Preconditioning and Locality in Algorithm Design

Jason Li
PhD Thesis
Problems Studied

Graph cut problems

- Mincut: edge/vertex, undirected/directed, global/terminal/all-pairs
Problems Studied

Graph cut problems

- Mincut: edge/vertex, undirected/directed, global/terminal/all-pairs

- Conductance and expander decomposition
Problems Studied

Graph cut problems
- Mincut: edge/vertex, undirected/directed, global/terminal/all-pairs

- Conductance and expander decomposition

Graph distance problems
- Approximate shortest path, transshipment, $L_1$ embedding (PRAM model)
Preconditioning and Locality

Preconditioning: worst case vs. average case

- Assume that the input is random
  - expander (graph cut problems), low aspect ratio (distance)
Preconditioning and Locality

Preconditioning: worst case vs. average case

- Assume that the input is random
  - expander (graph cut problems), low aspect ratio (distance)
- Reduce to random instances
  - expander decomposition

![Diagram showing expander decomposition and remaining graph with recursion.]
Preconditioning and Locality

Preconditioning: worst case vs. average case

- Assume that the input is random
  - expander (graph cut problems), low aspect ratio (distance)
- Reduce to random instances
  - expander decomposition

- Popularized by Spielman and Teng [ST’04] on Laplacian system solvers
Preconditioning and Locality

- Local algorithms: explore a small neighborhood around v.
Preconditioning and Locality

- Local algorithms: explore a small neighborhood around \( v \)
  - Run in time \( \sim \text{smaller side of cut} \)
  - e.g. PageRank Nibble for computing approximate conductance
- This talk: locality as a principle in algorithm design
Preconditioning and Locality

- Local algorithms: explore a small neighborhood around $v$
  - Run in time ~ smaller side of cut
  - e.g. PageRank Nibble for computing approximate conductance
- This talk: locality as a principle in algorithm design

Locality: unbalanced vs. balanced

- Assume that the target solution is local to some vertex
  - e.g. mincut cuts a small neighborhood around $v$
Preconditioning and Locality

- Local algorithms: explore a small neighborhood around v
  - Run in time ~ smaller side of cut
  - e.g. PageRank Nibble for computing approximate conductance
- This talk: locality as a principle in algorithm design

Locality: unbalanced vs. balanced

- Assume that the target solution is local to some vertex
  - e.g. mincut cuts a small neighborhood around v
- Reduce to unbalanced instances
  - Straight reduction, or handle balanced case separately
The Case For Preconditioning and Locality

Powerful
- Resolves fundamental open problems
The Case For Preconditioning and Locality

Powerful
- Resolves fundamental open problems

Versatile
- Applicable to all types of graph problems
The Case For **Preconditioning and Locality**

**Powerful**
- Resolves fundamental open problems

**Versatile**
- Applicable to all types of graph problems

**Cutting-edge**
- Mostly unexplored in the past => future potential
- Some results are remarkably simple
  - All tools were around 40+ years ago, was only missing perspective
Problems Studied in Talk

Locality:
- Minimum Isolating Cuts problem
  ⇒ simple, fastest Steiner mincut algorithm
  ⇒ simple, fastest single-source mincut algorithm
Problems Studied in Talk

Locality:
- Minimum Isolating Cuts problem
  ⇒ simple, fastest Steiner mincut algorithm
  ⇒ simple, fastest single-source mincut algorithm
- Directed mincut: simple, fastest algorithm
Problems Studied in Talk

Locality:
- Minimum Isolating Cuts problem
  ⇒ simple, fastest Steiner mincut algorithm
  ⇒ simple, fastest single-source mincut algorithm
- Directed mincut: simple, fastest algorithm

Preconditioning:
- Deterministic mincut: first almost-linear time algorithm
  - simple on expanders
Part I: Locality

1. Steiner mincut
2. Directed mincut
Steiner Mincut

Given a graph and a set $R$ of terminals, find the mincut that separates at least two terminals.
Steiner Mincut

Given a graph and a set $R$ of terminals, find the mincut that separates at least two terminals
- Generalizes s-t mincut: $R = \{s, t\}$
- Generalizes global mincut: $R = V$
Steiner Mincut

Given a graph and a set $R$ of terminals, find the mincut that separates at least two terminals
- Generalizes s-t mincut: $R = \{s,t\}$
- Generalizes global mincut: $R = V$
- Useful subroutine for GH tree,
  $\tilde{O}(m+nc^2)$ algorithm [Bhalgat-Cole-Hariharan-Panigrahi ‘07]
Steiner Mincut

