Edge-Disjoint Paths in Networks

(Part 3)

Sanjeev Khanna
University of Pennsylvania

Recap

Expander Embedding

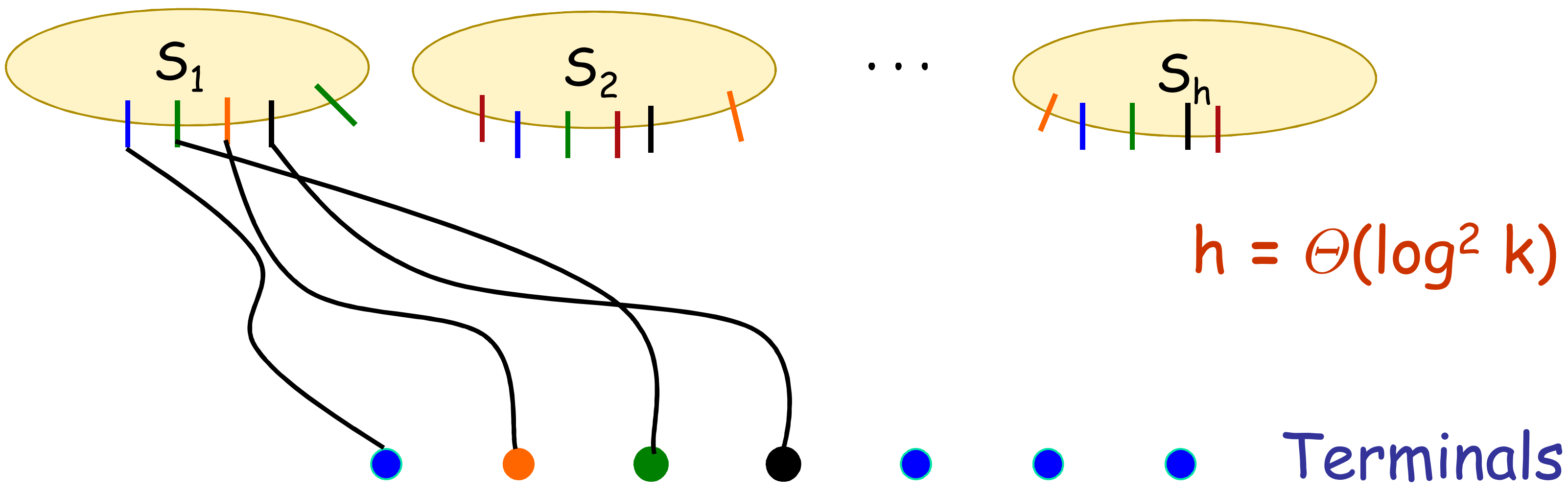
Starting Point:

- A graph G(V,E) that has a well-linked terminal set X of size k, the degree of each vertex in the graph is at most 4, and the degree of each terminal is 1.

Goal:

- Embed a low-degree expander of size k/polylog(k) on the terminals with constant congestion on the edges.

Idea 1: Good Family of Sets

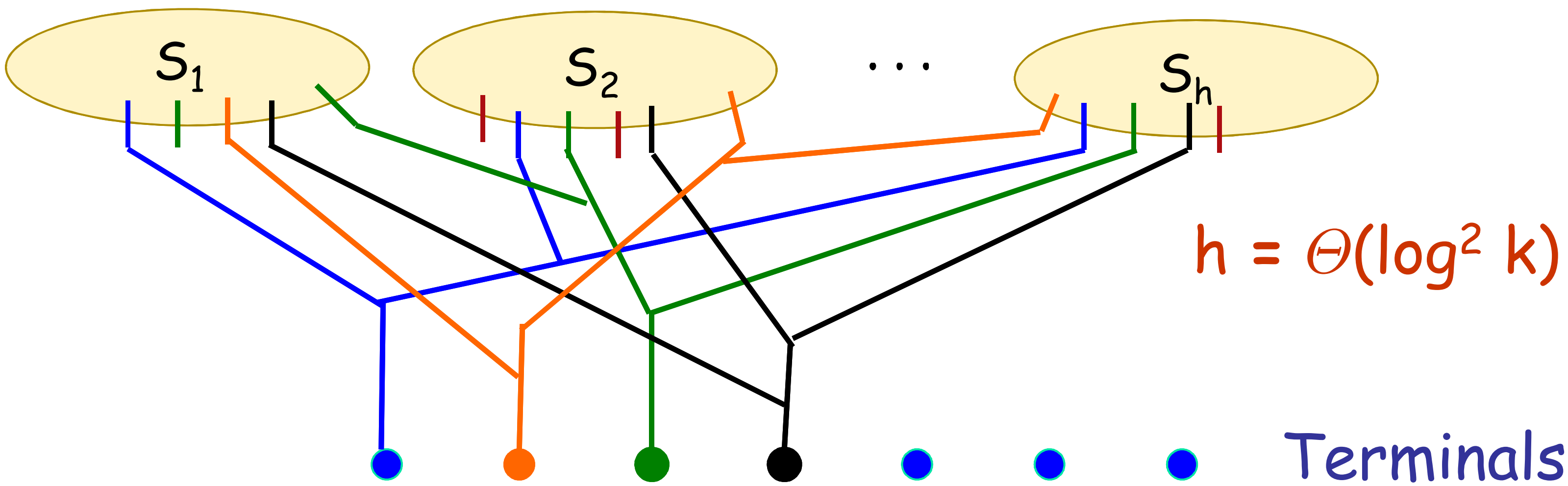


$|out(S_j)|$ = # of edges on boundary of S_j = k/polylog(k).

Each S_j is well-linked w.r.t. its boundary $out(S_j)$.

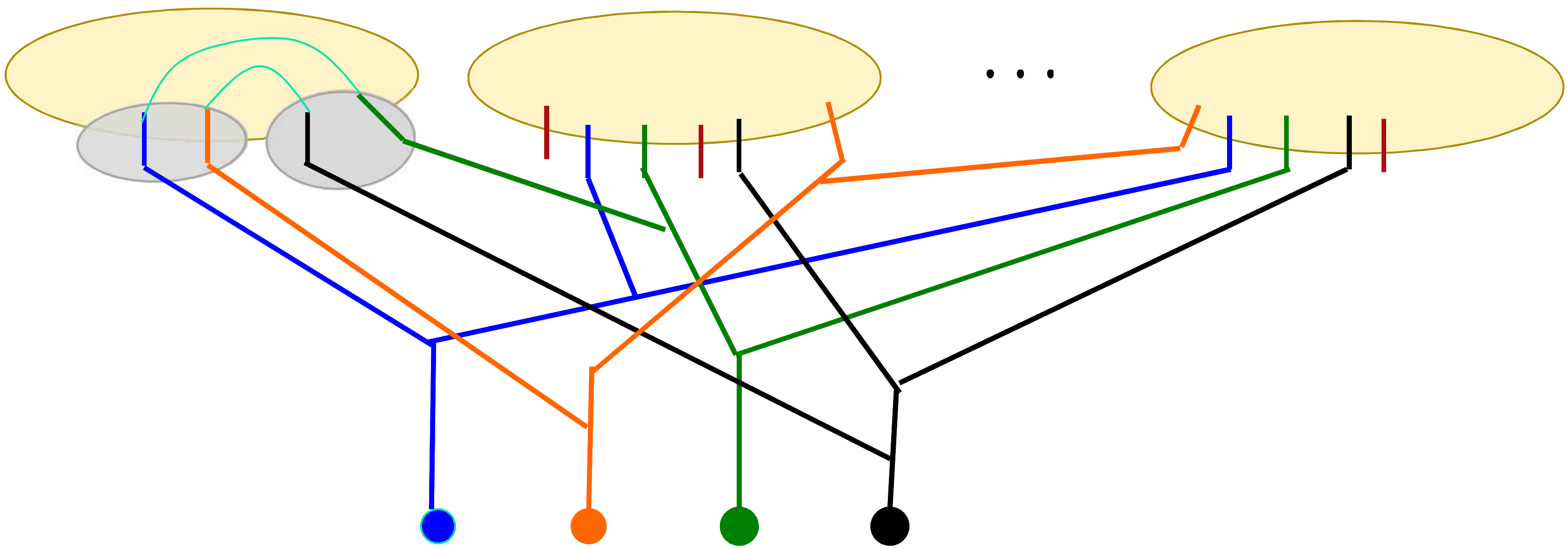
Each S_j can connect by edge-disjoint paths to k/polylog(k) terminals.

