The Power of Vertex Sparsifiers in Dynamic Graph Algorithms

Gramoz Goranci
University of Vienna, Austria

Monika Henzinger
University of Vienna, Austria

Pan Peng
University of Sheffield, UK

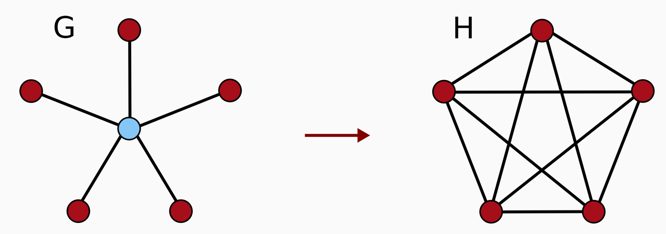
Vertex Sparsifier

- $G = (V, E)$ undirected, unweighted graph, $n = |V|$, $m = |E|$
- **Laplacian:** $L = D_G - A_G$, **Pseudo-inverse** $L^{\dagger}$
- The **effective resistance** between u and v in G is

$$R_G(u, v) := (1_u - 1_v)^{\intercal} L^{\dagger} (1_u - 1_v)$$

- Given G with k **terminals** $K \subset V(G)$, $H = (V_H, E_H, w_H)$ with $K \subset V_H$ is a α-**vertex resistance sparsifier** of G if for all t, t' from K:

$$\alpha \cdot R_G(t, t') \leq R_H(t, t') \leq R_G(t, t')$$



- (**Fast Vertex Sparsifier** [4]) There is a $\tilde{\mathcal{O}}(m\varepsilon^{-2})$ algorithm to compute a vertex resistance sparsifier H with

$$\alpha = (1 - \varepsilon) \text{ and } |E_H| = \tilde{\mathcal{O}}(k/\varepsilon^2)$$

Dynamic Effective Resistance

- Build data-structure that supports the following operations:
- **Insert**(u, v): insert the edge (u, v) in G
- **Delete**(u, v): delete the edge (u, v) from G
- **Query**(s, t): retun (approximate) $R_G(s, t)$

Motivation:

- natural, fundamental problem
- lots of recent work in the static setting e.g., fast laplacian solvers
- can it help us solve dynamic max-flow?

Our Result [1]

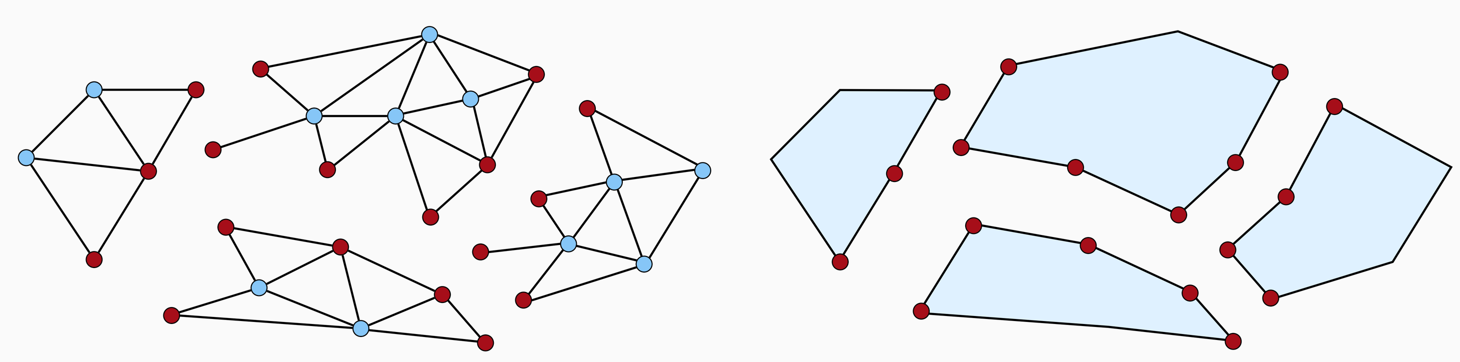
Theorem 1: When G is **planar**, there is a fully-dynamic algorithm that $(1 - \varepsilon)$-approximates effective resistances with

- $\tilde{\mathcal{O}}(n^{2/3}\varepsilon^{-2})$ update and query time

Idea: [Graph Clustering] + [Vertex Sparsification]

Data Structure

- partition G into n/r clusters G_i, each of size $\mathcal{O}(r)$
- each G_i has at most $\mathcal{O}(\sqrt{r})$ boundary nodes K_i
- compute a vertex sparsifiers H_i for each G_i w.r.t. K_i
- **cost:** $\tilde{\mathcal{O}}\left(n + (n/r) \cdot r\varepsilon^{-2}\right) = \tilde{\mathcal{O}}(n\varepsilon^{-2})$
- re-compute every $\mathcal{O}(n/r)$ operations



