The Expander Hierarchy and its Applications in Dynamic Graph Algorithms

Gramoz Goranci, Harald Räcke,
Thatchaphol Saranurak, Zihan Tan

Fakultät für Informatik
TU München

Tree Cut Sparsifier

- given (undirected) graph $G = (V, E)$
- compute tree $T = (V_T, E_T)$ with $V_T \supseteq V$ that approximates cuts in G

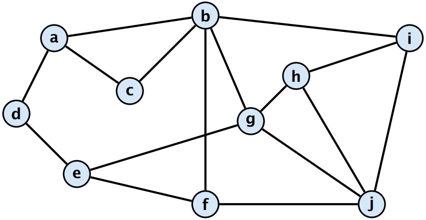
Formally, for all subsets $S \subseteq V$

$$\frac{1}{q}\,\mathrm{mincut}_T(S, V \setminus S) \le \mathrm{cut}_G(S, V \setminus S) \le \mathrm{mincut}_T(S, V \setminus S)$$

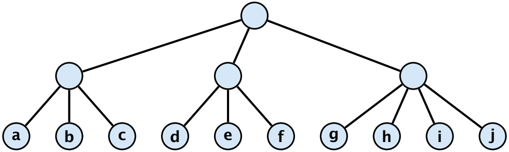
q is the quality of the T

Tree Cut Sparsifier

Graph G:



Tree T:



Tree Cut Sparsifier

Motivation:

Complicated cut-related problems can be (approximately) solved on G by only considering the problem on T.

- Minimum Bisection
- Simulteneous Source Location
- k-multicut
- Min-max graph partitioning
- Online Multicut

Previous Work

- **[R 02]**
  existence; quality $O(\log^3 n)$ (flow sparsifier)
- **[Bienkowski, Korzeniowski, R 03]**
  polynomial time; quality $O(\log^4 n)$ (flow sparsifier)
- **[Harrelson, Hildrum, Rao 03]**
  polynomial time; quality $O(\log^2 n \log\log n)$ (flow sparsifier)
- **[R, Shah 14]**
  polynomial time; quality $O(\log^{1.5} n \log\log n)$ (cut sparsifier)
  existence; quality $O(\log n \log\log n)$ (cut sparsifier)
- **[R, Shah, Täubig 14]**
  nearly linear time; quality $O(\log^4 n)$ (flow sparsifier)

Main Result

dynamic construction of a tree cut sparsifier for unweighted graphs

- update time: $n^{o(1)}$
- quality: $n^{o(1)}$

fully dynamic, deterministic, can be deamortized...

Main Result

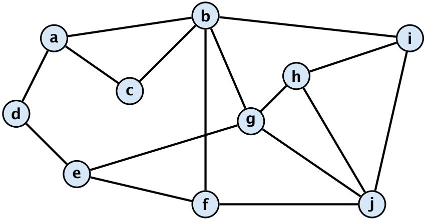
Consequences for Dynamic Graph Algorithms

- s-t maxflow/mincut
  approx: $n^{o(1)}$, update time: $n^{o(1)}$, query time: $O(\log n)$
- sparsest cut
  approx: $n^{o(1)}$, update time: $n^{o(1)}$, query time: $O(\log n)$
- multicommodity flow, multi-cut
  approx: $n^{o(1)}$, update time: $kn^{o(1)}$, query time: $O(k \log n)$
- treewidth-decomposition
  approx: $n^{o(1)}$, update time: $\mathrm{tw} \cdot n^{o(1)}$
- connectivity
  update time: $n^{o(1)}$, query time: $O(\log n)$

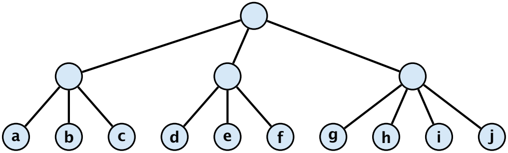
Proof Techniques of Existing Approaches

- leaf nodes of T correspond to vertices in G
- a level of the tree induces a partitioning of V into subsets

Graph G



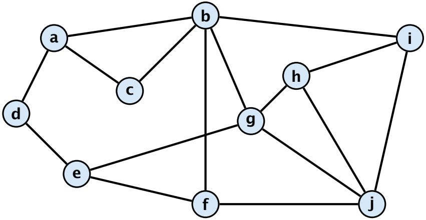
Tree T



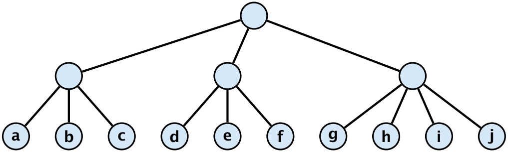
Proof Techniques of Existing Approaches

- an edge in the tree is assigned a capacity equal to the capacity of the corresponding cut in G

