Tree Flow Sparsifier via Expander Hierarchy

December 11, 2025

1 Cut/Flow Sparsifier w.r.t. Terminals

Definition 1.1. Given a graph $G = (V, E)$, a **cut sparsifier** H of G for terminal set S with quality q satisfies the following

1. $V(H) \supseteq S$
2. For every $A, B \subseteq S$, we have

$$\text{mincut}_G(A, B) \leq \text{mincut}_H(A, B) \leq q \cdot \text{mincut}_G(A, B).$$

Example 1.2. We saw this concept from the previous class.

- Let $G\{U\}$ be a graph with boundary
- Let H be obtained from $G\{U\}$ as follows
  - Let $U_1, \ldots, U_k$ be a α-boundary-linked decomposition of $G\{U\}$.
  - Contract each U_i into a vertex u_i.
  - Note that the size of H is just proportional to the boundary $|E(H)| = O(|\partial_G \langle U \rangle|)$
- We showed that H is a cut sparsifier of $G\{U\}$ for terminal set $\partial_G \langle U \rangle$ with quality $\frac{1}{\alpha}$.

Definition 1.3. Given a graph $G = (V, E)$, a **flow sparsifier** H of G for terminal set S with quality q satisfies the following

1. $V(H) \supseteq S$
2. Let $D = (S, E')$ be a demand between terminals S.

$$\mathsf{mcf}(H, D) \leq \mathsf{mcf}(G, D) \leq q \cdot \mathsf{mcf}(H, D)$$

   where $\mathsf{mcf}(G, D)$ is the minimum congestion for routing D in G. In words, if D is routable in G, then D is routable in H. Conversely, if D is routable in H, then D is routable in G with congestion q.

Exercise 1.4. [TODO in scribe node: add the proof of this too.] Show that the two concept are equivalent up to logarithmic factor:

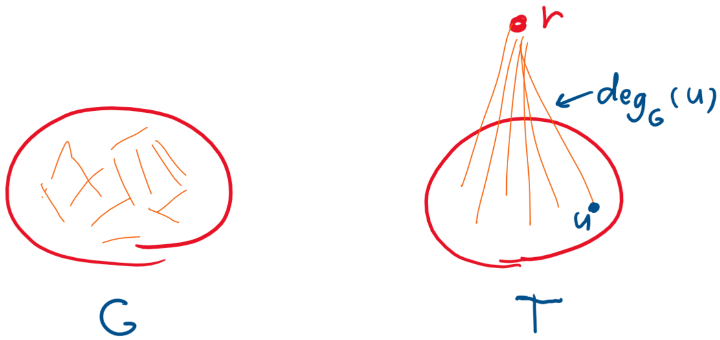
1. if H is a flow sparsifier of G with quality q, then H is a cut sparsifier of G with quality q.
2. if H is a cut sparsifier of G with quality q, then H is a flow sparsifier of G with quality $O(q \log n)$.

2 Flow Sparsifier of Expanders = Star

- Does it make sense to let terminal set be V?
  - We cannot reduce the number of vertices for sure.
  - But what is a non-trivial thing we can do?
- Let's look at this example. For any expander can be simplified to a star!

Example 2.1. Let $G = (V, E)$ be a ϕ-expander.

- Let T be a star where leaves corresponds to V. Let r be the root of the star.
- For each tree edge (v, r), set the capacity in T as $c_T(v, r) = \deg_G(v)$.
- T is a flow sparsifier of G for terminal set V with quality $\frac{\log(n)}{\phi}$.



Proof. Let D be a demand on terminal V. We argue two directions

- If D is routable in G, then D is routable in T.
  - Given D in T, just route everything to the root. The flow paths "match". Done.
  - No congestion in T
    - Note that the flow on (u, r) is exactly $\deg_D(u)$.
    - As D is routable in G, $\deg_D(u) \leq \deg_G(u)$ for all $u \in V$
    - But $\deg_G(u) = c_T(v, r)$ by construction.
- If D is routable in T, then D is routable in G with congestion $O(\log(n)/\phi)$.
  - If D is routable in T, then D is $\deg_G$-restricted. (i.e. $\deg_D(u) \leq \deg_G(u)$).
  - As G is a ϕ-expander, any $\deg_G$-restricted demand is routable with congestion $O(\log(n)/\phi)$.

$\square$

- From now, we just say "H is a flow sparsifier of G" when the terminal set is $V(G)$.

