Edge Conductance Estimation using MCMC

Ashish Bora[1] Vivek S. Borkar[2]
Dinesh Garg[3] Rajesh Sundaresan[4]

[1]Department of Computer Science, University of Texas at Austin

[2]Department of Electrical Engineering, IIT Bombay, Mumbai, India

[3]IBM India Research Lab, Bengaluru, India

[4]Department of Electrical Communication Engineering and the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bengaluru, India.

Allerton, 2016

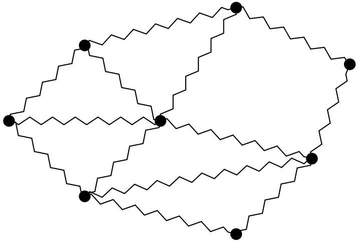
Outline

1. Motivation
   - What is conductance?
   - Why estimate conductances?
2. Notation
3. Prior Work
4. Algorithm
   - Motivation
   - Idea
   - Pseudocode
5. Theoretical results
6. Simulation Experiments
   - Cardinal Estimation
   - Ordinal Estimation
7. Discussion

Outline

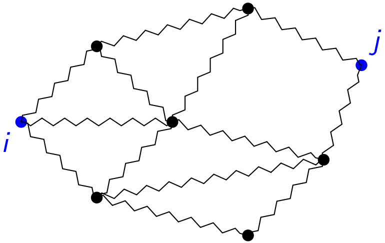
What is conductance?
Analogy

- Given a graph $G = (V, E)$, imagine each edge as a unit resistor.



What is conductance?

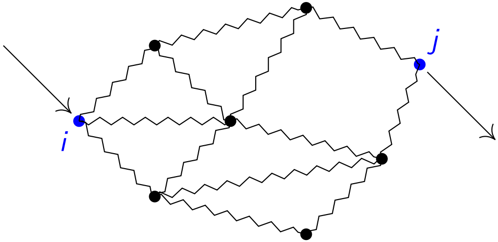
Definition



- Pick any two nodes $i, j \in V$.

What is conductance?

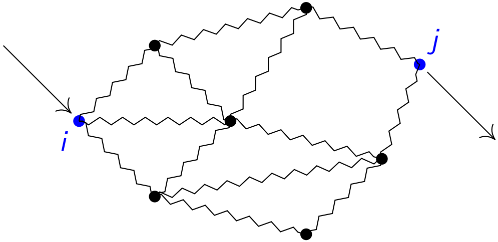
Definition



- Pick any two nodes $i, j \in V$.
- Inject unit current at i and extract it at j.

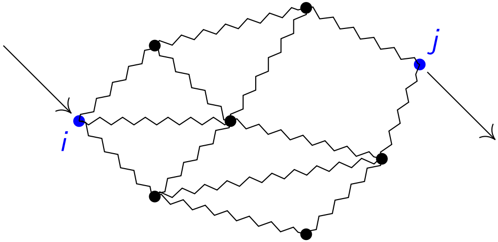
What is conductance?

Definition



- Pick any two nodes $i, j \in V$.
- Inject unit current at i and extract it at j.
- Effective resistance between i and j is the potential difference between them.

What is conductance?
Definition



- Pick any two nodes $i, j \in V$.
- Inject unit current at i and extract it at j.
- Effective resistance between i and j is the potential difference between them.
- Effective conductance is inverse of effective resistance.

Outline

Why estimate conductances?

- Effective resistance as a robust measure of distance ([1], [2])
  - Considers all paths
  - Less sensitive to edge or node insertions and deletions

Why estimate conductances?

- Effective resistance as a robust measure of distance ([1], [2])
  - Considers all paths
  - Less sensitive to edge or node insertions and deletions
- Sum of effective resistances across all pairs
  - measure of network robustness
  - equals network criticality parameter [4]

Why estimate conductances?

- Effective resistance as a robust measure of distance ([1], [2])
  - Considers all paths
  - Less sensitive to edge or node insertions and deletions
- Sum of effective resistances across all pairs
  - measure of network robustness
  - equals network criticality parameter [4]
- Edge resistances for graph sparsification [3]
  - Edges sampled (with replacement) according to their effective resistance
  - Approximately preseves quadratic form of Graph Laplacian (i.e. $x^\top Lx$)

Notation

- $G = (V, E)$ is an undirected, unweighted, connected, finite graph.
- $m = |E|$, $n = |V|$
- $\partial_i = \{j \mid (i, j) \in E\}$
- $d_i = |\partial_i|$
- $d_{max} = \max_{i \in V} d_i$, $d_{min} = \min_{i \in V} d_i$
- $d_{ij} = \min\{di, dj\}$, $D_{ij} = \max\{d_i, d_j\}$.
- $\pi_i = d_i/2m$, stationary distribution of simple random walk on G
- $d_{avg} = \sum_{i \in V} d_i \pi_i$
- G_{ij} = the effective conductance between i and j
- R_{ij} = the effective resistance between i and j

Prior Work

- Nodes of the graph can be embedded in Euclidean space so that the resulting pair-wise distances encode the effective resistances.