Idea 2: Routing Trees

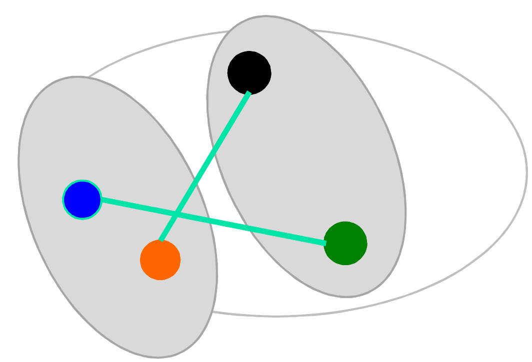


For each terminal t_i, there is a tree T_i that spans t_i and a distinct edge e_{ij} in $out(S_j)$ for each j.

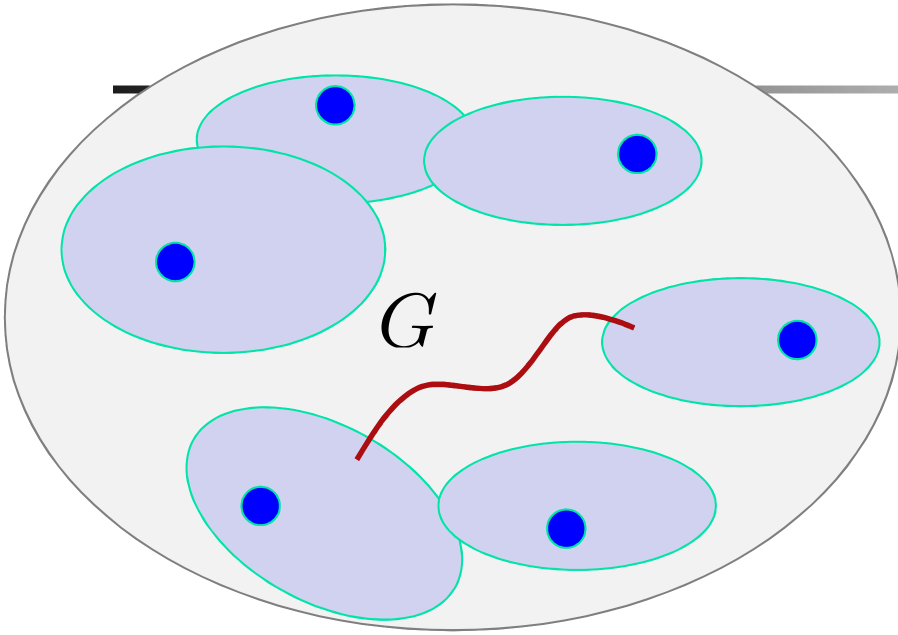
Embedding an Expander

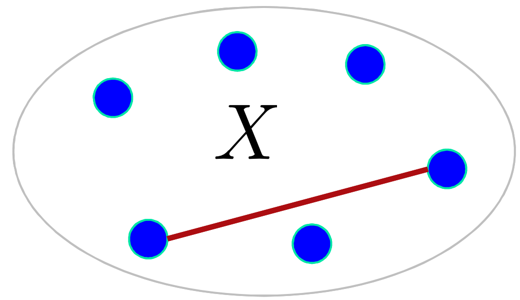


Implementing one round of the cut-matching game.



Routing on the Embedded Expander





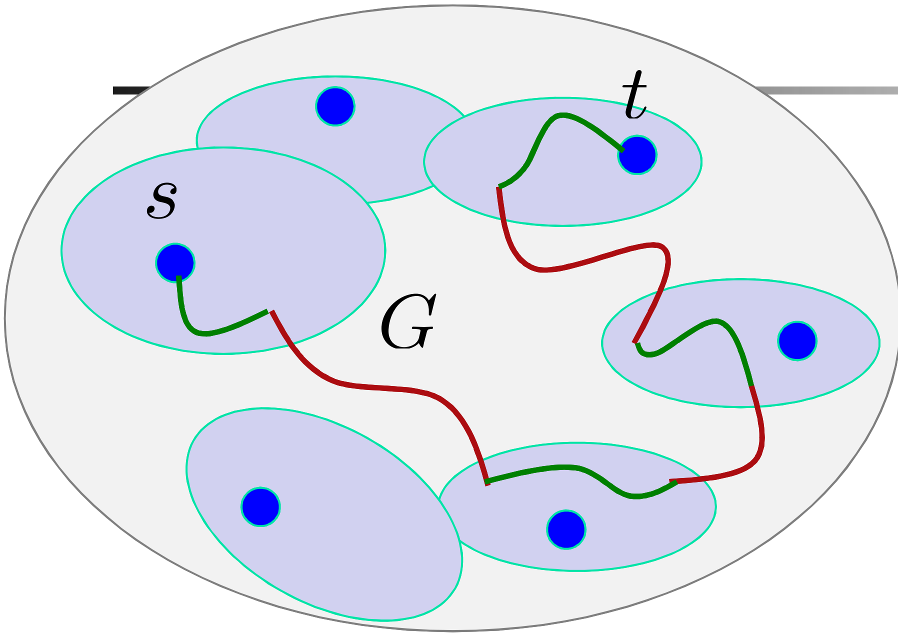
Expander vertex: a connected component in G containing the terminal.

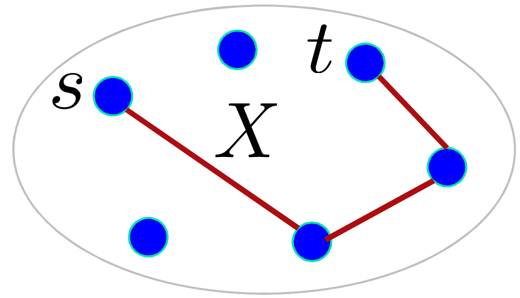
Expander edge: a path in G connecting some pair of vertices in the two components.