Given a graph and a set $R$ of terminals, find the mincut that separates at least two terminals
- Generalizes s-t mincut: $R = \{s,t\}$
- Generalizes global mincut: $R = V$
- Useful subroutine for GH tree,
  $\tilde{O}(m+n\sigma^2)$ algorithm [Bhalgat-Cole-Hariharan-Panigrahi ‘07]

Leap of faith: assume that Steiner mincut is unbalanced
- 1 terminal on one side?
Steiner Mincut

Given a graph and a set $R$ of terminals, find the mincut that separates at least two terminals

- Generalizes s-t mincut: $R = \{s, t\}$
- Generalizes global mincut: $R = V$
- Useful subroutine for GH tree, $\tilde{O}(m+nc^2)$ algorithm [Bhalgat-Cole-Hariharan-Panigrahi ‘07]

Leap of faith: assume that Steiner mincut is unbalanced

- 1 terminal on one side?

Can be reduced to this case! (random sampling)
Steiner Mincut

Theorem: unbalanced Steiner mincut can be solved in polylog(n) max-flow calls
Steiner Mincut

Theorem: unbalanced Steiner mincut can be solved in polylog(n) max-flow calls
- Minimum Isolating Cuts: new problem capturing the locality assumption
- Simple algorithm in $O(\log n)$ max-flows
Steiner Mincut

Theorem: unbalanced Steiner mincut can be solved in polylog(n) max-flow calls

- **Minimum Isolating Cuts**: new problem capturing the locality assumption
- **Simple** algorithm in $O(\log n)$ max-flows

Theorem: (general) Steiner mincut can be solved in polylog(n) max-flow calls

- **Simple** random sampling: reduce to unbalanced!
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal.
Minimum Isolating Cuts

Given a graph and a set \( R \) of terminals, find, for each terminal \( v \), the mincut \( S_v \) that isolates that terminal.

Trivial: \( |R| \) s-t mincuts

[L.-Panigrahi ‘20] \( O(\log |R|) \) s-t mincuts suffice!
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal.

Trivial: $|R|$ s-t mincuts

[L.-Panigrahi ’20] $O(\log |R|)$ s-t mincuts suffice!

$\Rightarrow$ unbalanced Steiner mincut in $O(\log |R|)$ max-flows
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal

Trivial: $|R| \cdot s-t$ mincuts

[L.-Panigrahi ’20] $O(\log |R|)$ $s-t$ mincuts suffice!

$\Rightarrow$ unbalanced Steiner mincut in $O(\log |R|)$ max-flows

Reduce general Steiner mincut to unbalanced:
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal.

Trivial: $|R|$ s-t mincuts

[Li-Panigrahi ’20] $O(\log |R|)$ s-t mincuts suffice!

$\Rightarrow$ unbalanced Steiner mincut in $O(\log |R|)$ max-flows

Reduce general Steiner mincut to unbalanced:
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal.

Trivial: $IRI$ $s$-$t$ mincuts

$[L.-Panigrahi '20] O(\log IRI)$ $s$-$t$ mincuts suffice!

$\Rightarrow$ unbalanced Steiner mincut in $O(\log IRI)$ max-flows

Reduce general Steiner mincut to unbalanced:

If sample at rate $\sim \frac{1}{|S\cup R|}$, then constant prob. success
Minimum Isolating Cuts

Given a graph and a set $R$ of terminals, find, for each terminal $v$, the mincut $S_v$ that isolates that terminal.

Trivial: IRI s-t mincuts

[L.-Panigrahi '20] $O(\log IRI)$ s-t mincuts suffice!

⇒ **unbalanced** Steiner mincut in $O(\log IRI)$ max-flows

Reduce general Steiner mincut to unbalanced:

Sample at rate $1/2$, $1/4$, $1/8$, ...