3 Flow Sparsifier from Boundary-linked Expander Decomposition

- Now, let's try to apply this idea for expanders to arbitrary graphs
  - How?
  - As usual, expander decomposition.
  - But we will need boundary-linkedness too.

3.1 Recap: Boundary-linked expander decomposition

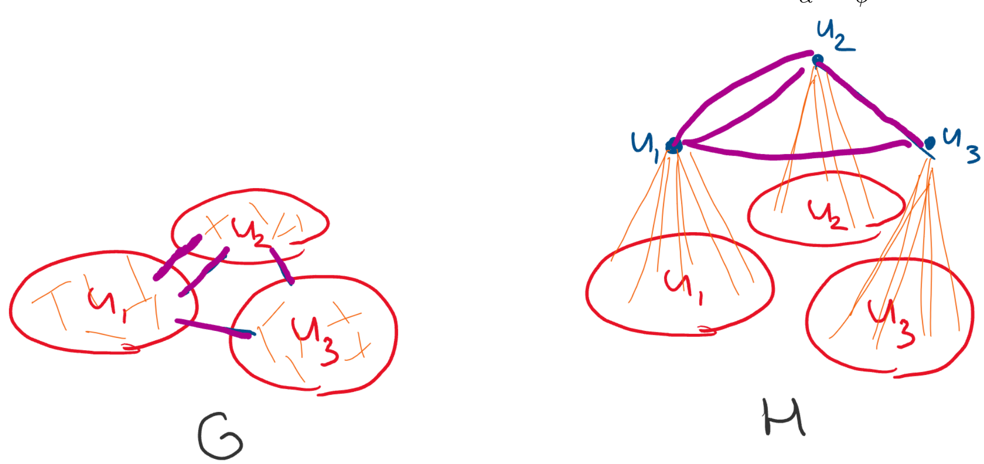
- **Input:**
  - an induced subgraph $G\{U\}$ with boundary vertices
  - a parameter ϕ.
- **Output:** a partition of $U_1, \ldots, U_k$ of U such that
  - U_i is (α, ϕ)-linked where $\alpha = \Theta(1/\log m)$. That is,
    - $G\{U\}$ is α-boundary-linked, and
    - $G\{U\}$ is a ϕ-expander.
  - $\sum_i |\partial_G \langle U_i \rangle \setminus \partial_G \langle U \rangle| = O(|\partial_G \langle U \rangle| + \phi \mathrm{vol}_G(U) \log m)$
- Throughout the lecture, think of $\alpha = \Theta(1/\log m)$ but $\phi = \Theta(1/2^{\sqrt{\log n}}) = 1/n^{o(1)}$. So $\phi \ll \alpha$.

3.2 Sparsifier Construction

Lemma 3.1. *Let $G = (V, E)$ be a graph. Do the following:*

1. *Compute a α-boundary-linked ϕ-expander decomposition $\mathcal{U} = \{U_1, \ldots, U_k\}$ of G.*
2. *Contract each U_i into a vertex u_i. Let $G_{\mathcal{U}}$ be the contracted graph.*
3. *Let H be obtained from $G_{\mathcal{U}}$ as follows. For each $U_i \in \mathcal{U}$, attach a star rooted at u_i in $G_{\mathcal{U}}$ as in Example 2.1. More formally, for each $w \in U_i$, add (w, u_i) with capacity $\deg_G(w)$ into $G_{\mathcal{U}}$.*

Then, we have that H is a flow sparsifier of G with quality $O((\frac{1}{\alpha} + \frac{1}{\phi}) \log m) = O(\frac{1}{\phi} \log m)$.

