Expander Decomposition and Pruning: Faster, Stronger, and Simpler

Jaehyun Koo (koosaga)

March 31, 2021

Introduction

Expander Decomposition

Divide and Conquer: Divide the problem into smaller, tractable, mergeable pieces.

Example: Heavy light decomposition divides the tree into chains.

Expander Decomposition

Divide and Conquer: Divide the problem into smaller, tractable, mergeable pieces.

Example: Heavy light decomposition divides the tree into chains.

It is hard to decompose the graph into tractable pieces. Usually those decomposition requires special properties of graph (e.g. low treewidth).

Expander decomposition decomposes any graph into expanders.

Expander Decomposition

Divide and Conquer: Divide the problem into smaller, tractable, mergeable pieces.

Example: Heavy light decomposition divides the tree into chains.

It is hard to decompose the graph into tractable pieces. Usually those decomposition requires special properties of graph (e.g. low treewidth).

Expander decomposition decomposes any graph into expanders.

Expander has good properties and so are more **tractable**.

Expander are well-separated and thus **mergeable**.

Some Definition

We consider unweighted graphs.

For a graph G, the **cut** is a vertex subset $S \neq \emptyset, S \subsetneq V$

$G[S]$ is an induced subgraph of S in G.

For disjoint set $A, B \subseteq V(G)$, $E(A, B)$ is the set of edges where two endpoints lie in A and B.

Let $\delta(S)$ be the value of cut: $\delta(S) = |E(S, V \setminus S)$

The **volume** of cut S is defined as $vol_G(S) = \sum_{v \in S} deg(v)$.

Some Definition

We consider unweighted graphs.

For a graph G, the **cut** is a vertex subset $S \neq \emptyset, S \subsetneq V$

$G[S]$ is an induced subgraph of S in G.

For disjoint set $A, B \subseteq V(G)$, $E(A, B)$ is the set of edges where two endpoints lie in A and B.

Let $\delta(S)$ be the value of cut: $\delta(S) = |E(S, V \setminus S)$

The **volume** of cut S is defined as $vol_G(S) = \sum_{v \in S} deg(v)$.

The **conductance** of cut S is defined as $\Phi_G(S) = \frac{\delta(S)}{min(vol_G(S), vol_G(V \setminus S))}$. For convenience, assume that $vol_G(S) \leq vol_G(V \setminus S)$ otherwise noted.

Some Definition

We consider unweighted graphs.

For a graph G, the **cut** is a vertex subset $S \neq \emptyset, S \subsetneq V$

$G[S]$ is an induced subgraph of S in G.

For disjoint set $A, B \subseteq V(G)$, $E(A, B)$ is the set of edges where two endpoints lie in A and B.

Let $\delta(S)$ be the value of cut: $\delta(S) = |E(S, V \setminus S)$

The **volume** of cut S is defined as $vol_G(S) = \sum_{v \in S} deg(v)$.

The **conductance** of cut S is defined as $\Phi_G(S) = \frac{\delta(S)}{min(vol_G(S), vol_G(V \setminus S))}$. For convenience, assume that $vol_G(S) \leq vol_G(V \setminus S)$ otherwise noted.

The **conductance** of graph G, $\Phi_G = \min_{S \text{ is a cut of } G} \Phi_G(S)$. If a cut does not exist $(|V| = 1)$, then $\Phi_G = 1$.

Some Definition

We consider unweighted graphs.

For a graph G, the **cut** is a vertex subset $S \neq \emptyset, S \subsetneq V$

$G[S]$ is an induced subgraph of S in G.

For disjoint set $A, B \subseteq V(G)$, $E(A, B)$ is the set of edges where two endpoints lie in A and B.

Let $\delta(S)$ be the value of cut: $\delta(S) = |E(S, V \setminus S)$

The **volume** of cut S is defined as $vol_G(S) = \sum_{v \in S} deg(v)$.

The **conductance** of cut S is defined as $\Phi_G(S) = \frac{\delta(S)}{min(vol_G(S), vol_G(V \setminus S))}$. For convenience, assume that $vol_G(S) \leq vol_G(V \setminus S)$ otherwise noted.