If sample at rate $\sim \frac{1}{|\text{ISNR}|}$, then constant prob. success.
Minimum Isolating Cuts

Idea: compute an “upper bound” for each isolating cut

- $C_v \geq S_v \quad \forall v \in R$
- $C_v$ are disjoint
Minimum Isolating Cuts

Idea: compute an "upper bound" for each isolating cut

- $C_v \supseteq S_v \quad \forall v \in R$
- $C_v$ are disjoint

For each $v \in R$, run max-flow on graph with $V \setminus S_v$ contracted
Minimum Isolating Cuts

Idea: compute an "upper bound" for each isolating cut

- $C_v \geq S_v \ \forall v \in R$
- $C_v$ are disjoint

For each $v \in R$, run max-flow on graph with $V \setminus S_v$ contracted
Each edge in at most 2 such graphs $\Rightarrow$ total size $\leq 2m$
$\Rightarrow$ max-flow time on $O(n)$ vertices, $O(m)$ edges
Minimum Isolating Cuts

- Compute $\log |R|$ bipartitions of $R$, $(X_k, Y_k)$
- Want: each pair $s, t$ in $R$ is separated in at least one of them
Minimum Isolating Cuts

- Compute log |R| bipartitions of R, (X_k, Y_k)
- Want: each pair s, t in R is separated in at least one of them
Minimum Isolating Cuts

- Compute \( \log |R| \) bipartitions of \( R \), \( (X_k, Y_k) \)
- Want: each pair \( s, t \) in \( R \) is separated in at least one of them
Minimum Isolating Cuts

- Compute log $|R|$ bipartitions of $R$, $(X_k, Y_k)$
- Want: each pair $s, t$ in $R$ is separated in at least one of them
- For each $k$, compute $(X_k, Y_k)$-min-cut
Minimum Isolating Cuts

- Compute \( \log |R| \) bipartitions of \( R \), \( (X_k, Y_k) \)
- Want: each pair \( s, t \) in \( R \) is separated in at least one of them
- For each \( k \), compute \( (X_k, Y_k) \)-min-cut
Minimum Isolating Cuts

- Compute $\log |R|$ bipartitions of $R$, $(X_k, Y_k)$
- Want: each pair $s,t$ in $R$ is separated in at least one of them
- For each $k$, compute $(X_k, Y_k)$-min-cut

Claim: Union of min-cuts separates all of $R$
Minimum Isolating Cuts

- Compute $\log |R|$ bipartitions of $R$, $(X_k, Y_k)$
  - Want: each pair $s, t$ in $R$ is separated in at least one of them
- For each $k$, compute $(X_k, Y_k)$-min-cut

Claim: Union of min-cuts separates all of $R$

**Upper Bound Lemma:**
In $G\setminus(\text{union of mincuts})$, $v$’s connected component contains $(v, R\setminus v)$-mincut
Minimum Isolating Cuts

- Compute \( \log |R| \) bipartitions of \( R \), \((X_k, Y_k)\)
- Want: each pair \( s, t \) in \( R \) is separated in at least one of them
- For each \( k \), compute \((X_k, Y_k)\)-min-cut

Claim: Union of min-cuts separates all of \( R \)

**Upper Bound Lemma:**
In \( G \backslash (\text{union of mincuts}) \), \( v \)'s connected component contains \((v, R \backslash v)\)-mincut

Proof: submodularity/uncrossing

\[
1 + 2 \geq 1 + 3
\]
Minimum Isolating Cuts

- Compute $\log |R|$ bipartitions of $R$, $(X_k, Y_k)$
- Want: each pair $s,t$ in $R$ is separated in at least one of them
- For each $k$, compute $(X_k, Y_k)$-min-cut

Claim: Union of min-cuts separates all of $R$

Upper Bound Lemma:
In $G\setminus(\text{union of mincuts})$, $v$’s connected component contains $(v, R\setminus v)$-mincut

Proof: submodularity/uncrossing
Minimum Isolating Cuts

- Compute $\log |R|$ bipartitions of $R$, $(X_k, Y_k)$
- Want: each pair $s, t$ in $R$ is separated in at least one of them
- For each $k$, compute $(X_k, Y_k)$-min-cut

Claim: Union of min-cuts separates all of $R$

**Upper Bound Lemma:**
In $G \setminus (\text{union of mincuts})$, $v$’s connected component contains $(v, R \setminus v)$-mincut

Proof: submodularity/uncrossing
Recap: Steiner mincut

Thm: Steiner mincut in polylog(n) max-flows

Assumption inspired by locality: Steiner mincut is unbalanced (1 terminal on one side)
- Reduces to Minimum Isolating Cuts

Simple algorithm for Min. Iso. Cuts

Simple reduction from general Steiner mincut to unbalanced: random sampling
Recap: Steiner mincut

Thm: Steiner mincut in polylog(n) max-flows

Assumption inspired by locality: Steiner mincut is unbalanced (1 terminal on one side)
- Reduces to Minimum Isolating Cuts