An edge of G belongs only to O(1) components/paths.

Degree of each expander vertex is $\Theta(\log^2 k)$.

Routing on the Embedded Expander





Routing on vertex-disjoint paths in the expander corresponds to a constant congestion routing in G !

Two Challenges

- How does one find a good family of sets?
- How do you use a good family to find the routing trees?

[Chuzhoy '12] tackles both challenges.

We will primarily focus on the second task.

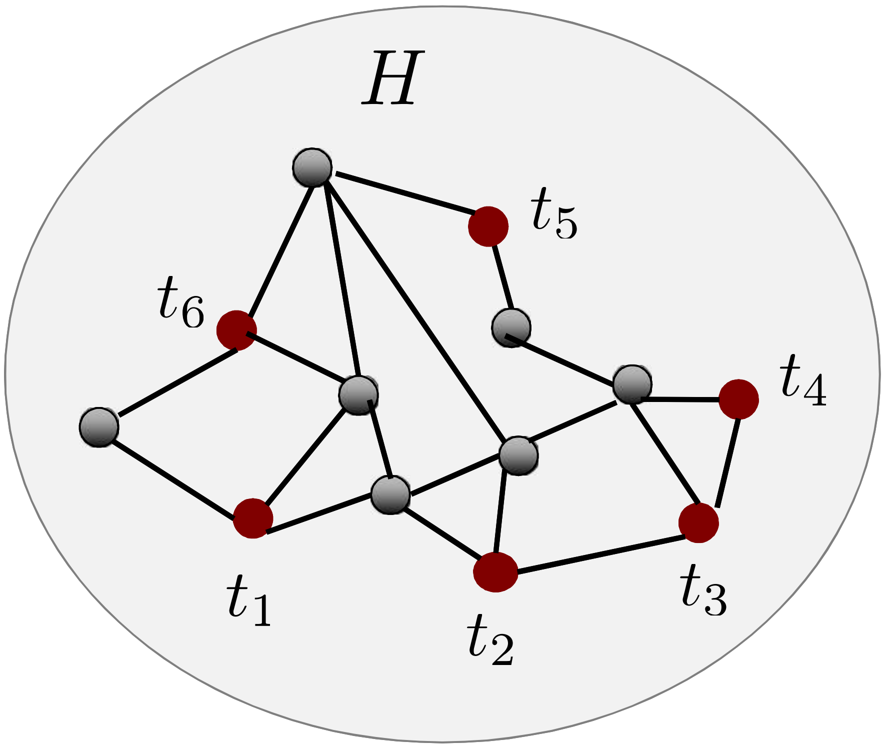
Tools

- Mader's Theorem.
- Toughness and bounded degree spanning trees.

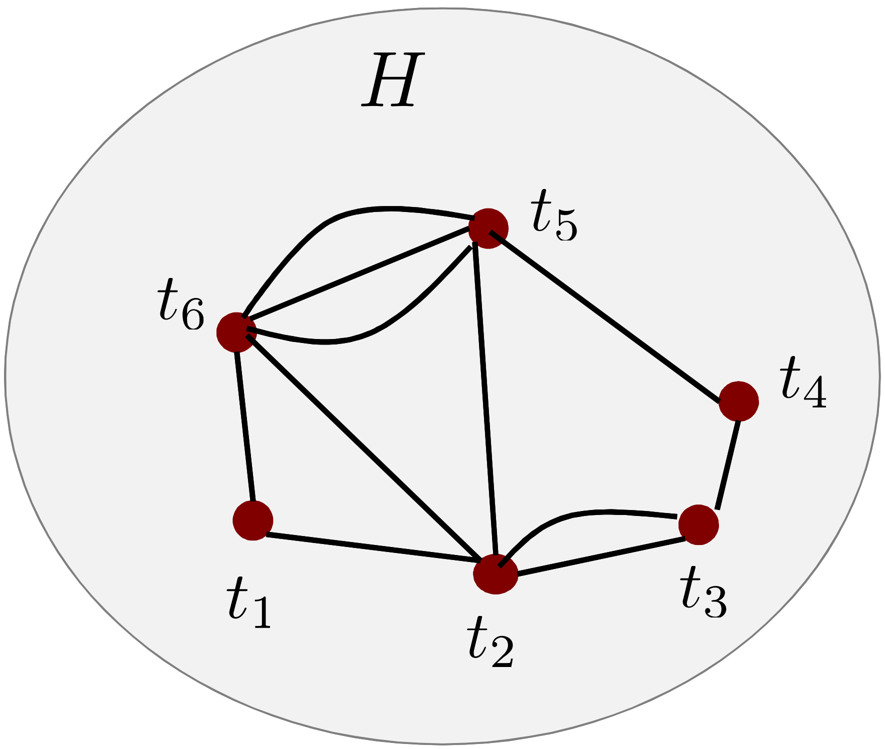
Mader's Theorem

Mader's Theorem: Given any undirected graph G and a vertex v of degree not equal to 3 such that there is no cut-edge incident on v, there always exists a splittable pair of edges incident on v.

Starting from an Eulerian graph, we can repeatedly apply Mader's theorem to preserve connectivity between a special set of vertices while eliminating edges incident on other vertices.



Splitting off to preserve pairwise edge connectivities between the t_i vertices.



- Every edge in new graph is a path in the old graph.
- These paths are edge-disjoint.
- Degree of each t_i vertex remains unchanged.
- Edge-connectivity between the t_i vertices is preserved.

Toughness and Bounded Degree Spanning Trees

The toughness $\tau(G)$ of a connected undirected graph G is defined as the ratio

$$\tau(G) = \min_S |S|/c(S)$$

where the minimum is taken over $c(S) > 1$.

- Toughness of a star is $1/(n-1)$.
- Toughness of a cycle is 1.

Theorem [Furer and Raghavachari '94]

In any connected graph G, one can find in poly-time a spanning tree T such that the maximum degree in T is bounded by $1/\tau(G) + 3$.

Routing Trees for Terminals

Starting Point: Good Family of Sets

A collection of $h = \Theta(\log^2 k)$ vertex-disjoint subgraphs $S_1, S_2, \ldots, S_h$ such that

- $out(S_i)$ is well-linked in $G[S_i]$ and has size $k_1 = k/polylog(k)$,
- $out(S_i)$ can send $k_1 = k/polylog(k)$ units of flow without congestion to a fixed set X' of k_1 terminals.

Goal: Routing Trees

Find $k/polylog(k)$ trees in G, say, $T_1, T_2, \ldots$ such that

- each tree T_i is rooted at a distinct terminal,
- each tree T_i connects to a distinct edge on the boundary $out(S_j)$ of each S_j, and
- no edge in the graph is used by more than $O(1)$ trees.

Routing Trees for Terminals

Step One (The graph H_1)

- Add new vertices $s_1, s_2, \ldots, s_h$ to G.
- Connect vertex s_i to the boundary of S_i.
- Double all edges so that we have an Eulerian graph.
- $\lambda(s_i, s_j)$ = edge connectivity between s_i and $s_j = 2k_1$.