The **conductance** of graph G, $\Phi_G = \min_{S \text{ is a cut of } G} \Phi_G(S)$. If a cut does not exist $(|V| = 1)$, then $\Phi_G = 1$.

A graph is a ϕ-**expander** if $\Phi_G \geq \phi$.

Why expander is good?

Expander is a well-connected graph. This property makes several problems easy.

Example: Decremental Spanning Forest. Deletion of tree edge disconnects the component. How to recover it from the heap of back-edges? If a graph is expander, back edge reconnects the graph with ϕ probability, which makes random sampling work.

Why expander is good?

Expander is a well-connected graph. This property makes several problems easy.

Example: Decremental Spanning Forest. Deletion of tree edge disconnects the component. How to recover it from the heap of back-edges? If a graph is expander, back edge reconnects the graph with ϕ probability, which makes random sampling work.

Example: Global Minimum Cut. In a dense unweighted graph, cut is highly likely to be unbalanced. With this intuition, the global min-cut in expander can be thought as a cut finding, where the smaller side has at most $polylog(n)$ cardinality. This is a rough idea of **Deterministic Mincut in Almost-Linear Time** (Jason Li, STOC'21)

Why expander is good?

Expander is a well-connected graph. This property makes several problems easy.

Example: Decremental Spanning Forest. Deletion of tree edge disconnects the component. How to recover it from the heap of back-edges? If a graph is expander, back edge reconnects the graph with ϕ probability, which makes random sampling work.

Example: Global Minimum Cut. In a dense unweighted graph, cut is highly likely to be unbalanced. With this intuition, the global min-cut in expander can be thought as a cut finding, where the smaller side has at most $polylog(n)$ cardinality. This is a rough idea of **Deterministic Mincut in Almost-Linear Time** (Jason Li, STOC'21)

Usually, those applications requires $\phi = 1/\log^{O(1)} m$.

Real Definition

Given a graph G and two argument, a (ϵ, ϕ)-**expander decomposition** is a partition of a vertex set $V_1 \cup V_2 \cup \ldots \cup V_k = V$ where:

(1) $G[V_i]$ is an ϕ-expander (tractable)

(2) there are at most $\epsilon|E|$ edges going through different sets (mergeable)

Lemma. For any $\phi \in (0, 1)$, $(\phi \log m, \phi)$-expander decomposition exists.

Proof by recursive algorithm. If G is not an expander, find a cut with minimum conductance, and recursively decompose $G[S], G[\overline{S}]$. As $T(m) = T(a) + T(m-a) + \phi min(a, m-a)$, $\epsilon \leq \phi \log m$.

Remark. For fixed ϕ we can't get better ϵ bound.

Real Definition

Given a graph G and two argument, a (ϵ, ϕ)-**expander decomposition** is a partition of a vertex set $V_1 \cup V_2 \cup \ldots \cup V_k = V$ where:

(1) $G[V_i]$ is an ϕ-expander (tractable)

(2) there are at most $\epsilon|E|$ edges going through different sets (mergeable)

Lemma. For any $\phi \in (0, 1)$, $(\phi \log m, \phi)$-expander decomposition exists.

Proof by recursive algorithm. If G is not an expander, find a cut with minimum conductance, and recursively decompose $G[S], G[\overline{S}]$. As $T(m) = T(a) + T(m-a) + \phi min(a, m-a)$, $\epsilon \leq \phi \log m$.

Remark. For fixed ϕ we can't get better ϵ bound.

Issue. Minimum conductance cut (aka sparsest cut) is NP-hard.

Real Definition

Given a graph G and two argument, a (ϵ, ϕ)-**expander decomposition** is a partition of a vertex set $V_1 \cup V_2 \cup \ldots \cup V_k = V$ where:

(1) $G[V_i]$ is an ϕ-expander (tractable)

(2) there are at most $\epsilon|E|$ edges going through different sets (mergeable)

Lemma. For any $\phi \in (0, 1)$, $(\phi \log m, \phi)$-expander decomposition exists.

Proof by recursive algorithm. If G is not an expander, find a cut with minimum conductance, and recursively decompose $G[S], G[\overline{S}]$. As $T(m) = T(a) + T(m - a) + \phi min(a, m - a)$, $\epsilon \leq \phi \log m$.