Simple algorithm for Min. Iso. Cuts

Simple reduction from general Steiner mincut to unbalanced: random sampling
Minimum Isolating Cuts: Applications

[L.-Panigrahi ‘21]: **All-pairs mincut** and **Gomory-Hu tree**: (1+ε)-approximation in polylog(n) exact max-flows

[L.-Nanongkai-Panigrahi-Saranurak-Yingchareonthawornchai ‘21] **vertex connectivity** in polylog(n) max-flows

[Chekuri-Quanrud, Mukhopadhyay-Nanongkai ‘21] **Symmetric bisubmodular function minimization, hypergraph connectivity, element connectivity**
Directed Mincut

Directed mincut: partition \((S, V \setminus S)\) s.t. no directed edge from \(S\) to \(V \setminus S\)
Directed Mincut

Directed mincut: partition \((S, V\setminus S)\) s.t. no directed edge from \(S\) to \(V\setminus S\)

Much harder than undirected:
- Karger’s randomized contraction doesn’t work
- Sparsifiers for directed graphs are hard/impossible
Directed Mincut

Directed mincut: partition \((S, V \setminus S)\) s.t. no directed edge from \(S\) to \(V \setminus S\)

Much harder than undirected:
- Karger’s randomized contraction doesn’t work
- Sparsifiers for directed graphs are hard/impossible

Previous best: \(\tilde{O}(mn)\) [Hao-Orlin’94]

[Cen-L.-Nanongkai-Panigrahi-Saranurak] \(\sqrt{n}\) max-flows \(\Rightarrow O(m\sqrt{n} + n^2)\)
This talk: \((1+\varepsilon)\)-approximation
Directed Mincut

Why are directed sparsifiers hard?
Directed Mincut

Why are directed sparsifiers hard?

Thm [Karger]: For undirected graphs, if sample each edge w.p. \( p \sim \frac{\log n}{\varepsilon^2 \lambda} \), then w.h.p., each cut has \((1 \pm \varepsilon)p\) fraction edges sampled.

Proof: cut counting: use fact that \( \leq n^{2\alpha} \) many \( \alpha \)-approximate mincuts.
Directed MinCut

Why are directed sparsifiers hard?

Thm [Karger]: For *undirected* graphs, if sample each edge w.p. \( p \sim \frac{\log n}{\varepsilon^2 \lambda} \),
then w.h.p., each cut has \((1\pm \varepsilon)p\) fraction edges sampled.

Proof: cut counting: use fact that \( \leq n^{2\alpha} \) many \( \alpha \)-approximate mincuts.

Not true for directed graphs!
Directed Mincut

Why are directed sparsifiers hard?

Thm [Karger]: For undirected graphs, if sample each edge w.p. $p \sim \frac{\log n}{\varepsilon^2 \lambda}$, then w.h.p., each cut has $(1 \pm \varepsilon)p$ fraction edges sampled

Proof: cut counting: use fact that $\leq n^{2\alpha}$ many $\alpha$-approximate mincuts

Not true for directed graphs!

Locality assumption: mincut is $k$-unbalanced: $\leq k$ vertices on one side

Partial sparsification: preserve only $k$-unbalanced cuts ($\leq n^k$ of them)
Directed Mincut

Why are directed sparsifiers hard?
Thm [Karger]: For undirected graphs, if sample each edge w.p. \( p \sim \frac{\log n}{\varepsilon^2 \lambda} \),
then w.h.p., each cut has \((1 \pm \varepsilon)p\) fraction edges sampled
Proof: cut counting: use fact that \( \leq n^{2\alpha} \) many \( \alpha \)-approximate mincuts
Not true for directed graphs!

Locality assumption: mincut is \( k \)-unbalanced: \( \leq k \) vertices on one side
Partial sparsification: preserve only \( k \)-unbalanced cuts (\( \leq n^k \) of them)

Balanced case: sample \( s,t \) at random and compute \( s-t \) mincut
\( s \rightarrow \hat{S} \rightarrow t \) occurs w.p. \( \geq k/n \) \( \Rightarrow \) repeat \( \sim n/k \) times
Directed Mincut

Algorithm: compute partial sparsifier $H$, then find directed mincut $\partial_H S$ in sparsifier. Output $\partial_G S$

Assumption: directed mincut is $k$-unbalanced
Directed Mincut

Algorithm: compute partial sparsifier $H$, then find directed mincut $\partial_h S$ in sparsifier. Output $\partial_e S$