Routing Trees for Terminals

Step Two (The graph H_2)

- Apply Mader's theorem to split off all edges incident on the original vertices in G.

- Theorem applies since we have an Eulerian graph.

- We end up with a new multigraph H_2 with only vertices $s_1, s_2, \ldots, s_h$ such that $\lambda(s_i, s_j) = 2k_1$.

- Edges in H_2 correspond to edge-disjoint paths in G.

Routing Trees for Terminals

Step Three (The graph H_3)

- Degree of each vertex in H_2 is $2k_1$.

- Discard from H_2 any edges with multiplicity less than $k_2 = k_1/h^2$ to get a new multigraph H_3.

- Thus any pair of adjacent vertices in H_3 has at least k_1/h^2 parallel edges which correspond to k_1/h^2 edge-disjoint paths in G.

The Graph H_3

Claim: There is a spanning tree T of degree at most 5 in the graph H_3.

- Suffices to show that toughness of H_3 is at least $\frac{1}{2}$.
- Suppose deleting a set Z of vertices creates p connected components, say, $C_1, C_2, \ldots, C_p$ in H_3.
- Each C_i has at least $2k_1$ edges leaving it in H_2.
- At most $h^2 (k_1/h^2) = k_1$ edges are discarded overall in going from H_2 to H_3.

At least pk_1 edges must be leaving $C_1, C_2, \ldots, C_p$ in H_3.

The Graph H_3

- On the other hand, total number of edges entering Z is bounded by $2k_1|Z|$ since degree of any vertex in H_3 is at most $2k_1$.

- It follows that $pk_1 \leq 2k_1|Z|$, and hence $|Z| \geq p/2$.

So $\pi(H_3) \geq \frac{1}{2}$.

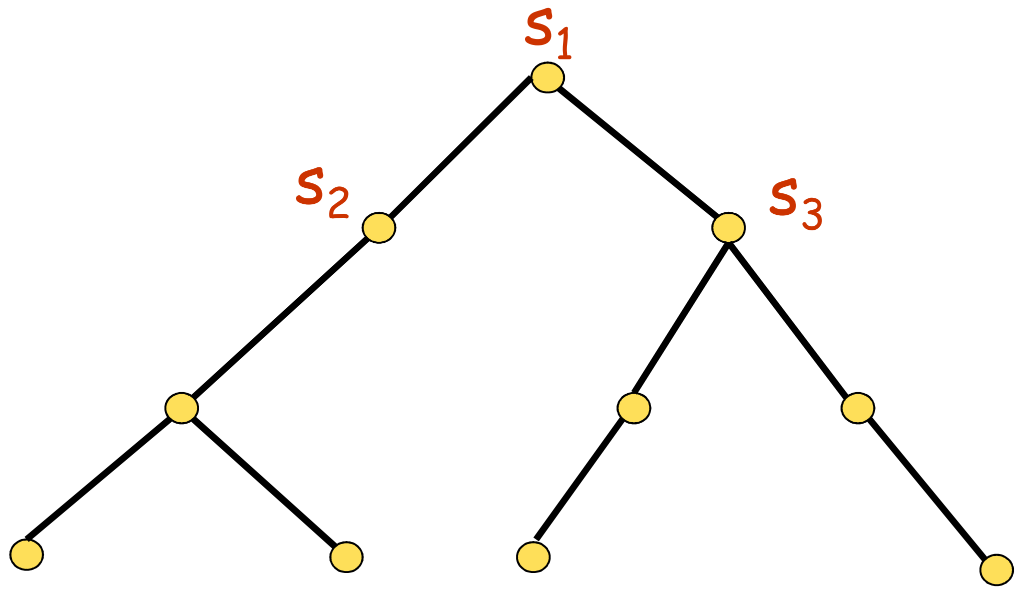
By [Furer and Raghavachari '94] theorem, H_3 has a spanning tree with maximum degree $3 + 1/\pi(H_3) = 5$.

Constructing the Routing Trees

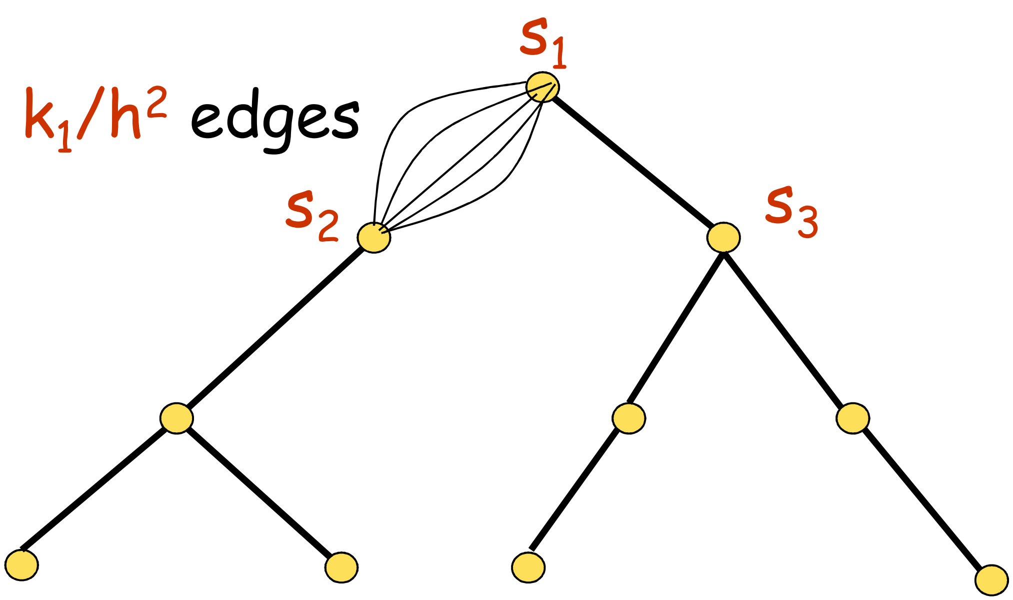
Final Step (Construct the Routing Trees)

- Fix any spanning tree T of degree at most 5 in H_3.
- Each edge of T corresponds to $k_2 = k_1/h^2$ parallel edges (which in turn correspond to edge-disjoint paths in G).
- Arbitrarily root the tree T and replace each vertex s_i by the good set S_i.