Handling Updates and Queries

Insert(u, v)

- declare u and v **boundary** in the clusters G_u and G_v
- re-compute from scratch sparsifiers H_u and H_v
- set (u, v) to be a **new** cluster
- **cost:** $\tilde{\mathcal{O}}(r\varepsilon^{-2})$

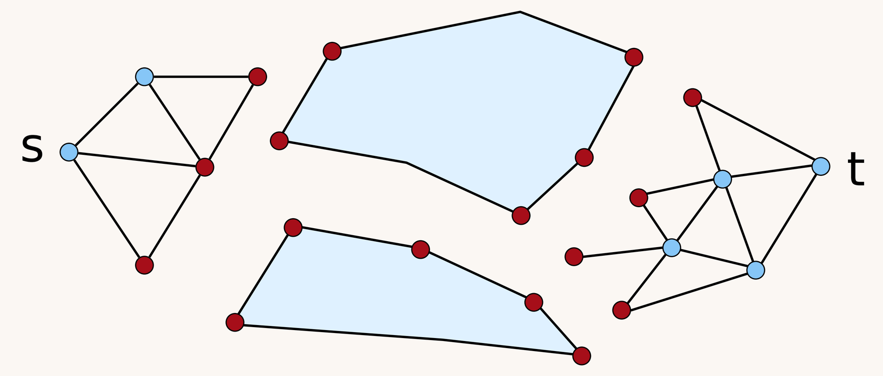
Delete(u, v)

- recompute from scratch a sparsifier $H_{(u,v)}$ of $G_{(u,v)}$
- **cost:** $\tilde{\mathcal{O}}(r\varepsilon^{-2})$

Query(s, t)

- let G_s and G_t the clusters containing s and t
- define **auxiliary** $H = G_s \cup G_t \cup \bigcup_{i \neq s,t} H_i$
- compute $R_H(s, t)$ using a Laplacian solver
- **cost:** $\tilde{\mathcal{O}}\left((r + n/\sqrt{r})\varepsilon^{-2}\right)$

Claim: $R_H(s, t)$ is a $(1 - \varepsilon)$–approx. to $R_G(s, t)$



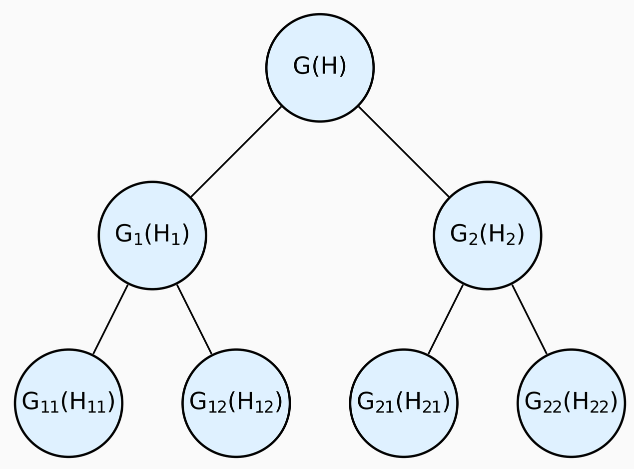
More Results [2]

Theorem 2: For **planar** graphs, can get the following improvement:

- $\tilde{\mathcal{O}}(n^{1/2}\varepsilon^{-2})$ update and query time.

Idea:

- Nested Dissection + Vertex Spars.



Theorem 3: Assuming **OMv** conjecture, in **general** graphs no dynamic algorithm can maintain **exact** $R_G(s, t)$ with

- $\mathcal{O}(n^{1-\delta})$ update and $\mathcal{O}(n^{2-\delta})$ query time

Recent Development [3]

For **general** graphs, there is a fully-dynamic algorithm that $(1 - \varepsilon)$-approximates effective resistances with

- $\tilde{\mathcal{O}}(m^{4/5}\varepsilon^{-4})$ update and query time.

References

[1] G. Goranci, M. Henzinger, and P. Peng. The Power of Vertex Sparsifiers in Dynamic Graph Algorithms. ESA 2017.
[2] G. Goranci, M. Henzinger, and P. Peng. Dynamic Effective Resistances and Approximate Schur Complement on Separable Graphs. ArXiv 2018.
[3] D. Durfee, Y. Gao, G. Goranci and R. Peng. Fully Dynamic Effective Resistances. ArXiv 2018.
[4] D. Durfee, R. Kyng, J. Peebles, A. B. Rao and S. Sachdeva. Sampling random spanning trees faster than matrix multiplication. STOC 2017.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.