Assumption: directed mincut is $k$-unbalanced

Thm: suppose sampled graph $H$ satisfies (for some $p$)
- all $k$-unbalanced cuts have $(1 \pm \varepsilon)p$ fraction edges sampled
- all $k$-balanced cuts have size $>> p\lambda$ ($\lambda = \text{mincut}$)
then mincut in $H$ is $(1 + \varepsilon)$-mincut in $G$
Directed Mincut

Sample each edge with prob. $p \sim \frac{k \log n}{\varepsilon^2 \lambda}$
Directed Mincut

Sample each edge with prob. \( p \sim \frac{k}{\varepsilon \log n} \frac{1}{\varepsilon^2 \gamma} \)

- Each cut fails to be within \( 1 \pm \varepsilon \) w.p. \( \ll n^{-k/\varepsilon} \)
- \( \sim n^{k/\varepsilon} \frac{k}{\varepsilon} \)-unbalanced cuts: all preserved w.h.p.
Directed MinCut

Sample each edge with prob. $p \sim \frac{k \log n}{\varepsilon^2 \lambda}$

- Each cut fails to be within $1 \pm \varepsilon$ w.p. $<< n^{-k/\varepsilon}$
- $\sim n^{k/\varepsilon} \frac{k}{\varepsilon}$-unbalanced cuts: all preserved w.h.p.

Force all $\frac{k}{\varepsilon}$-balanced cuts to have size $>> p \lambda$

by overlaying an expander: $|\partial S| \approx \frac{2\varepsilon \lambda}{k} |S|$ for $|S| \leq n/2$
Directed Mincut

Sample each edge with prob. $p \sim \frac{k \log n}{\epsilon^2 \lambda}$

- Each cut fails to be within $1 \pm \epsilon$ w.p. $< n^{-k/\epsilon}$
- $\sim n^{k/\epsilon} \frac{k}{\epsilon}$-unbalanced cuts: all preserved w.h.p.

Force all $\frac{k}{\epsilon}$-balanced cuts to have size $>> p \lambda$

by overlaying an expander: $|\partial S| \approx \frac{2\epsilon \lambda}{k} |S|$ for $|S| \leq n/2$

- $\frac{k}{\epsilon}$-balanced cut increases by $\geq 2\lambda$
- $k$-unbalanced cut increases by $\leq 2\epsilon \lambda$

(including mincut)
Directed MinCut

Sample each edge with prob. $p \sim \frac{k\log n}{\varepsilon^2 \lambda}$
Directed MinCut

Sample each edge with prob. \( p \sim \frac{k \log n}{\varepsilon^2 \lambda} \)

Partial sparsifier with mincut \( p^\lambda \sim \frac{k \log n}{\varepsilon^3} \)

Run Gabow's algorithm on \( H \): \( \widetilde{O}(m \lambda_H) \) time = \( \widetilde{O}(\frac{km}{\varepsilon^3}) \)
Directed Mincut

Sample each edge with prob. $p \sim \frac{k \log n}{\varepsilon^2 \lambda}$

Partial sparsifier with mincut $p \lambda \sim \frac{k \log n}{\varepsilon^3}$

Run Gabow’s algorithm on $H$: $\tilde{O}(m \lambda_{\|})$ time $= \tilde{O}\left(\frac{km}{\varepsilon^3}\right)$

Overall running time: $\sim \frac{km}{\varepsilon}$ time unbalanced (approx), $\sim \frac{n}{k}$ max flows balanced (exact)
Directed Mincut

Sample each edge with prob. $p \sim \frac{k \log n}{\varepsilon^2 \lambda}$

Partial sparsifier with mincut $p \lambda \sim \frac{k \log n}{\varepsilon^3}$

Run Gabow’s algorithm on $H$: $\tilde{O}(m \lambda_H)$ time $= \tilde{O}(\frac{km}{\varepsilon^3})$

Overall running time: $\sim km$ time unbalanced (approx),
$\sim \frac{n}{k}$ max flows balanced (exact)

Arborescence packing + minimum 1-respecting cut:
$\sim k$ max flows unbalanced, exact
$k = \sqrt{n} : \sim \sqrt{n}$ max flows
Recap: directed mincut

Thm: directed mincut in $\sqrt{n}$ max-flows

Directed sparsification is hard
Locality: partial sparsification of only unbalanced cuts
Balanced case: different strategy this time

$\Rightarrow$ simple $(1+\varepsilon)$-approximate directed mincut
few extra steps for exact
Part II: Preconditioning