Remark. For fixed ϕ we can't get better ϵ bound.

Issue. Minimum conductance cut (aka sparsest cut) is NP-hard.

This paper: $(\phi \log^3 m, \phi)$, in time $O(m \log^4 m/\phi)$. Randomized.

Naive Expander Decomposition

First poly-time algorithm for Expander Decomposition **(Kannan Vempala Veta, FOCS'00)**.

Naive Expander Decomposition

First poly-time algorithm for Expander Decomposition **(Kannan Vempala Veta, FOCS'00)**.

The algorithm is simple: Since we can't find a minimum conductance cut, we just find an approximation instead.

If V is not an ϕ-expander, we find an $\tilde{O}(\phi)$ sparse cut instead.

As a tradeoff, we get $\epsilon \leq \phi \log^{1.5} n$ instead of $\epsilon \leq \phi \log n$, but it's fine.

Naive Expander Decomposition

First poly-time algorithm for Expander Decomposition **(Kannan Vempala Veta, FOCS'00)**.

The algorithm is simple: Since we can't find a minimum conductance cut, we just find an approximation instead.

If V is not an ϕ-expander, we find an $\tilde{O}(\phi)$ sparse cut instead.

As a tradeoff, we get $\epsilon \leq \phi \log^{1.5} n$ instead of $\epsilon \leq \phi \log n$, but it's fine.

Issue. If the sparse cut is unbalanced, recursion depth might grow to $\Omega(n)$, resulting in at least quadratic algorithm.

We call a cut balanced, if the smaller side have at least $polylog(n)$ size.

Attempts in between

First near-linear time algorithm for Expander Decomposition **(Spielman Tang STOC'04)**

Spectral method for sparse cut approx: Random walks, or eigenvalue / vectors of graph Laplacian.

How to get near-linear time with unbalanced cuts?

Attempts in between

First near-linear time algorithm for Expander Decomposition **(Spielman Tang STOC'04)**

Spectral method for sparse cut approx: Random walks, or eigenvalue / vectors of graph Laplacian.

How to get near-linear time with unbalanced cuts?

ST04 finds a balanced cut, or the larger side $G[\overline{S}]$ is contained in *some* expander subgraph. It can't find which subgraph it is.

Strictly speaking, ST04 **does not find** Expander Decomposition, but it is enough for many applications.

Here $\epsilon \leq \sqrt{\phi} \log^{O(1)} m$.

Attempts in between

First near-linear time algorithm for Expander Decomposition **(Spielman Tang STOC'04)**

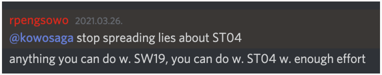
Spectral method for sparse cut approx: Random walks, or eigenvalue / vectors of graph Laplacian.

How to get near-linear time with unbalanced cuts?

ST04 finds a balanced cut, or the larger side $G[\overline{S}]$ is contained in *some* expander subgraph. It can't find which subgraph it is.

Strictly speaking, ST04 **does not find** Expander Decomposition, but it is enough for many applications.

Here $\epsilon \leq \sqrt{\phi} \log^{O(1)} m$.



Expander Pruning

Let's revisit the example of Decremental Spanning Forest.

Tractable case: Expander. Using the expander decomposition, we can maintain spanning forest for each partition.

Mergeable case: Cross-partition edges. Since we have poly-logarithmic number of such edges, we can naively consider them. (Let's not get into details.)

Problem solved?

Expander Pruning

Let's revisit the example of Decremental Spanning Forest.

Tractable case: Expander. Using the expander decomposition, we can maintain spanning forest for each partition.

Mergeable case: Cross-partition edges. Since we have poly-logarithmic number of such edges, we can naively consider them. (Let's not get into details.)

Problem solved?

Unfortunately, removal of edge alters the conductance. Each partition may not be expander anymore.

Goal: Maintain Expander Decomposition dynamically given edge deletion queries.

Expander Pruning

Removing an edge from an expander *hurts* the expander.

Expander *decays* near the position it got *hurt*.

You should *prune* the decayed part.