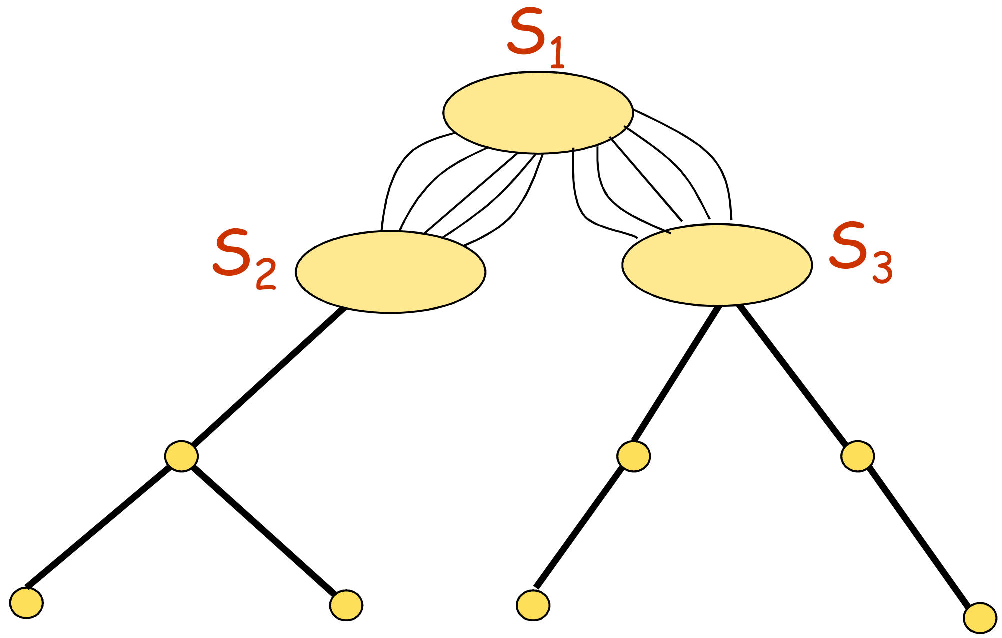
Low Degree Spanning Tree



Low Degree Spanning Tree Expanded



Low Degree Spanning Tree Expanded



Recovering the Routing Trees

- Using the fact that each S_i is well-linked w.r.t. its boundary, we can now recover T_1, T_2, …, T_{k_2} such that
  - each T_i is rooted at a distinct terminal, and
  - no edge in the graph is used by more than $O(1)$ trees.
- Recovery creates congestion = Max degree in T.
- This is where the bounded degree assumption helps!

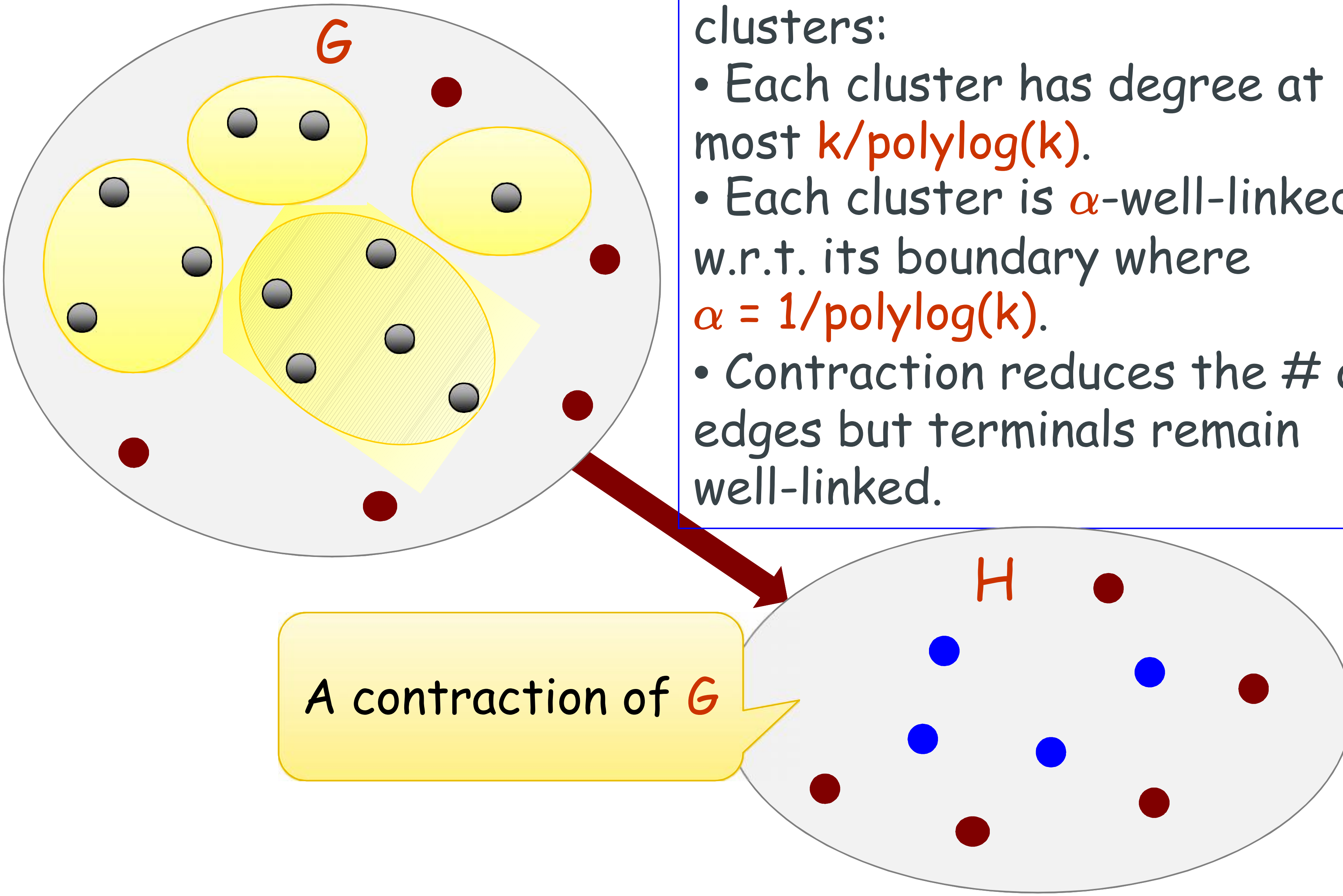
Finding a Good Family of Sets

Legal Contracted Graph (LCG)

- Let r = k/polylog(k).
- For any set S of vertices, $G[S]$ – subgraph of G induced by the set S.
- A graph H is an LCG of G if
  - H is obtained by contracting a disjoint subset of vertices that do not contain terminals.
  - Degree of each vertex in H is at most r.
  - For any vertex v where v possibly represents a contracted set S of vertices, the graph $G[S]$ is α-well-linked w.r.t. out(S) in G for α = 1/polylog(k).

Partition of non-terminals into clusters:

- Each cluster has degree at most k/polylog(k).
- Each cluster is α-well-linked w.r.t. its boundary where α = 1/polylog(k).
- Contraction reduces the # of edges but terminals remain well-linked.



Properties of LCG

- The initial graph G is an LCG of itself.
- Terminals remain well-linked in any LCG H of
  - Any cut in the LCG H maps to a cut of the same value in G.
- Since maximum degree r in an LCG H is much smaller than k, there must be $\Omega(k)$ edges in H that are incident only on non-terminals.
- The last two properties will play a crucial role.

The Algorithm

Let m = # of edges between non-terminals.

- Start by randomly partitioning all non-terminals into h sets, say, $X_1, X_2, \ldots, X_h$.

- With constant probability, each X_i satisfies:
  - $|Out(X_i)| \leq 10m/h$.
  - $|E(X_i)| \geq m/10h^2$.

- Note that $|Out(X_i)|$ and $|E(X_i)|$ are separated only by a factor of $h = \Theta(\log^2 k)$.