1. Deterministic mincut
Deterministic Mincut

Global mincut: given a graph, find minimum # edges whose removal disconnects the graph
Deterministic Mincut

Global mincut: given a graph, find minimum # edges whose removal disconnects the graph

- Karger ‘93, ‘96: $\tilde{O}(n^2)$ time, $\tilde{O}(m)$ time randomized
- Kawarabayashi-Thorup ‘15: $\tilde{O}(m)$ deterministic for simple graphs
- L.-Panigrahi ‘20: deterministic Steiner mincut in $\sim$max-flow time
- L.: deterministic mincut in $m^{1+o(1)}$ time
Deterministic Mincut

Global mincut: given a graph, find minimum # edges whose removal disconnects the graph
- Karger ‘93, ‘96: $\tilde{O}(n^2)$ time, $\tilde{O}(m)$ time randomized
- Kawarabayashi-Thorup ‘15: $\tilde{O}(m)$ deterministic for simple graphs
- L.-Panigrahi ‘20: deterministic Steiner mincut in $\sim$max-flow time
- L.: deterministic mincut in $m^{1+o(1)}$ time

Preconditioning assumption: assume input is an expander
- Expander case: simple algorithm following [Karger ‘96]
- General case: expander decomposition (technical)
Mincut by Sparsification + Tree Packing

Thm [Karger ‘96]: Suppose given a skeleton graph $H$ s.t.
- $H$ has $O(m)$ edges
- The mincut of $H$ is $\lambda_H \geq \rho \lambda$
- For the mincut $\partial_e S^*$ in $G$,
  the $|\partial_h S^*| \leq (1+\varepsilon)\rho \lambda$

Then, can compute exact mincut in $G$ in $m\lambda_H$ additional deterministic time
Mincut by Sparsification + Tree Packing

Thm [Karger ‘96]: Suppose given a skeleton graph $H$ s.t.
- $H$ has $O(m)$ edges  
- The mincut of $H$ is $\lambda_H \geq \rho \lambda$
- For the mincut $\partial_e S^*$ in $G$,
  the $|\partial_h S^*| \leq (1+\varepsilon)\rho \lambda$

Then, can compute exact mincut in $G$ in $m\lambda_H$ additional deterministic time
Mincut by Sparsification + Tree Packing

Thm [Karger ‘96]: Suppose given a skeleton graph $H$ s.t.
- $H$ has $O(m)$ edges
- The mincut of $H$ is $\lambda_H \geq \rho \lambda$
- For the mincut $\partial_e S^*$ in $G$,
  the $|\partial_H S^*| \leq (1+\varepsilon)\rho \lambda$

Then, can compute exact mincut in $G$ in $m\lambda_H$ additional
deterministic time

Karger: randomized skeleton via graph sparsification

(1+\varepsilon) approximate cut sparsifier

suffices: sample $(1\pm \varepsilon)p$ fraction of
all cuts
Mincut by Sparsification + Tree Packing

Thm [Karger ‘96]: Suppose given a skeleton graph $H$ s.t.
- $H$ has $O(m)$ edges
- The mincut of $H$ is $\lambda_H \geq \rho \lambda$
- For the mincut $\partial_c S^*$ in $G$, the $|\partial_H S^*| \leq (1+\varepsilon)\rho \lambda$

Then, can compute exact mincut in $G$ in $m\lambda_H$ additional deterministic time

This talk: deterministic skeleton for expander
Skeleton Graph: Sparsification

Sample each edge in G with prob \( p := \frac{100 \log n}{\varepsilon^2 \lambda} \). Let \( H = \) sampled edges.
Skeleton Graph: Sparsification

Sample each edge in $G$ with prob $p := \frac{100 \log n}{\varepsilon^2 \lambda}$. Let $H = \text{sampled edges}$

Thm [Karger] w.h.p., each cut $\partial S (S \subseteq V)$ has $(1 \pm \varepsilon)p$ fraction sampled
Skeleton Graph: Sparsification

Sample each edge in $G$ with prob $p := \frac{100 \log n}{\varepsilon^2 \lambda}$. Let $H = \text{sampled edges}$

Thm [Karger] w.h.p., each cut $\partial S (S \subseteq V)$ has $(1 \pm \varepsilon)p$ fraction sampled

Proof: “smart union bound over all cuts”
Skeleton Graph: Sparsification

Sample each edge in $G$ with prob \( p := \frac{100 \log n}{\varepsilon^2 \lambda} \). Let $H = \text{sampled edges}$

