(\ell)}(u) \leftarrow \boldsymbol{q}^{(\ell)}(u) + \boldsymbol{r}^{(\ell)}(u)$

for $v \in N(u)$:

  $\boldsymbol{r}^{(\ell+1)}(v) \leftarrow \boldsymbol{r}^{(\ell+1)}(v) + \boldsymbol{r}^{(\ell)}(u)/d(u)$

$\boldsymbol{r}^{(\ell)}(u) \leftarrow 0$

+ maintaining BITs



Algorithm: Monte Carlo Phase

- A key insight: following **invariant** holds throughout the Forward Push phase [Modified from Banerjee et al., NIPS'15, and Lofgren et al., WSDM'16]:

$$p^{(\ell)}(s,t) = \boldsymbol{q}_s^{(\ell)}(t) + d(t) \sum_{k=0}^{\ell} \sum_{v \in V} \frac{\boldsymbol{r}_s^{(\ell-k)}(v)}{d(v)} p^{(k)}(t,v).$$

- We can sample random walks to estimate $p^{(k)}(t,v)$, which allows us to construct an estimator for the truncated ER $R_{L_{\max}}(s,t)$.

Algorithm: Monte Carlo Phase

- Sample multiple $L_{\max}$-step random walks from both s and t.
- For the i-th walk, denote the sampled nodes be $s = v_{s,i}^{(0)}, v_{s,i}^{(1)}, \cdots, v_{s,i}^{(L_{\max})}$, $t = v_{t,i}^{(0)}, v_{t,i}^{(1)}, \cdots, v_{t,i}^{(L_{\max})}$.
- The estimator is defined as follows:

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

Theoretical Analysis

- First, our estimator $\hat{R}_{L_{\max}}(s,t)$ is **unbiased** for the truncated ER $R_{L_{\max}}(s,t)$ and satisfies the error guarantee.
- Then, through a refined analysis, we derive the **worst-case time complexity** of our **BiSPER** algorithm

$$\tilde{O}\left(\min\left\{\frac{L_{\max}^3}{\epsilon^2 d^2}, \frac{L_{\max}^{7/3}}{\epsilon^{2/3}}, mL_{\max}\right\}\right).$$

Theoretical Analysis

- Comparision with other algorithms:

Method	Query Time
EstEff-TranProb [Peng et al., KDD 2021]	$\tilde{O}\left(\frac{L_{\max}^4}{\epsilon^2}\right)$
AMC / GEER [Yang et al., SIGMOD 2023]	$\tilde{O}\left(\frac{L_{\max}^3}{\epsilon^2 d^2}\right)$
EstEff-MC [Peng et al., KDD 2021]	$\tilde{O}\left(\frac{m}{(1-\lambda_2)^2\epsilon^2 d}\right)$
Laplacian Solvers	$\tilde{O}\left(m\right)$
BiSPER **(Ours)**	$\tilde{O}\left(\min\left\{\frac{L_{\max}^{7/3}}{\epsilon^{2/3}}, \frac{L_{\max}^3}{\epsilon^2 d^2}, mL_{\max}\right\}\right)$

Experiments

- We tested three scenarios on **six real-world** datasets and **one Erdős-Rényi random graph** with parameters $(n, p) = (5000, 0.005)$;
- Graph statistics:

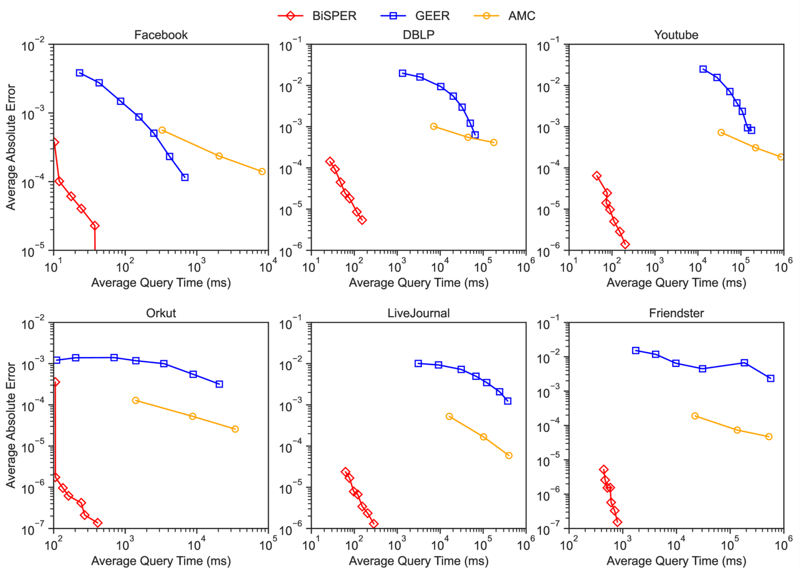
Name	n	m	$d_{\min}$	$d_{\max}$	$\bar{d}$	λ
Facebook	4,039	88,234	1	1045	43.69	0.9992
DBLP	317,080	1,049,866	1	343	6.62	0.9973
Youtube	1,134,890	2,987,624	1	28754	5.27	0.9980
Orkut	3,072,441	117,185,083	1	33313	76.28	0.9948
LiveJournal	3,997,962	34,681,189	1	14815	17.35	0.9999
Friendster	65,608,366	1,806,067,135	1	5214	55.06	0.9995

- For each dataset, we randomly selected 100 node pairs to evaluate the performance of each algorithm.

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

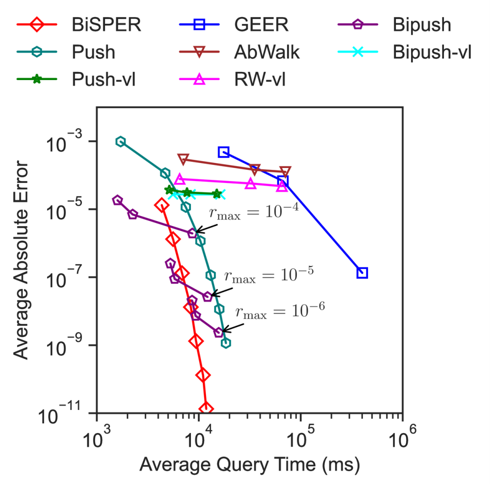
- Set $L_{\max} = 100$ and measure algorithms' query efficiency for truncated ER $R_{L_{\max}}(s,t)$ on real-world graphs;
- Accurately approximating SPER requires large $L_{\max}$ values $(10^3 \sim 10^4)$, and computing the ground-truth on large graphs via Power Iteration is impractical.

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



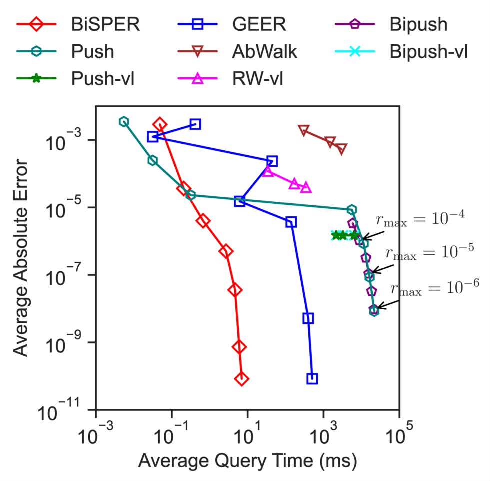
Experiment II: Efficiency for $R(s,t)$ on Real-World Graphs

- For smaller graphs, we can afford a large enough $L_{\max}$ to accurately approximate ER.



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

- Follows the same setup as Experiment II, but on a synthetic Erdős-Rényi random graph with parameters $(n, p) = (5000, 0.005)$.



Q & A

Thanks!

Contact: cuiguanyu@ruc.edu.cn