The Algorithm

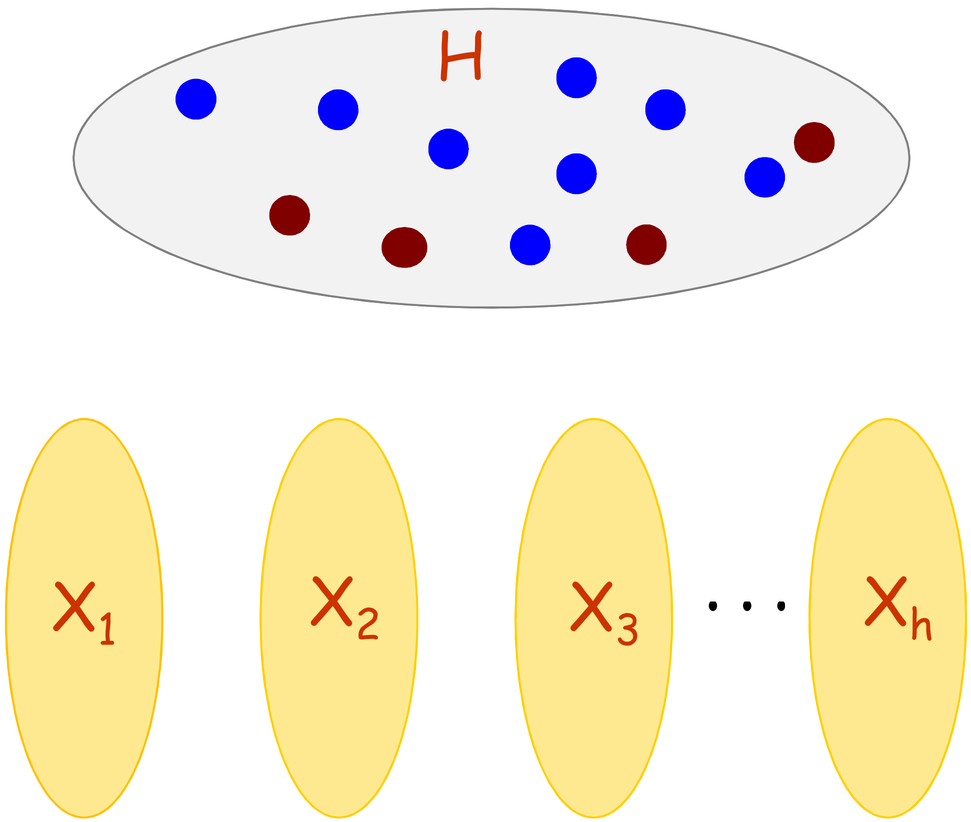
Consider a set X_i.

- Uncontract all vertices inside X_i.
- If $G[X_i]$ is α-well-linked w.r.t. $Out(X_i)$, then X_i is a good set.
- If not then do a α-well-linked decomposition inside X_i.
  - If the decomposition creates a α-well-linked piece with boundary of size at least r, this is a good set.
  - Otherwise, the process fails.
  - But total # of edges cut in the well-linked decomposition process is bounded by $\alpha|Out(X_i)|(\log^2 k) < |E(X_i)|$ -- a reduction in the size of the LCG if we contract new pieces.

The Algorithm

- If each of $X_1, X_2, \ldots, X_h$ succeeds, we get a good family of sets.
- Otherwise, some X_i fails and we get a new LCG that has fewer edges than before.
- We repeat this process until we succeed.

Random Partitioning

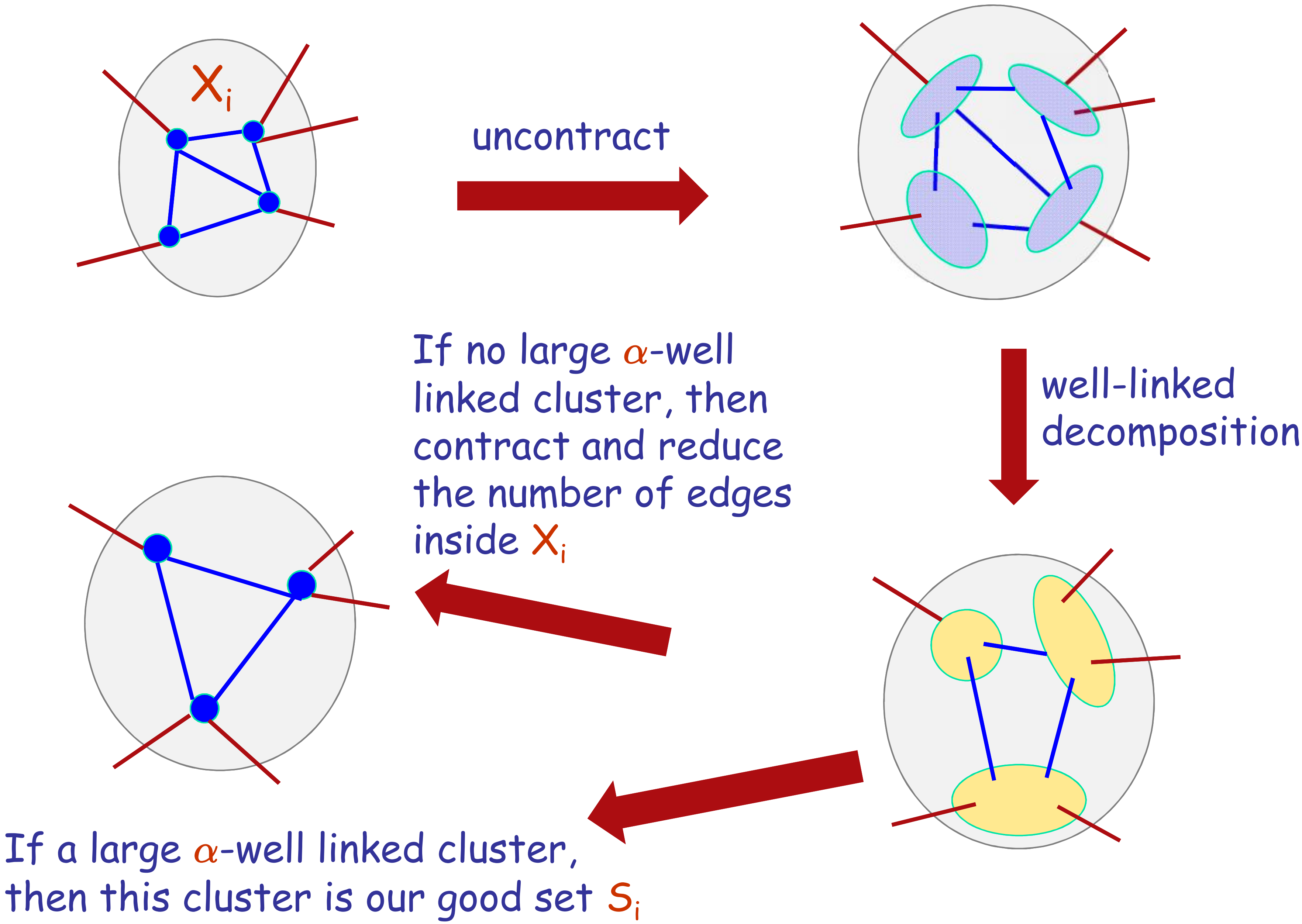


Randomly assign each non-terminal to one of the $h = \Theta(\log^2 k)$ clusters.