Thm [Karger] w.h.p., each cut $\partial S (S \subseteq V)$ has $(1 \pm \varepsilon)p$ fraction sampled

Proof: "smart union bound over all cuts"

Derandomization?
Skeleton Graph: Sparsification

Sample each edge in $G$ with prob $p := \frac{100 \log n}{\varepsilon^2 \lambda}$. Let $H = \text{sampled edges}$

Thm [Karger] w.h.p., each cut $\partial S (S \subseteq V)$ has $(1 \pm \varepsilon)p$ fraction sampled

Proof: “smart union bound over all cuts”

Derandomization?

Even verification is hard! $2^n$ cuts to check
Need to “union bound” more efficiently
Skeleton Graph: Sparsification

Sample each edge in $G$ with prob $p := \frac{100 \log n}{\varepsilon^2 \lambda}$. Let $H$ = sampled edges

Thm [Karger] w.h.p., each cut $\partial S (S \subseteq V)$ has $(1 \pm \varepsilon) p$ fraction sampled

Proof: “smart union bound over all cuts”

Derandomization?

Even verification is hard! $2^n$ cuts to check

Need to “union bound” more efficiently

**Locality assumption:** $(1+\varepsilon)$-preserve only unbalanced cuts

-mincut is unbalanced for expander

Balanced cuts: overlay expander (same as before)
Expanders

Conductance of a graph: \( \Phi(G) = \min_{S \subseteq V} \frac{|E(S, V \setminus S)|}{\text{vol}(S)} \leq \text{vol}(S) \leq \text{vol}(V \setminus S) \)

G is a \( \phi \)-expander if \( \Phi(G) \geq \phi \)

"Volume" of \( S \): sum of degrees in \( S \)
Expanders

Conductance of a graph: \( \Phi(G) = \min_{S \subseteq V} \frac{|E(S, V \setminus S)|}{\text{vol}(S)} \) \( \text{vol}(S) \leq \text{vol}(V \setminus S) \)

\( \text{``volume'' of } S: \text{ sum of degrees in } S \)

G is a \( \phi \)-expander if \( \Phi(G) \geq \phi \)

Why expanders? \([KT'15]\)

Claim: in a \( \phi \)-expander, any \( \alpha \)-approx mincut \( \exists S \) \( (|S| \leq \alpha \lambda) \) must have \( |S| \leq \alpha / \phi \)
Expanders

Conductance of a graph: \( \Phi(G) = \min_{S \subseteq V} \frac{|E(S, V \setminus S)|}{\text{vol}(S)} \)

G is a \( \phi \)-expander if \( \Phi(G) \geq \phi \)

Why expanders? [KTT'15]

Claim: in a \( \phi \)-expander, any \( \alpha \)-approx mincut \( \exists S \) (|\partial S| \leq \alpha \lambda)

must have \( |S| \leq \alpha/\phi \)

Proof: All degrees \( \geq \lambda \) [\( \lambda = \text{mincut} \)]

so \( \text{vol}(S) \geq \lambda |S| \)

\( \phi \)-expander: \( |\partial S| \geq \phi \text{ vol}(S) \geq \phi \lambda |S| \)
Derandomization: Unbalanced Cuts

First goal: ensure that sample $(1 \pm \varepsilon)p$ for all unbalanced cuts $\mathcal{S}$: $|S| \leq \frac{\alpha}{\phi}$

(includes all $\alpha$-approximate mincuts for a $\phi$-expander)
Derandomization: Unbalanced Cuts

First goal: ensure that sample \((1 \pm \varepsilon)p\) for all unbalanced cuts \(\mathcal{A}: |S| \leq \frac{\alpha}{\phi}\)

(includes all \(\alpha\)-approximate mincuts for a \(\phi\)-expander)

Lemma: suffices to ensure that: sample \(p\) fraction \(\pm \varepsilon (\frac{\phi}{\alpha})^2 \lambda\) of:

\[
\deg(u) \quad \#(u,v)
\]

only \(n+m\) constraints!
Derandomization: Unbalanced Cuts

First goal: ensure that sample \((1 \pm \varepsilon)p\) for all unbal. cuts \(\mathcal{S}: |S| \leq \frac{\alpha}{\phi}\)

(includes all \(\alpha\)-approximate mincuts for a \(\phi\)-expander)

Lemma: suffices to ensure that: sample p fraction \(\pm \varepsilon \left(\frac{\phi}{\alpha}\right)^2 \lambda\) of:

Proof: Let \(|S| \leq \frac{\alpha}{\phi}\)

\[ |\mathcal{S}| = \sum_{v \in S} \deg(v) - 2 \sum_{u,v \in S} \#(u,v) \]

only \(n+m\) constraints!
Derandomization: Unbalanced Cuts

First goal: ensure that sample \((1 \pm \varepsilon)p\) for all unbal. cuts \(\mathcal{S}: |S| \leq \frac{\alpha}{\phi}\)
(includes all \(\alpha\)-approximate mincuts for a \(\phi\)-expander)

Lemma: suffices to ensure that: sample \(p\) fraction \(\pm \varepsilon (\frac{\phi}{\alpha})^2 \lambda\) of:

Proof: Let \(|S| \leq \frac{\alpha}{\phi}\)

\[|\mathcal{S}| = \sum_{v \in S} \deg(v) - 2 \sum_{u, v \in S} \#(u, v)\]

\(|S|^2 \leq \left(\frac{\alpha}{\phi}\right)^2\) terms, each with

\(\leq \varepsilon (\frac{\phi}{\alpha})^2 \lambda\) additive error

\(\Rightarrow \varepsilon \lambda\) total additive error

\((1 + \varepsilon)\) multiplicative error
Derandomization: Unbalanced Cuts

First goal: ensure that sample \((1 \pm \varepsilon)p\) for all unbal. cuts \(\mathcal{S}: |S| \leq \frac{\alpha}{\phi}\)
(includes all \(\alpha\)-approximate mincuts for a \(\phi\)-expander)

Lemma: suffices to ensure that: sample \(p\) fraction \(\pm \varepsilon \left(\frac{\phi}{\alpha}\right)^2 \lambda\) of:

Proof: Let \(|S| \leq \frac{\alpha}{\phi}\)

\[|\mathcal{S}| = \sum_{v \in S} \text{deg}(v) - 2 \sum_{u, v \in S} \#(u, v)\]

\(|S|^2 \leq \left(\frac{\alpha}{\phi}\right)^2\) terms, each with
\[\leq \varepsilon \left(\frac{\phi}{\alpha}\right)^2 \lambda\] additive error
\[\Rightarrow \varepsilon \lambda\] total additive error
\[(1 + \varepsilon)\) multiplicative error

only \(n+m\) constraints!
Pessimistic estimators method: \(\tilde{O}(m)\) time
Recap: Deterministic Mincut

Thm: deterministic mincut in $m^{1+o(1)}$ time

Karger: reduces to computing mincut sparsifier

Deterministic sparsifier is hard: $2^n$ many cuts to preserve

Preconditioning assumption: input is expander

Locality assumption: mincut is unbalanced
Recap: Deterministic Mincut

Thm: deterministic mincut in $m^{1+o(1)}$ time

Karger: reduces to computing mincut sparsifier

Deterministic sparsifier is hard: $2^n$ many cuts to preserve

Preconditioning assumption: input is expander

Locality assumption: mincut is unbalanced

- Unbalanced cuts: only need to preserve $\deg(v)$ and $\#(u,v)$
- Balanced cuts: overlay expander

$\Rightarrow$ simple mincut sparsifier for expander

General graphs: expander decomposition
Summary

Preconditioning
- current fastest...
- det. exp. decomp.
- parallel SSSP
- transshipment

Locality
- current fastest...
- Steiner mincut
- vertex mincut
- approx. GH tree
- directed mincut

*expander*
Summary

Preconditioning
- current fastest...
- det. exp. decomp.
- parallel SSSP
- transshipment

Locality
- current fastest...
- Steiner mincut
- vertex mincut
- approx. GH tree
- directed mincut

Misc.
- current fastest...
- minimum k-cut
- constant-approx.
- planar sparsest cut
- Feedback Vertex Set (FPT)
Summary

Preconditioning
- current fastest...
- det. exp. decomp.
- parallel SSSP
- transshipment

Locality
- current fastest...
- Steiner mincut
- vertex mincut
- approx. GH tree
- directed mincut

Misc.
- current fastest...
- minimum k-cut
- constant-approx.
- planar sparsest cut
- Feedback Vertex Set (FPT)

Future work: Gomory-Hu tree in polylog(n) max-flows?

Know: GH tree for expanders in polylog(n) max-flows (Min. Iso. Cuts)

Don’t know general case ⇒ expander case reduction!