Expander Pruning

Removing an edge from an expander *hurts* the expander.

Expander *decays* near the position it got *hurt*.

You should *prune* the decayed part.

Maintain: For each expanders, maintain a **pruned set** $P \subseteq V$ where $G[V - P]$ is expander, and P grows slowly after each queries.

Expander Pruning: Result of this paper

Let G be a ϕ-expander with m edges. There is a deterministic algorithm that can handle at most $q \leq \phi m/10$ deletions in G. After i-th deletion, the algorithm maintains a set P_i such that:

1. $P_0 = \emptyset, P_i \subseteq P_{i+1}$
2. $vol(P_i) \leq \frac{8i}{\phi}$ and $|\delta(P_i)| \leq 4i$
3. $G_i[V - P_i]$ is a $\phi/6$ expander, where G_i is a graph with i edge deleted.

Expander Pruning: Result of this paper

Let G be a ϕ-expander with m edges. There is a deterministic algorithm that can handle at most $q \leq \phi m/10$ deletions in G. After i-th deletion, the algorithm maintains a set P_i such that:

1. $P_0 = \emptyset, P_i \subseteq P_{i+1}$
2. $vol(P_i) \leq \frac{8i}{\phi}$ and $|\delta(P_i)| \leq 4i$
3. $G_i[V - P_i]$ is a $\phi/6$ expander, where G_i is a graph with i edge deleted.

The total time for update takes $O(q \log m/\phi^2)$.

Expander Pruning: Result of this paper

Let G be a ϕ-expander with m edges. There is a deterministic algorithm that can handle at most $q \leq \phi m/10$ deletions in G. After i-th deletion, the algorithm maintains a set P_i such that:

1. $P_0 = \emptyset, P_i \subseteq P_{i+1}$
2. $vol(P_i) \leq \frac{8i}{\phi}$ and $|\delta(P_i)| \leq 4i$
3. $G_i[V - P_i]$ is a $\phi/6$ expander, where G_i is a graph with i edge deleted.

The total time for update takes $O(q \log m/\phi^2)$.

Going back to DSF. Note that P is incremental, which makes it easy to maintain a spanning tree. G_i remains an expander, so decremental method works. Glue those two spanning trees.

The Algorithm

This paper

Flow based method for sparse cut approx: Push-relabel flow algorithm.

Improves the $O(n^{o(1)})$ time bound on NSW17 (*My paper for season 1*)

Uses a simple combinatorial method, thus likely more practical. *Author believes this is implementable.*

This paper

Flow based method for sparse cut approx: Push-relabel flow algorithm.

Improves the $O(n^{o(1)})$ time bound on NSW17 (*My paper for season 1*)

Uses a simple combinatorial method, thus likely more practical. *Author believes this is implementable.*

SW19 finds a cut that is either balanced, or find a unbalanced cut such that the larger side is *nearly* ϕ-expander.

Nearly ϕ-expander can be easily processed.

Unlike ST04, it does find the Expander Decomposition.

The Algorithm

The algorithm is consisted with two major components: **Cut-Matching Game**. and **Trimming**.

The Algorithm

The algorithm is consisted with two major components: **Cut-Matching Game**. and **Trimming**.

Given (G, ϕ), the **Cut-Matching Game** certifies either $\Phi_G \geq \phi$ or returns a sparse cut of conductance $O(\phi \log^2 m)$.

The cut either have $vol(A) \geq \Omega(m/\log^2 m)$, or the larger side is a nearly-ϕ expander.

Time: $O((m \log m)/\phi)$.

The Algorithm

The algorithm is consisted with two major components: **Cut-Matching Game**. and **Trimming**.

Given (G, ϕ), the **Cut-Matching Game** certifies either $\Phi_G \geq \phi$ or returns a sparse cut of conductance $O(\phi \log^2 m)$.

The cut either have $vol(A) \geq \Omega(m/\log^2 m)$, or the larger side is a nearly-ϕ expander.

Time: $O((m \log m)/\phi)$.

Trimming takes a nearly-ϕ expander A, and returns a subgraph $A' \subseteq A$ such that A' is a $\phi/6$ expander. A' is sufficiently large, so that $V - A'$ remains small.