With constant probability, for each i

- $|out(X_i)| \leq 10m/h$
- $|E(X_i)| \geq m/10h^2$

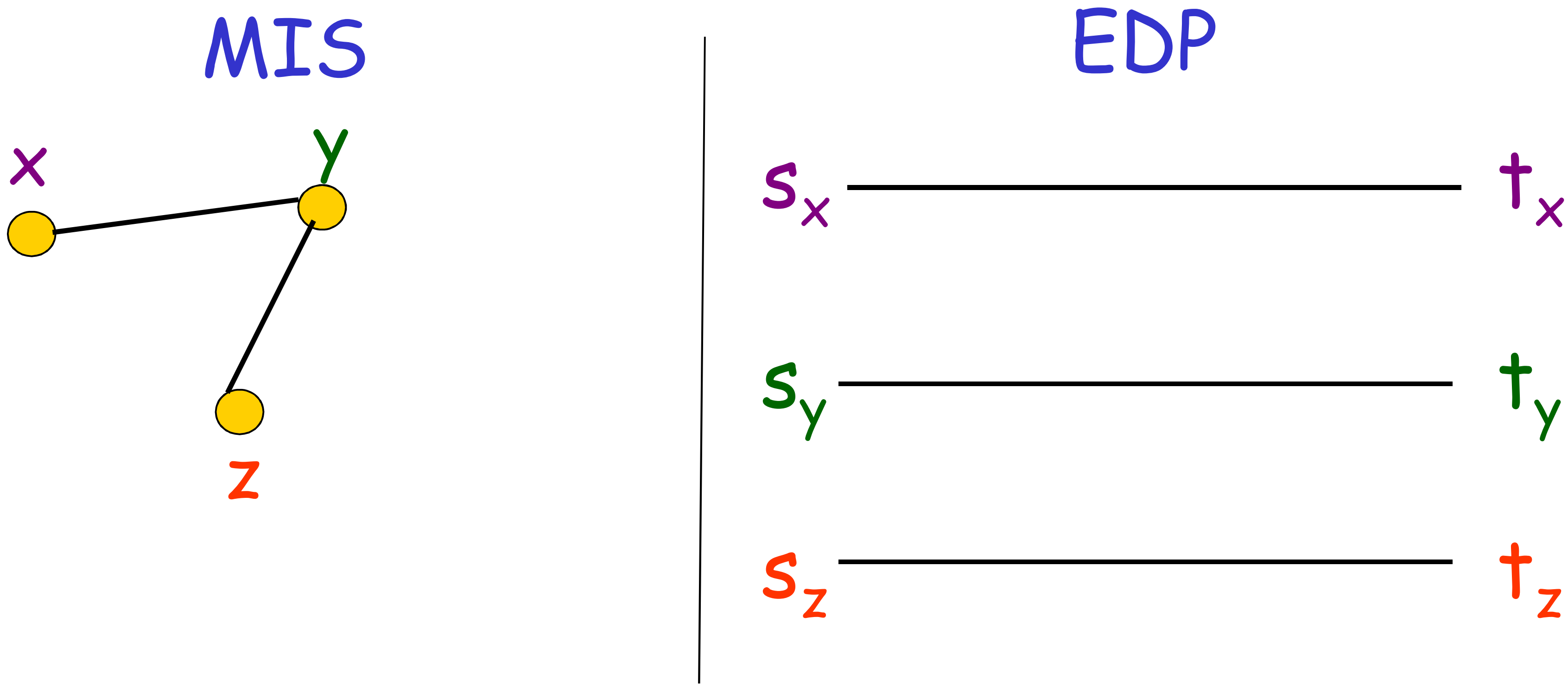
m = # of edges between non-terminals



X_i
uncontract
well-linked decomposition
If no large α-well linked cluster, then contract and reduce the number of edges inside X_i
If a large α-well linked cluster, then this cluster is our good set S_i

EDP Hardness Results

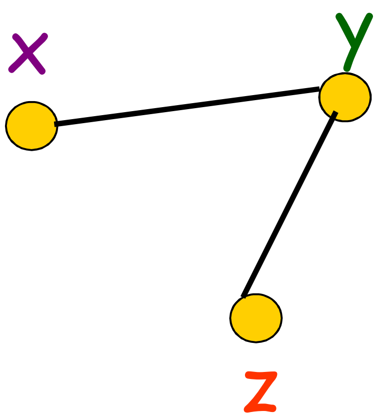
Max Independent Set (MIS) to EDP



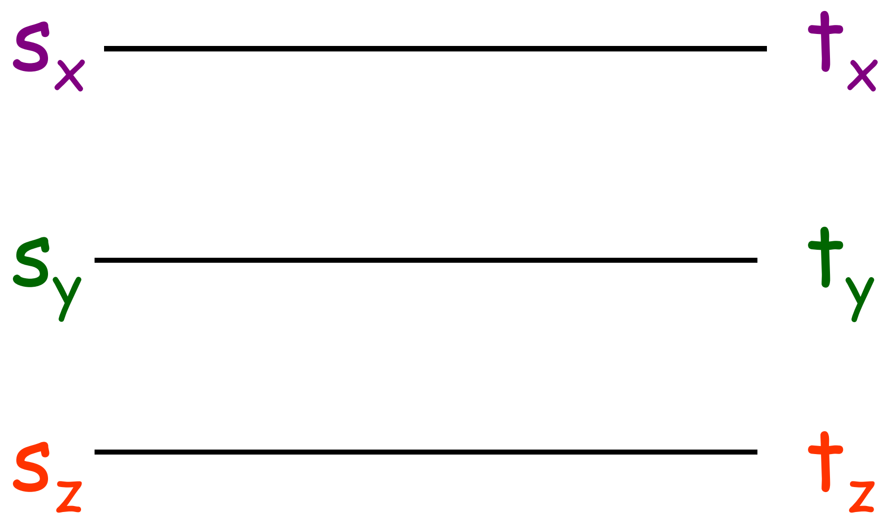
For each vertex v in the MIS instance, there is an s_v-t_v pair and a canonical path connecting s_v to t_v.