Prior Work

- Nodes of the graph can be embedded in Euclidean space so that the resulting pair-wise distances encode the effective resistances.
- The embedding depends on the edge-node adjacency matrix and the Laplacian of the graph.

Prior Work

- Nodes of the graph can be embedded in Euclidean space so that the resulting pair-wise distances encode the effective resistances.
- The embedding depends on the edge-node adjacency matrix and the Laplacian of the graph.

- [3] uses low dimensional random projection to preserve pairwise distances to estimate resistances. Takes only $\widetilde{O}(m/\epsilon^2)$ steps, but requires centralized computation.

Outline

Algorithm
Motivation

For large dynamic graphs, we would like algorithms that

- Are distributed and use minimal local communication

Algorithm

Motivation

For large dynamic graphs, we would like algorithms that

- Are distributed and use minimal local communication
- Have low memory footprint

Algorithm

Motivation

For large dynamic graphs, we would like algorithms that

- Are distributed and use minimal local communication
- Have low memory footprint
- Use very few computations per step

For large dynamic graphs, we would like algorithms that

- Are distributed and use minimal local communication
- Have low memory footprint
- Use very few computations per step
- Are easily parallelizable

For large dynamic graphs, we would like algorithms that

- Are distributed and use minimal local communication
- Have low memory footprint
- Use very few computations per step
- Are easily parallelizable
- Are incremental and adaptive

Outline

Algorithm

Idea

- We focus on edge effective conductance estimation as needed for graph sparsification algorithm in [3].

Algorithm

Idea

- We focus on edge effective conductance estimation as needed for graph sparsification algorithm in [3].
- We use random walks on the graph to estimate effective conductance and effective resistances.

Algorithm

Idea

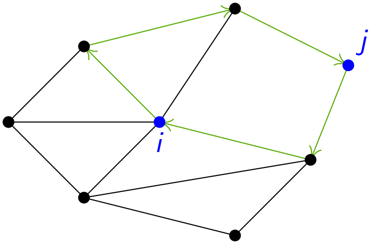
- We focus on edge effective conductance estimation as needed for graph sparsification algorithm in [3].
- We use random walks on the graph to estimate effective conductance and effective resistances.
- A random walk on the graph picks, from the current position, one of the neighbors with equal probability. Such random walks naturally give us many of the desired properties.

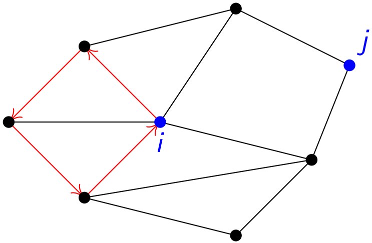
Algorithm

Idea

- We focus on edge effective conductance estimation as needed for graph sparsification algorithm in [3].
- We use random walks on the graph to estimate effective conductance and effective resistances.
- A random walk on the graph picks, from the current position, one of the neighbors with equal probability. Such random walks naturally give us many of the desired properties.
- We assume positive recurrence of the associated Markov Chain.

- Let p_{ij} denote the probability that a random walk starting at node i visits node j before returning to node i.





Algorithm

Idea

- A key fact that underlies our algorithm is:

$$p_{ij} = G_{ij}/d_i$$

Algorithm

Idea

- A key fact that underlies our algorithm is:

$$p_{ij} = G_{ij}/d_i$$

- We estimate this probability by averaging results from several i to i paths in a random walk.

- A key fact that underlies our algorithm is:

$$p_{ij} = G_{ij}/d_i$$

- We estimate this probability by averaging results from several i to i paths in a random walk.
- We will show how this can be done only with local communication for edge conductance estimation.

Algorithm

Variables

Let's introduce some variables that will be used in the algorithm

Let's introduce some variables that will be used in the algorithm

- $\widetilde{p}_{ij}$ is boolean. It denotes the the success or failure of visiting node j in an instance of a return path from node i to node i of the random walk.

Algorithm
Variables

Let's introduce some variables that will be used in the algorithm

- $\widetilde{p}_{ij}$ is boolean. It denotes the the success or failure of visiting node j in an instance of a return path from node i to node i of the random walk.
- N_i is the number of times node i was visited.