Time: $O(|E(A, \overline{A})| \log m/\phi^2) \leq O((m \log m)/\phi)$

The Algorithm

Given a graph G, the algorithm checks either $\Phi_G \geq 6\phi$ or not. This step is called the **Cut-Matching Game.**

If $\Phi_G \geq 6\phi$, we are over.

The Algorithm

Given a graph G, the algorithm checks either $\Phi_G \geq 6\phi$ or not. This step is called the **Cut-Matching Game.**

If $\Phi_G \geq 6\phi$, we are over.

Otherwise, the **Cut-Matching Game** returns sparse cut of conductance $O(\phi \log^2 m)$.

If this sparse cut is not balanced, it is guaranteed that the larger set is a *near*-6ϕ expander.

The Algorithm

Given a graph G, the algorithm checks either $\Phi_G \geq 6\phi$ or not. This step is called the **Cut-Matching Game.**

If $\Phi_G \geq 6\phi$, we are over.

Otherwise, the **Cut-Matching Game** returns sparse cut of conductance $O(\phi \log^2 m)$.

If this sparse cut is not balanced, it is guaranteed that the larger set is a *near*-6ϕ expander.

We use **Trimming** to make it a ϕ-expander.

Recurse in the subproblem with size at most $\Omega(m/\log^2 m)$.

The Algorithm: Proof

Assuming all the subproblems, we prove the main result.

The Algorithm: Proof

Assuming all the subproblems, we prove the main result.

Correctness: Algorithm only creates ϕ or 6ϕ expanders.

The Algorithm: Proof

Assuming all the subproblems, we prove the main result.

Correctness: Algorithm only creates ϕ or 6ϕ expanders.

Time: If all cuts are balanced, the algorithm have maximum $O(\log^3 m)$ recursion depth, and each subproblems in recursion depth is disjoint, summing to $O(m \log^4 m/\phi)$.

The trimming step reduces the size of the graph into $1/4$, so the bound remains.

The Algorithm: Proof

Assuming all the subproblems, we prove the main result.

Correctness: Algorithm only creates ϕ or 6ϕ expanders.

Time: If all cuts are balanced, the algorithm have maximum $O(\log^3 m)$ recursion depth, and each subproblems in recursion depth is disjoint, summing to $O(m \log^4 m/\phi)$.

The trimming step reduces the size of the graph into $1/4$, so the bound remains.

Edge Count: We have $O(\log^2 m)$ approximation on sparse cut. Trimming does not change this bound.

Combining this with previous small-to-large observation, we easily obtain $O(\phi \log^3 m)$.

We don't know the bounds of Trimming step, but let's not crunch numbers.

Cut-Matching Game

Cut-Matching Game

Cut-Matching Game is an abstraction of certain optimization paradigm in sparsest cut. (Khandekar, Rao, Vazirani '06)

In this paradigm, there exists an $O(\log^2 n)$ near-linear time approximation algorithm for sparsest cut. (KRV09)

As you see, this is not a new contribution from this paper.

We will only provide a high-level idea relevant to this problem. Following explanation is not a original description of Cut-Matching Game.

Cut-Matching Game

In a graph G, there is a **Cut** player, and **Matching** player.

Cut player C thinks sparsest cut exists. C gives a bisection of vertex set.

Matching player M thinks G is an expander. M finds a large matching between given bisections.

Cut-Matching Game

In a graph G, there is a **Cut** player, and **Matching** player.

Cut player C thinks sparsest cut exists. C gives a bisection of vertex set.

Matching player M thinks G is an expander. M finds a large matching between given bisections.

If M always finds a large matching for $O(\log^2 n)$ turns, we can take the union to prove that G is an expander.

If M fails in some turn, C takes the small matching, and use it to find sparse cut. Remind that matchings are flow, and it is dual to cut.

Cut-Matching Game

In a graph G, there is a **Cut** player, and **Matching** player.

Cut player C thinks sparsest cut exists. C gives a bisection of vertex set.

Matching player M thinks G is an expander. M finds a large matching between given bisections.

If M always finds a large matching for $O(\log^2 n)$ turns, we can take the union to prove that G is an expander.