Graph G



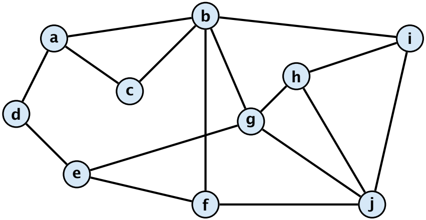
Tree T



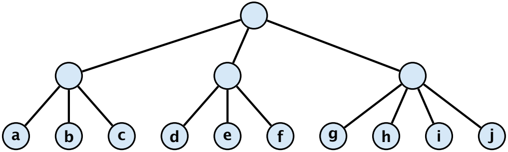
Proof Techniques of Existing Approaches

- equivalently a graph edge contributes to the capacity of every tree edge on the path between its endpoints in T

Graph G



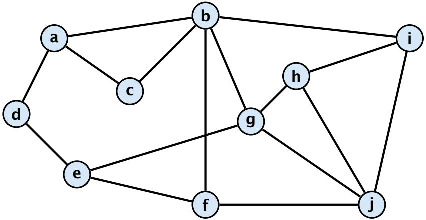
Tree T



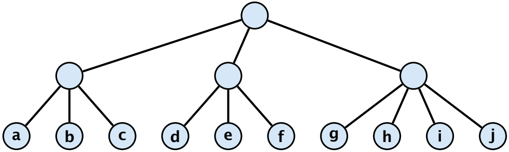
Proof Techniques of Existing Approaches

- this already guarantees that
  $\text{cut}_G(S, V \setminus S) \leq \text{mincut}_T(S, V \setminus S)$

Graph G



Tree T



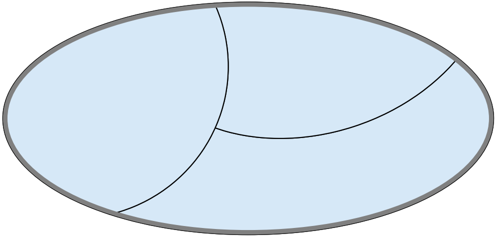
Proof Techniques of Existing Approaches

- let $\mathcal{P}_i$ be the partitioning on level i; level 0 is the leaf level
- let $G_{\mathcal{P}}$ be the graph obtained from G by contracting subsets in $\mathcal{P}$

Property I:
For a cluster S on some level $i+1$ the graph $G\{S\}_{\mathcal{P}_i}$ must expand well

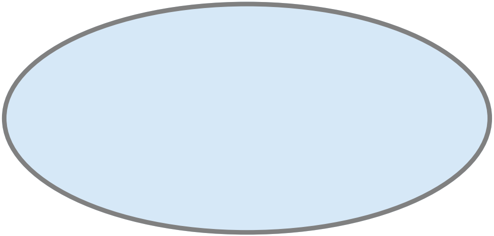
Property II:
The set S must have good boundary-expansion in G

Property I



- cluster S on level $i+1$ partitioned into sub-clusters
- the graph $G\{S\}_{\mathcal{P}_i}$ is obtained by
  - take induced subgraph $G[S]$ but turn edges leaving S into self-loops
  - then contract subsets of $\mathcal{P}_i$
- expands well means we can route an all-to-all flow problem **between edges** of $G\{S\}_{\mathcal{P}_i}$ with small congestion (C_{I})

Property II



- good boundary-expansion means we can route an all-to-all flow problem between boundary edges of S with small congestion (C_{II})

Proof

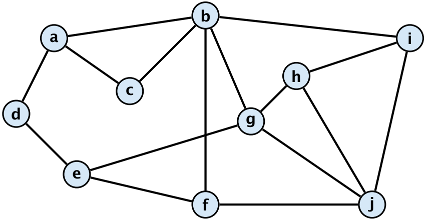
$$q \cdot \mathrm{cut}_G(S, V \setminus S) \geq \mathrm{mincut}_T(S, V \setminus S)$$

- take any multicommodity flow that can be routed in T with congestion at most 1
- route it in G with congestion at most q
- demand for the multicommodity flow is between edges of G
- an edge sends/receives at most one unit of flow in this demand

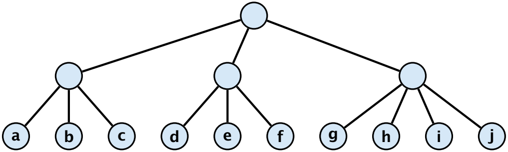
Proof Techniques of Existing Approaches

- what does demand between edges mean?

Graph G

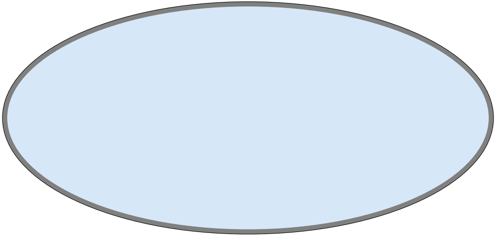


Tree T



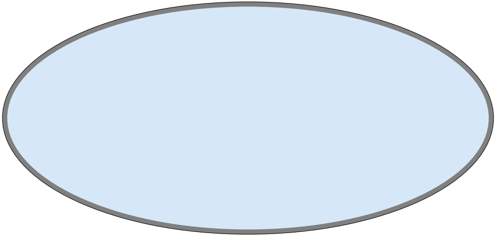
Route demand in G

- go top down level by level



Route demand in G

- go top down level by level

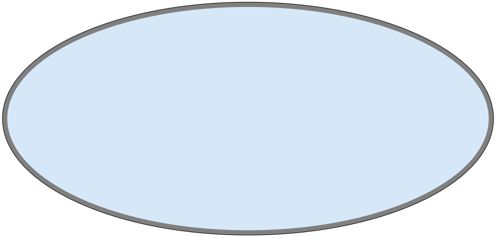


- route in the contracted graph (congestion $2C_{\mathsf{I}}$)
- undo the contraction and fill the gaps (congestion $2C_{\mathsf{I}}C_{\mathsf{II}}$)

Bottom Up Construction

For a bottom-up construction it is difficult to guarantee a good value for C_{II}.