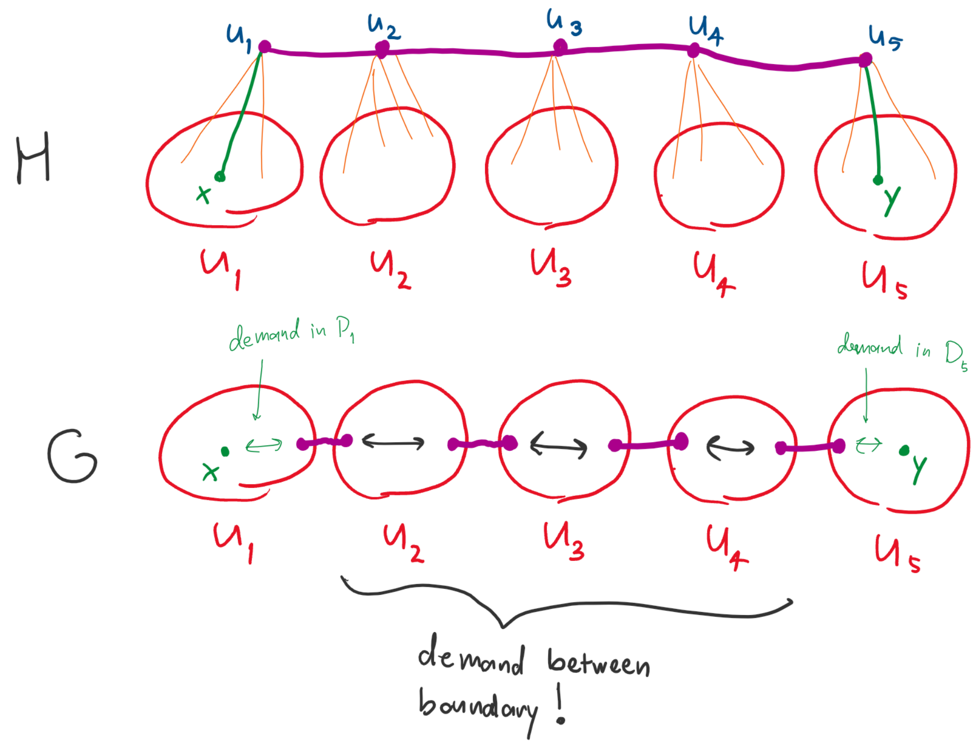


Proof. Let D be a demand on terminal V. Let $D_{\mathcal{U}}$ be a *projection* of D to $V(G_{\mathcal{U}})$, i.e., for $u_i, u_j \in V(G_{\mathcal{U}})$, we define

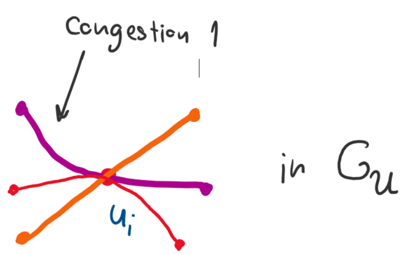
$$D_{\mathcal{U}}(u_i, u_j) = \sum_{x \in U_i, y \in U_k} D(x, y).$$

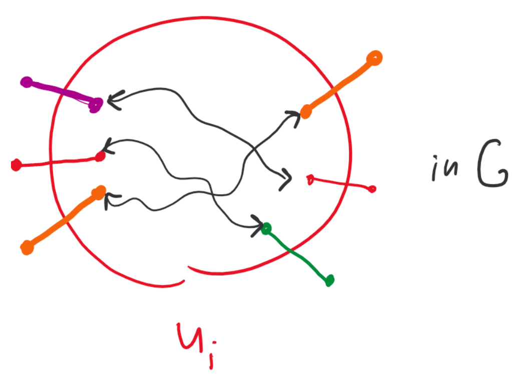
There are two directions:

1. Suppose D is routable in G. Then D is routable in H.
   - For each demand $(x, y) \in E(D)$ where $x \in U_i$ and $y \in U_j$, we split the demand into three parts $x \leftrightarrow u_i$, $u_i \leftrightarrow u_j$, $u_j \leftrightarrow y$.
     (a) The first/third parts, even after summing over all demand pairs, can be routed with congestion using the star edges.
     (b) For the second part, after summing over all demand pairs, this is to route $D_{\mathcal{U}}$ in $G_{\mathcal{U}}$.
       - $D_{\mathcal{U}}$ is routable in $G_{\mathcal{U}}$.
       - **Exercise:** $D_{\mathcal{U}}$ is routable in $G_{\mathcal{U}}$ iff D is routable in G when the edge capacity inside each U_i is infinite.
   - The edges of H used in the two steps above are disjoint. So there is no congestion.
2. Suppose D is routable in H. Then D is routable in G with congestion $O((\frac{1}{\alpha} + \frac{1}{\phi}) \log m)$.
   - We write $D = \sum_{U_i \in \mathcal{U}} D_i + D_{dif}$ where
     - D_i contains all demand pairs (x, y) where both $x, y \in U_i$
     - D^{dif} contains all demand pairs (x, y) where x, y are in different parts.
   - Each D_i is routable in $G[U_i]$ with congestion $O(\frac{\log m}{\phi})$.
     - D_i is $\deg_{G\{U\}}$-restricted. (D_i may not be $\deg_{G[U_i]}$-restricted)
     - $G\{U_i\}$ is a ϕ-expander. So any $\deg_{G\{U\}}$-restricted demand is routable in $G\{U_i\}$ with congestion $O(\frac{\log m}{\phi})$.
     - Actually D_i is routable in $G[U_i]$ with congestion $O(\frac{\log m}{\phi})$.
       * Boundary edges of $G\{U_i\}$ are not used for routing inside $G\{U_i\}$ anyway.
   - Now consider D^{dif}.
     - Let F_H be the flow that routes D^{dif} in H.
     - Let $F_{G_{\mathcal{U}}}$ be the flow in $G_{\mathcal{U}}$ is obtained from F_H by restricting to edges in $G_{\mathcal{U}}$ (note that $H = G_{\mathcal{U}}$ + stars).
     - Observe that $F_{G_{\mathcal{U}}}$ routes $D_{\mathcal{U}}^{dif}$ in $G_{\mathcal{U}}$.
     - $F_{G_{\mathcal{U}}}$ can be viewed a "broken flow" in G.
       * Purple flow in picture below is the flow path of $F_{G_{\mathcal{U}}}$.
       * The green edge extends the flow-path to endpoints in $V(G)$



- To complete the broken paths from $F_{G_{\mathcal{U}}}$ in G, this induces a new demand.
  * The total black demand (from the whole flow-path except at the initial/last cluster) induces the 1-restricted demand between boundary vertices of each $G\{U_i\}$.





    · Can route this black part using $O(\frac{\log n}{\alpha})$ inside each $G[U_i]$
    · because $G\{U_i\}$ is α-boundary-linked.
  * The total green demand (from the flow-path at the initial/last cluster) induces the $\deg_{G\{U_i\}}$-restricted demand between vertices inside $G\{U_i\}$.

    · Can route this green part using $O(\frac{\log n}{\phi})$ inside each $G[U_i]$.
    · This is just exactly situation as when we want to route demand D_i inside $G[U_i]$.
  * Done by concatenation flow paths.

□

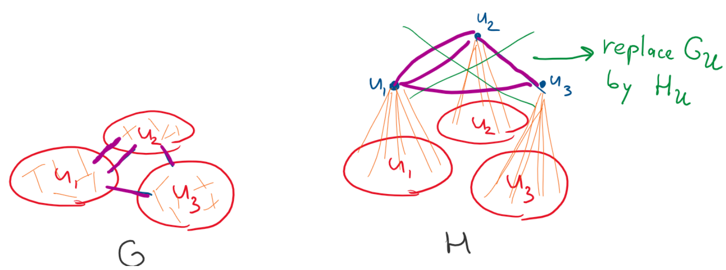
4 Recurse!

- Summary: Given any graph G and its boundary-linked ϕ-expander decomposition $\mathcal{U}$: we can get flow sparsifier of G as follows:
  - Contract graph G into $G_{\mathcal{U}}$ (the part is small $|E(G_{\mathcal{U}})| = \tilde{O}(\phi|E(G)|)$).
  - Attach the star on each expander $U \in \mathcal{U}$ (this part is very simple).
- This should inspire you to recurse on $G_{\mathcal{U}}$ to simplify this smaller graph $G_{\mathcal{U}}$ further.
- The following theorem is the key tool:

Theorem 4.1. *Let $G = (V, E)$ be a graph. Do the following:*

1. *Compute a α-boundary-linked ϕ-expander decomposition $\mathcal{U} = \{U_1, \ldots, U_k\}$ of G.*
2. *Contract each U_i into a vertex u_i. Let $G_{\mathcal{U}}$ be the contracted graph.*
3. *Let $H_{\mathcal{U}}$ be a flow sparsifier of $G_{\mathcal{U}}$ with quality q.*
4. *Let H be obtained from $H_{\mathcal{U}}$ as follows. For each $U_i \in \mathcal{U}$, attach a star rooted at u_i in $H_{\mathcal{U}}$ as in Example 2.1. More formally, for each $w \in U_i$, add (w, u_i) with capacity $\deg_G(w)$ into $H_{\mathcal{U}}$.*

Then, we have that H is a flow sparsifier of G with quality $O((\frac{q}{\alpha} + \frac{1}{\phi}) \log m)$.