If M fails in some turn, C takes the small matching, and use it to find sparse cut. Remind that matchings are flow, and it is dual to cut.

How to find bisection? It uses spectral methods, that does sampling based on the matching stack M provided.

How to guarantee near-6ϕ expander? Original work of HRV09 doesn't guarantee it. The author does some modification for it. It is in Appendix, so I will skip it.

Trimming

Trimming

Trimming step keeps the expander decomposition balanced.

Key technical contribution of this paper.

Trimming

Trimming step keeps the expander decomposition balanced.

Key technical contribution of this paper.

Assume that all the volumes are w.r.t G but not w.r.t induced subgraph.

Nearly Expander. $A \subset V$ is a nearly ϕ-expander in G if,
$\forall S \subseteq A, vol(S) \leq vol(A)/2$: $|E(S, V-S)| \geq \phi vol(S)$.

Note that A is a ϕ-expander if $|E(S, A-S)| \geq \phi vol(S)$.

Nearly expander is a relaxed definition, where edges outside of induced subgraphs are counted.

Trimming

Trimming step keeps the expander decomposition balanced.

Key technical contribution of this paper.

Assume that all the volumes are w.r.t G but not w.r.t induced subgraph.

Nearly Expander. $A \subset V$ is a nearly ϕ-expander in G if,
$\forall S \subseteq A, vol(S) \leq vol(A)/2$: $|E(S, V - S)| \geq \phi vol(S)$.

Note that A is a ϕ-expander if $|E(S, A - S)| \geq \phi vol(S)$.

Nearly expander is a relaxed definition, where edges outside of induced subgraphs are counted.

Trimming finds $A' \subseteq A$ such that

$\forall S \subseteq A', vol(S) \leq vol(A')/2$: $|E(S, A' - S)| \geq \phi vol(S)/6$.

Trimming

Best scenario is where we don't do any trimming and certify the nearly expander as expander.

Let's try this. Suppose not. A is a ϕ-near expander, but not $(\phi/6)$-expander.

Trimming

Best scenario is where we don't do any trimming and certify the nearly expander as expander.

Let's try this. Suppose not. A is a ϕ-near expander, but not $(\phi/6)$-expander.

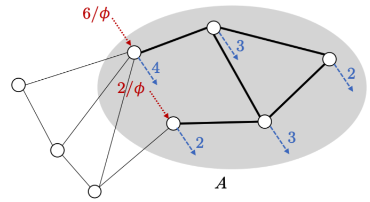
It means the relaxation is heavily abused.

It means that, there exists a cut S where most edges go outside of A.

$|E(S, V - A)| \geq 5|E(S, A - S)|$.

Capture the gap with flows

Consider the following flow instance.

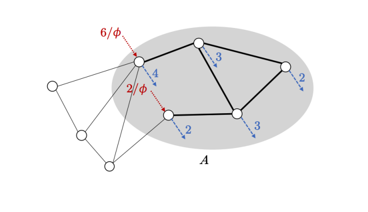


Each edge in $E(A, V - A)$ is replaced to $2/\phi$ supply from source.

Each vertex have $deg_G(v)$ demand from sink.

Each internal edges of A have $2/\phi$ capacity.

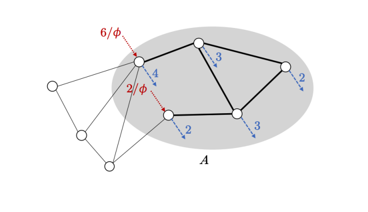
Capture the gap with flows



The sum of supply is $\frac{2}{\phi}|E(A, V - A)|$.

Claim. If A is not $\phi/6$ expander, max flow is less than the sum of supply.

Capture the gap with flows



The sum of supply is $\frac{2}{\phi}|E(A, V-A)|$.

Claim. If A is not $\phi/6$ expander, max flow is less than the sum of supply.

Proof. There exists a set S where $|E(S, A-S)| \leq \frac{|E(S,V-A)|}{5}$.