There is a (trivial) guarantee of C_{I}^{h}.



Bottom Up Construction

Property I:

For a cluster S on some level $i+1$ the graph $G\{S\}_{\mathcal{P}_i}$ allows all-to-all routing between edges with congestion C_{I}

Property II′:

For a cluster S on some level $i+1$ the graph $G\{S\}_{\mathcal{P}_i}$ allows all-to-all routing between boundary-edges with congestion C'_{II}

Then we guarantee Property II with $(C'_{\mathsf{II}})^h$

Expander Decomposition

[Thatchaphol Saranurak, Di Wang 2019]

Given graph $G = (V, E)$, and parameter ϕ partition V into disjoint pieces $U_1, U_2, \ldots$ s.t.

- $G\{U_i\}$ can route all-to-all on its edges with congestion $1/\phi$
- $\sum_i |E(U_i, V \setminus U_i)| \leq \tilde{O}(\phi m)$

This only gives something good for Property I...

Expander Decomposition

Given graph $G = (V, E)$, and parameter ϕ partition V into disjoint pieces $U_1, U_2, \ldots$ s.t.

- $G[U_i]^{\alpha/\phi}$ can route all-to-all on its edges with congestion $\log m/\phi$
- $\sum_i |E(U_i, V \setminus U_i)| \leq \tilde{O}(\phi m)$

where $\alpha = \Omega(1/\operatorname{polylog} n)$.

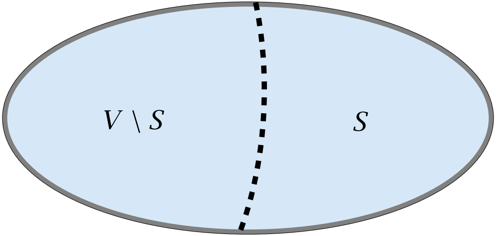
$G[U_i]^{\alpha/\phi}$ is $G[U_i]$ where every outgoing edge is transformed into α/ϕ many self-loops.

This means we get $C'_{\|} = \log m/\alpha$!!!

Existence

- give every edge money $\phi \log |S|$ for each cluster S of one of its end-points
- total amount of money handed out is $2\phi m \log n$
- distribute money to cut-edges so that in the end every cut-edges has money at least one –> small number of cut-edges

Existence



- congestion $\geq 2\log n/\phi \Rightarrow$ cut $\leq \log n \cdot \frac{1}{\text{congestion}} \cdot \mathrm{vol}(S)$
- every edge incident to S reduces its money by at least ϕ
- money available: $\geq \phi \cdot \mathrm{vol}(S)$
- every edge in the cut needs (at most)

$$1 + \alpha/\phi \cdot 2\phi \log n \leq 2$$

Choosing Parameters

In each iteration the number of edges reduces by a factor of ϕ.

Height $h \le \log_{1/\phi} m$.

$C_{\mathrm{I}} = \frac{1}{\phi} \log n$

$C_{\mathrm{II}} = (\log m/\alpha)^h$

Quality: $h \cdot C_{\mathrm{I}} \cdot C_{\mathrm{II}}$

Choose $\phi = 1/e^{\sqrt{\log n}}$

Making Things Dynamics...

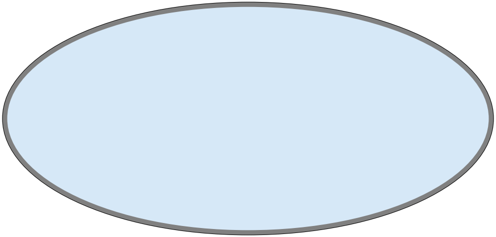
Expander Pruning

- given G and subset U with $G[U]^{\alpha/\phi}$ a ϕ-expander
- ($\leq \phi\,\mathrm{vol}(U)$) edge-updates for which one endpoint is in U

We can maintain a pruned set P such that

- $P_0 = \emptyset$; $P_i \subseteq P_{i+1}$
- $\mathrm{vol}(P_i) \leq 32i/\phi$ and $|E(P_i, U \setminus P_i)| \leq 16i$
- $|E(P_i, V \setminus U)| \leq 16i/\alpha$
- $G[U]^{\alpha/\phi}$ is a $\phi/38$-expander

Pruning



Maintaining the Expander Decomposition

Open Problems

- Better guarantee on the quality?
- Guarantees for vertex sparsifiers, i.e., sparsifiers w.r.t. a subset of vertices?