- It will be very crucial that the factor q appears only at the $\frac{1}{\alpha}$ term and not $\frac{1}{\phi}$.
- Our plan:
  1. Specify the recursive algorithm more precisely. When we get is something called the **expander hierarchy**.
  2. Assuming Theorem 4.1, show that expander hierarchy is a flow sparsifier with good quality.
  3. Proof Theorem 4.1.

4.1 The recursive structure: Expander Hierarchy

- Let's be more specific what we mean by recursive algorithm.

  1. Initialize $G_0 = G$.
  2. For $i = 0, 1, \dots$
     - Compute the decomposition $\mathcal{U}_i$ of G_i.
     - Set $G_{i+1} \leftarrow G_{\mathcal{U}_i}$.
     - If G_{i+1} has a single vertex, break.
  3. Let h be the maximum level i. Let $T_h = G_h$ be a trivial flow sparsifier of G_h.
  4. For $i = h-1, \dots, 1$
     - Given $G_{\mathcal{U}_i}$ and a flow sparsifier T_{i+1} of $G_{\mathcal{U}_i}$, compute a flow sparsifier T_i of G_i using Theorem 4.1.

- At the end, T_0 is a flow sparsifier of G.

  - T_0 is a tree! This looks great. It is very simple.

- We call this tree T_0 **the expander hierarchy** of G.

  - If each $\mathcal{U}_i$ is an α-boundary-linked ϕ-expander decomposition of G_i.
  - Then we say that T_0 is a (α, ϕ)-expander hierarchy of G.

- In other words, expander hierarchy obtained by recursively computing α-boundary-linked ϕ-expander decomposition in the contracted graph.

4.2 Quality of Expander Hierarchy

- First, we point out why it is so crucial that

  - The quality in Theorem 4.1 is $O((\frac{q}{\alpha} + \frac{1}{\phi}) \log m)$ not just $O(q(\frac{1}{\alpha} + \frac{1}{\phi}) \log m) = O(\frac{q}{\phi} \log m)$.
  - Note that: $O(q(\frac{1}{\alpha} + \frac{1}{\phi}) \log m)$ is quite natural to expect.
    - The non-recursive version in Lemma 3.1 gives $O((\frac{1}{\alpha} + \frac{1}{\phi}) \log m)$ quality.
    - We recurse on the sparsifier of quality q. So we should pay an extra factor of q.

- Why $O(\frac{q}{\phi} \log m)$ is not good enough?

  - By induction, for each i, T_i would be a flow sparsifier of G_i with quality $O(\frac{\log m}{\phi})^{h-i}$.
  - $h = \log_{\tilde{O}(\phi)} m \geq \log_{\phi} m$ because the graph size reduces by $\tilde{O}(\phi)$ factor for each level.
  - So the quality of T_0 is $\Theta(\frac{\log m}{\phi})^h \geq \Theta(m)$... very bad.

Exercise 4.2. Given a unit-capacity graph G, show that any spanning tree of G is a flow sparsifier of G with quality $O(m)$ or $O(m \log m)$. Given a capacitated graph, show that any maximum spanning tree of G is a flow sparsifier of G with quality $O(m)$ or $O(m \log m)$.

- Why $O((\frac{q}{\alpha} + \frac{1}{\phi}) \log m)$ is good enough?

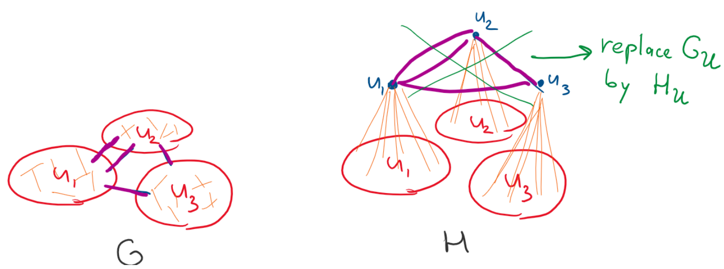
  - By induction, we can prove that T_0 has quality $O((\frac{\log m}{\alpha})^h \frac{1}{\phi})$.