Flow toward $S \geq \frac{2}{\phi}|E(S, V-A)| \geq \frac{1}{\phi}|E(S, V-A)| + \frac{5}{\phi}|E(S, A-S)|$

Flow from S
$\leq vol(S) + \frac{2}{\phi}E(S, A-S) \leq \frac{1}{\phi}(E(S, V-A) + E(S, A-S)) + \frac{2}{\phi}E(S, A-S)$

So something is lost on S.

Capture the gap with flows

In other words, if the maximum flow satisfies all supply, we can prove that A is a $\phi/6$ expander.

Otherwise, we failed to direct all the supply.

Capture the gap with flows

In other words, if the maximum flow satisfies all supply, we can prove that A is a $\phi/6$ expander.

Otherwise, we failed to direct all the supply.

We find a minimum cut.

Note that the minimum cut saturates all edges in the flow.

So if we take the source-side cut from A, all supplies are saturated in the current flow solution.. **WHAT?**

Capture the gap with flows

In other words, if the maximum flow satisfies all supply, we can prove that A is a $\phi/6$ expander.

Otherwise, we failed to direct all the supply.

We find a minimum cut.

Note that the minimum cut saturates all edges in the flow.

So if we take the source-side cut from A, all supplies are saturated in the current flow solution.. **WHAT?**

Subquadratic Trimming. Use Dinic or Madry13 to compute the maximum flow. If the cut is not trivial, prune the source-side cut.

Capture the gap with flows

Now let's bound the size of source-side cut.

Note that the source-side cut S must have its sink saturated, because it is a cut.

The sum of saturated demand is (S).

The sum of total supply is $\frac{2}{\phi}|E(A, V - A)|$

$vol(S) \leq \frac{2}{\phi}|E(A, V - A)|$

$vol(A') \geq vol(A) - \frac{2}{\phi}|E(A, V - A)|$

Fortunately, the cut-matching guarantees $|E(A, V - A)| \leq \phi m/10$, so $A' - A$ does not blow up.

Capture the gap with flows

Let's also prove that the conductance does not blow up.

Observe that every edge in $E(A', V - A')$ is routed: They are exactly the ones that disconnects the artificial source from sink.

They are all distinct, so they can't sum to the total supply.

$|E(A', V - A')| \leq |E(A, V - A)|$

Since the volue didn't blow up, conductance does not blow up either.

We established that Trimming $\rightarrow$ Expander Decomposition.

Untold stories

Near-linear time trimming. To achieve near-linear time, we should use approximate max-flow techniques. This makes it faster to compute a single cut, but pruning an approximate source-side cut does not make A an $\phi/6$ expander.

Untold stories

Near-linear time trimming. To achieve near-linear time, we should use approximate max-flow techniques. This makes it faster to compute a single cut, but pruning an approximate source-side cut does not make A an $\phi/6$ expander.

However, if we prune the cut repeatedly, it can be shown to converge.

This gives an extra overhead, so we have to design an efficient algorithm that does not compute flow from scratch after cut removal.

This efficient trimming algorithm is based on push-relabel flow. It seems pretty complicated. I will skip it.

Untold stories

Near-linear time trimming. To achieve near-linear time, we should use approximate max-flow techniques. This makes it faster to compute a single cut, but pruning an approximate source-side cut does not make A an $\phi/6$ expander.

However, if we prune the cut repeatedly, it can be shown to converge.

This gives an extra overhead, so we have to design an efficient algorithm that does not compute flow from scratch after cut removal.

This efficient trimming algorithm is based on push-relabel flow. It seems pretty complicated. I will skip it.

Trimming implies Expander Pruning. High-level idea is to simply apply near-linear time trimming after edge removal, and append a pruning set.

Concluding Remarks

Concluding Remarks

If you want a better explanation on this topic, I recommend these:

https://www.youtube.com/watch?v=Q8hxG11zVdc

https://northwestern.hosted.panopto.com/Panopto/Pages/Embed.aspx?id=5b72411f-0b43-40c0-b25f-ab0d013d85db

Concluding Remarks

If you want a better explanation on this topic, I recommend these:

https://www.youtube.com/watch?v=Q8hxG11zVdc

https://northwestern.hosted.panopto.com/Panopto/Pages/Embed.aspx?id=5b72411f-0b43-40c0-b25f-ab0d013d85db

Thank you for listening! See you after the army break...