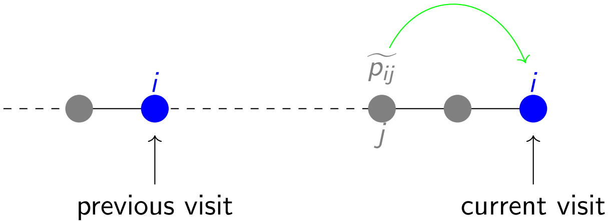
Algorithm

Variables

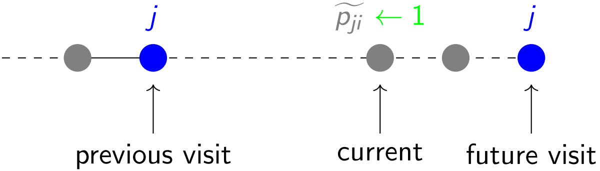
Let's introduce some variables that will be used in the algorithm

- $\widetilde{p}_{ij}$ is boolean. It denotes the the success or failure of visiting node j in an instance of a return path from node i to node i of the random walk.
- N_i is the number of times node i was visited.
- $\widehat{p}_{ij}$ is a running estimate of p_{ij}.

- In the long run, every visit to node i marks the end of a return path. Thus, on visiting node i, we can update $\widehat{p}_{ij}$ using $\widetilde{p}_{ij}$.



- A visit to node i at time t **will be** a part of a cycle that originated at node j prior to time t and a subsequent return to node j after time t (with probability 1, because of positive recurrence). Thus a visit to node i can be used to update $\widetilde{p_{ji}}$.



Outline

Algorithm

Pseudocode

Algorithm 1 Visit Before Return

Require: T, $G = (V, E)$

$\forall\ i \in V, N_i \leftarrow 0$

$\forall\ (i, j) \in E, \widehat{p}_{ij} = \widetilde{p}_{ij} = 0.$

Sample initial node X_1 from the stationary distribution π.

for $t = [1, 2, 3, \cdots, T]$ **do**

  Let $i = X_t$

  for all j in ∂_i **do**

    $\widehat{p}_{ij} \leftarrow (\widehat{p}_{ij} N_i + \widetilde{p}_{ij})/(N_i + 1)$

    $\widetilde{p}_{ij} \leftarrow 0$

    $\widetilde{p}_{ji} \leftarrow 1$

  $N_i \leftarrow N_i + 1$

  Jump to a neighbor of the current node as identified by the walk.

For every $(i, j) \in E$, output $\widehat{G}_{ij} = \max\left(1, \frac{d_i}{2}\widehat{p}_{ij} + \frac{d_j}{2}\widehat{p}_{ji}\right)$.

Complexity result

Theorem (Performance of **VisitBeforeReturn**)

Fix an edge $(i, j) \in E$. *For any* $0 < \epsilon < d_{ij}^2/(4m)$ *and* $0 < \delta < 1/2$,

$$T = \widetilde{O}\left(D_{ij} \cdot \max\{m, D_{ij} t_{mix}\} \cdot \frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$$

steps suffice to ensure that the output $\widehat{G}_{ij}$ *of the algorithm* **VisitBeforeReturn** *satisfies*

$$\mathbb{P}(|\widehat{G}_{ij} - G_{ij}| \geq \epsilon) \leq \delta.$$

If the algorithm is run for T *steps, it requires* $O(d_{avg} T)$ *computation steps on the average (worst case* $O(d_{max} T)$ *computations), and uses* $O(m \log T)$ *space.*

Proof ideas

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N_i} \widetilde{p}_{ij}^{k}}{N_i}$$

.

Proof ideas

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N_i} \widetilde{p}_{ij}^{k}}{N_i}$$

.

- Get concentration for N_i using McDiarmids inequality for Markov chains.

Proof ideas

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N_i} \widetilde{p}_{ij}^k}{N_i}$$

.

- Get concentration for N_i using McDiarmids inequality for Markov chains.
- Get concentration for $\widehat{p}_{ij}$ for a fixed N_i using Hoeffding's inequality.

Proof ideas

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N_i} \widetilde{p}_{ij}^k}{N_i}$$

.

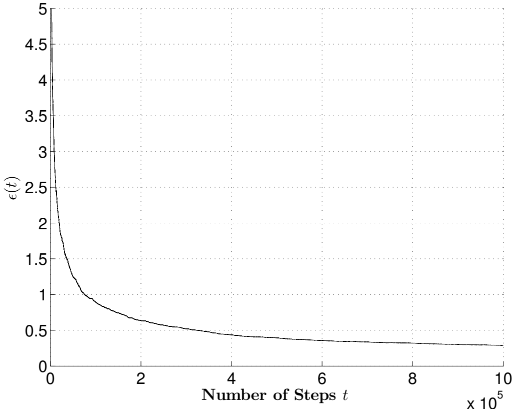
- Get concentration for N_i using McDiarmids inequality for Markov chains.
- Get concentration for $\widehat{p}_{ij}$ for a fixed N_i using Hoeffding's inequality.
- Combine the two.