        * Basically, the quality loss per level is now $O(\frac{\log m}{\alpha}) \ll O(\frac{\log m}{\phi})$.
  - By setting $\phi = 1/2^{\sqrt{\log m}}$ we have $h = \log_{\tilde{O}(\phi)} m = O(\sqrt{\log m} \log\log m)$.
  - So the quality of T_0 is $2^{O(\sqrt{\log m} \log\log m)} = n^{o(1)}$.

- Now, we see how crucial it is that factor q appears only at the $\frac{1}{\alpha}$ term and not $\frac{1}{\phi}$.
- Let's prove it.

4.3 Proof of Theorem 4.1

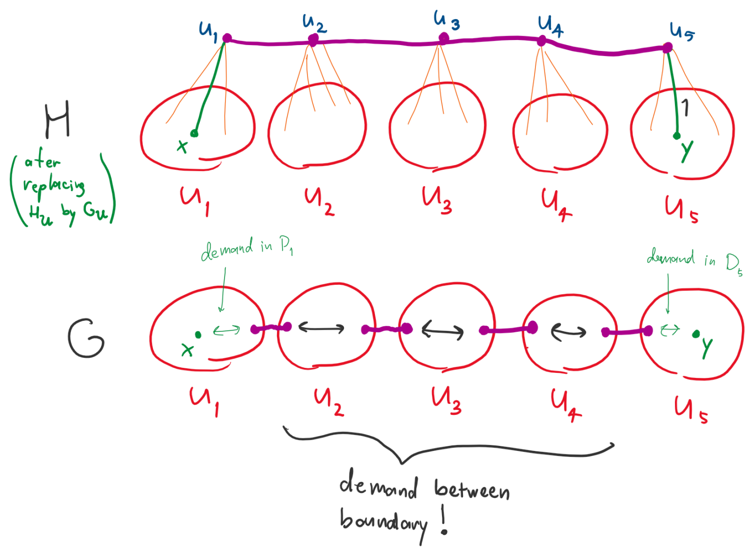
- The prove actually is quite simple. Just need to inspect the proof of the non-recursive Lemma 3.1.



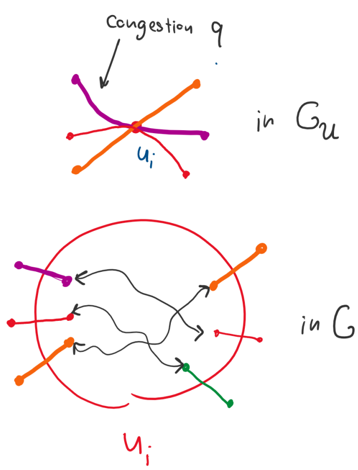
I copied the proof below and high-light the change. There are two directions:

1. Suppose D is routable in G. Then D is routable in H.
   - For each demand $(x, y) \in E(D)$ where $x \in U_i$ and $y \in U_j$, we split the demand into three parts $x \leftrightarrow u_i$, $u_i \leftrightarrow u_j$, $u_j \leftrightarrow y$.
     (a) The first/third parts, even after summing over all demand pairs, can be routed with congestion using the star edges.
     (b) For the second part, after summing over all demand pairs, this is to route $D_{\mathcal{U}}$ in $H_{\mathcal{U}}$.
       - $D_{\mathcal{U}}$ is routable in $G_{\mathcal{U}}$.
       - As $H_{\mathcal{U}}$ is a flow sparsifier of $G_{\mathcal{U}}$, $D_{\mathcal{U}}$ is routable in $H_{\mathcal{U}}$
   - The edges of H used in the two steps above are disjoint. So there is no congestion.
2. Suppose D is routable in H. Then D is routable in G with congestion $O((\frac{q}{\alpha} + \frac{1}{\phi}) \log m)$.
   - We write $D = \sum_{U_i \in \mathcal{U}} D_i + D^{dif}$ where
     - D_i contains all demand pairs (x, y) where both $x, y \in U_i$
     - D^{dif} contains all demand pairs (x, y) where x, y are in different parts.
   - Each D_i is routable in $G[U_i]$ with congestion $O(\frac{\log m}{\phi})$.
     - D_i is $\deg_{G\{U\}}$-restricted. (D_i may not be $\deg_{G[U_i]}$-restricted)
     - $G\{U_i\}$ is a ϕ-expander. So any $\deg_{G\{U\}}$-restricted demand is routable in $G\{U_i\}$ with congestion $O(\frac{\log m}{\phi})$.
     - Actually D_i is routable in $G[U_i]$ with congestion $O(\frac{\log m}{\phi})$.
        * Boundary edges of $G\{U_i\}$ are not used for routing inside $G\{U_i\}$ anyway.
   - Now consider D^{dif}.