Outline

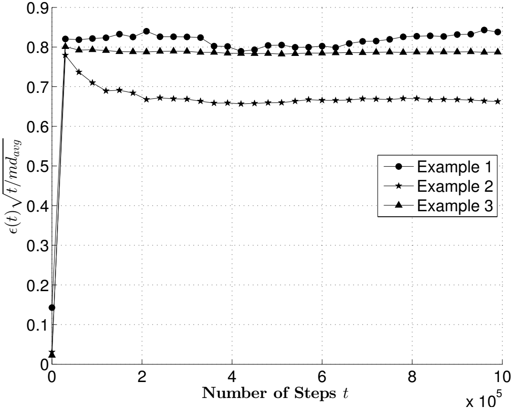
Simulation Experiments

Cardinal Estimation

$$\epsilon(t) = \frac{1}{m} \sum_{(i,j)\in E} \left| \widehat{G}_{ij}(t) - G_{ij} \right|$$



Simulation Experiments
Cardinal Estimation

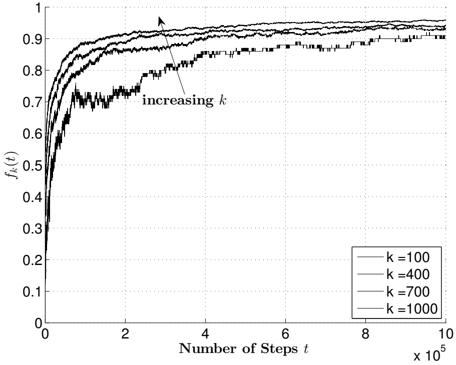


Outline

Simulation Experiments

Top-k set estimation

- Incremental approximate estimation algorithms can typically recover ordering much faster than exact values.
- We test performance of our algorithm for recovering top-k edges with high conductance by plotting fraction of top-k largest conductance edges correctly identified at time t.



Discussion

- Our algorithm can be naturally distributed: each node stores information about its neighbors.

Discussion

- Our algorithm can be naturally distributed: each node stores information about its neighbors.
- Parallelization can be easily obtained by running multiple random walks and avergaing their results.

Discussion

- Our algorithm can be naturally distributed: each node stores information about its neighbors.
- Parallelization can be easily obtained by running multiple random walks and avergaing their results.
- Our guarantees are weaker as compared to [3]. Particularly, the restriction on ϵ forces it to be too small. Whether this can be removed is an open question.

Discussion

- Our algorithm can be naturally distributed: each node stores information about its neighbors.
- Parallelization can be easily obtained by running multiple random walks and avergaing their results.
- Our guarantees are weaker as compared to [3]. Particularly, the restriction on ϵ forces it to be too small. Whether this can be removed is an open question.

Conclusion

- We have presented a MCMC based scheme to approximate edge conductances.
- Our algorithm is incremental and iterative, can be easily distributed, works with local communication, and uses very little memory and computation per step.
- We provide theoretical guarantees on the performance.
- Simulation experiments support our theoretical results.

References

- Aykut Firat, Sangit Chatterjee, and Mustafa Yilmaz. "Genetic clustering of social networks using random walks". In: *Computational Statistics & Data Analysis* 51.12 (2007), pp. 6285–6294.
- Francois Fouss et al. "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation". In: *Knowledge and Data Engineering, IEEE transactions on* 19.3 (2007), pp. 355–369.
- Daniel A Spielman and Nikhil Srivastava. "Graph sparsification by effective resistances". In: *SIAM Journal on Computing* 40.6 (2011), pp. 1913–1926.
- Ali Tizghadam and Alberto Leon-Garcia. "On robust traffic engineering in transport networks". In: *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE. 2008, pp. 1–6.

Thank you for your attention!
Questions?

Appendix A

- Estimating resistances from conductances

$$\begin{aligned}
\mathbb{P}\left(\frac{|\widehat{R}_{ij} - R_{ij}|}{R_{ij}} \geq \epsilon\right) &= \mathbb{P}\left(\left|\frac{\widehat{R}_{ij}}{R_{ij}} - 1\right| \geq \epsilon\right) \\
&= \mathbb{P}\left(\left|\frac{G_{ij}}{\widehat{G}_{ij}} - 1\right| \geq \epsilon\right) \\
&= \mathbb{P}(|G_{ij} - \widehat{G}_{ij}| \geq \epsilon \widehat{G}_{ij}) \\
&\leq \mathbb{P}(|\widehat{G}_{ij} - G_{ij}| \geq \epsilon),
\end{aligned}$$

where the last inequality follows because $\widehat{G}_{ij} \geq 1$.

Appendix B

- Our algorithm can be easily adapted for estimating the conductance value across any pair of nodes: maintain and update the variable $\widehat{p_{ij}}$ and $\widetilde{p_{ij}}$.
- If effective conductances between far-off nodes is desired, the communication is however no longer local.