– Let F_H be the flow that routes D^{dif} in H.
– Let $F_{H_\mathcal{U}}$ be the flow in $H_\mathcal{U}$ is obtained from F_H by restricting to edges in $H_\mathcal{U}$
– Observe that $F_{H_\mathcal{U}}$ routes $D^{dif}_\mathcal{U}$ in $H_\mathcal{U}$.
– As $H_\mathcal{U}$ is a flow sparsifier of $G_\mathcal{U}$ with quality q, there is a flow $F_{G_\mathcal{U}}$in $G_\mathcal{U}$ that routes $D^{dif}_\mathcal{U}$ with congestion q.
– $F_{G_\mathcal{U}}$ can be viewed a “broken flow” in G.
  * Purple flow in picture below is the flow path of $F_{G_\mathcal{U}}$.
  * The green edge extends the flow-path to endpoints in $V(G)$



– To complete the broken paths from $F_{G_\mathcal{U}}$ in G, this induces a new demand.
  * The total black demand (from the whole flow-path except at the initial/last cluster) induces the q-restricted demand between boundary vertices of each $G\{U_i\}$.



    · The demand is q-restricted because $F_{G_\mathcal{U}}$ has congestion q
    · Can route this black demand using $O(q \cdot \frac{\log m}{\alpha})$ congestion inside each $G[U_i]$
    · because $G\{U_i\}$ is α-boundary-linked.
  * The total green demand (from the flow-path at the initial/last cluster) induces the $\deg_{G\{U_i\}}$-restricted demand between vertices inside $G\{U_i\}$.

- This is not a $q \cdot \deg_{G\{U_i\}}$-restricted demand because the endpoints u_i of the flow-path in $G_{\mathcal{U}}$ receives flow equals to its demand. There is no blow-up factor of q here.
- Can route this green part using $O(\frac{\log n}{\phi})$ inside each $G[U_i]$.
- This is just exactly situation as when we want to route demand D_i inside $G[U_i]$.

* Done by concatenation flow paths.

5 Tree Flow Sparsifier

- When a cut/flow sparsifier of G is a tree, then we call it a tree cut/flow sparsifier.
- This is also called **Räcke Tree** because its first construction with polylog(n) quality was discovered by Harald Räcke[1]
- We saw that the expander hierarchy gives $n^{o(1)}$-quality tree flow sparsifier.

5.1 State of the Art

- Polynomial time (slow)
  - Tree cut sparsifier with quality $O(\log^{1.5} n \log\log n)$ (or $O(\log n \log\log n)$ for existence only)[2]
  - Tree flow sparsifier with quality $O(\log^2 n \log\log n)$[3]
- Almost linear time
  - Tree flow sparsifier with quality $O(\log^4 n)$[4]
  - Tree flow sparsifier with quality $n^{o(1)}$.[5]
    * Today: Expander hierarchy.
    * Simplest construction. Can be made dynamic.
- All known constructions guarantee that T has low depth $O(\log n)$. This is useful.
- Open:
  - $O(\log n)$ quality is possible?
  - Simple algorithm with polylog(n) quality.

[1] https://home.ttic.edu/~harry/pdf/min_congestion.pdf
[2] https://link.springer.com/chapter/10.1007%2F978-3-662-44777-2_64
[3] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.640.8407&rep=rep1&type=pdf
[4] https://epubs.siam.org/doi/pdf/10.1137/1.9781611973402.17
[5] https://arxiv.org/abs/2005.02369

6 Perspective: Trees are good

- Trees are very computational friendly.
  - Think of top tree: we can compute and maintain almost everything on trees very quickly.
- A theme in graph algorithms: compute tree representations that faithfully preserve fundamental properties of a given graph.
  - spanning forests (preserving connectivity)
  - shortest path trees (preserving distances from a source)
  - Gomory-Hu trees (preserving pairwise minimum cuts)
  - low stretch spanning trees (preserving average distances between pairs of vertices)
  - treewidth decomposition (preserving “tree-like” structure)
- Tree flow sparsifier is an astonishingly strong tree.
  - It approximately preserves the values of *all cuts*.
  - Amazing that it exists at all.
- Today, we just saw a simplest known way to compute tree flow sparsifier.