MIS to EDP

MIS

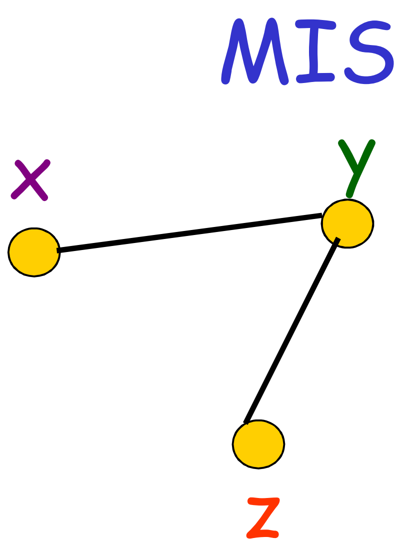


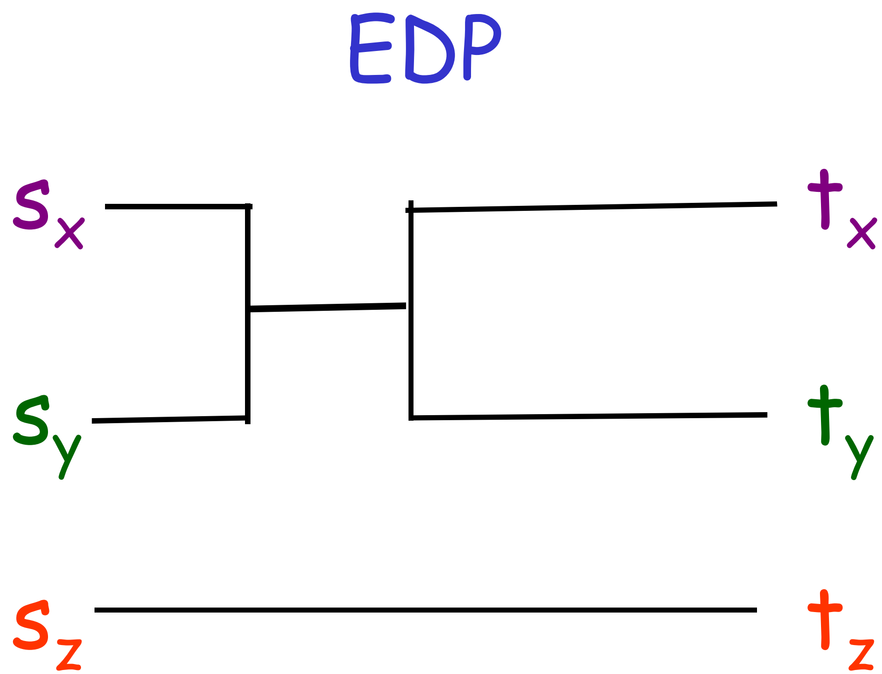
EDP



Edge between two vertices in the MIS instance ↔ Canonical paths share an edge in the EDP instance.

MIS to EDP

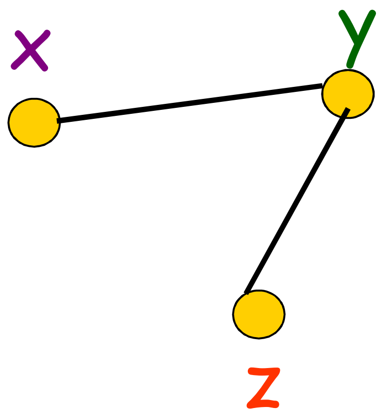




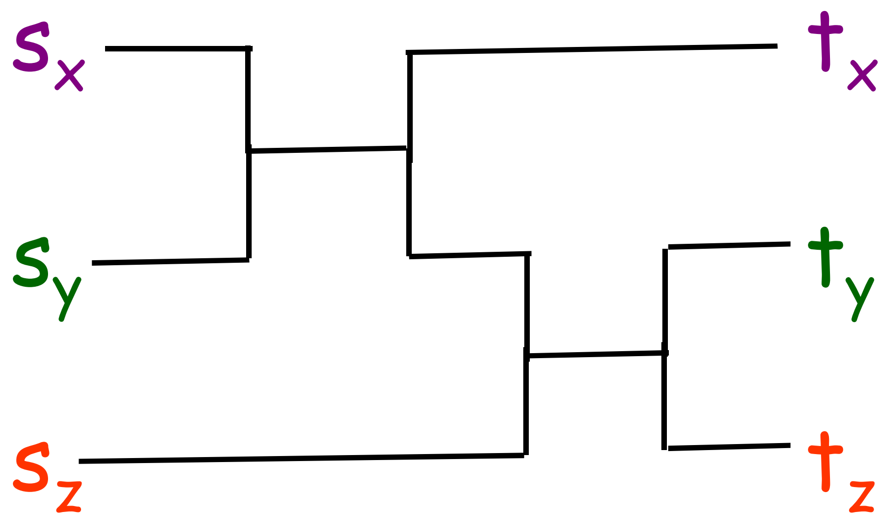
Edge between two vertices in the MIS instance ↔ Canonical paths share an edge in the EDP instance.

MIS to EDP

MIS



EDP



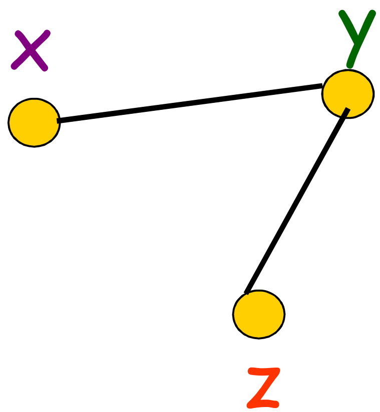
Edge between two vertices in the MIS instance $\leftrightarrow$
Canonical paths share an edge in the EDP instance.

Seems Promising …

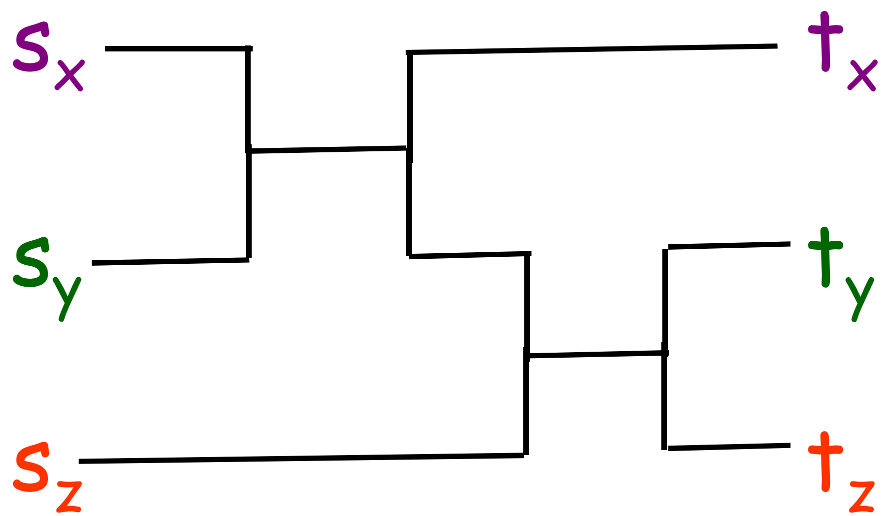
- If we could enforce that every routed pair only uses its canonical path, we would get $n^{\Omega(1)}$-hardness.
- But the path intersections create cheating (non-canonical) paths.

MIS to EDP

MIS



EDP



Edge between two vertices in the MIS instance $\leftrightarrow$ Canonical paths share an edge in the EDP instance.

Seems Promising …

- If we could enforce that every routed pair only uses its canonical path, we would get $n^{\Omega(1)}$-hardness.
- But path intersections create cheating (non-canonical) paths.
- How do we deal with them?

Directed Graphs

- Efficient labeling schemes to encode intersections of canonical paths that eliminate all non-canonical paths.
  - Once you leave the canonical path, you can not return to the original path.
  - So each pair is connected only by a canonical path.

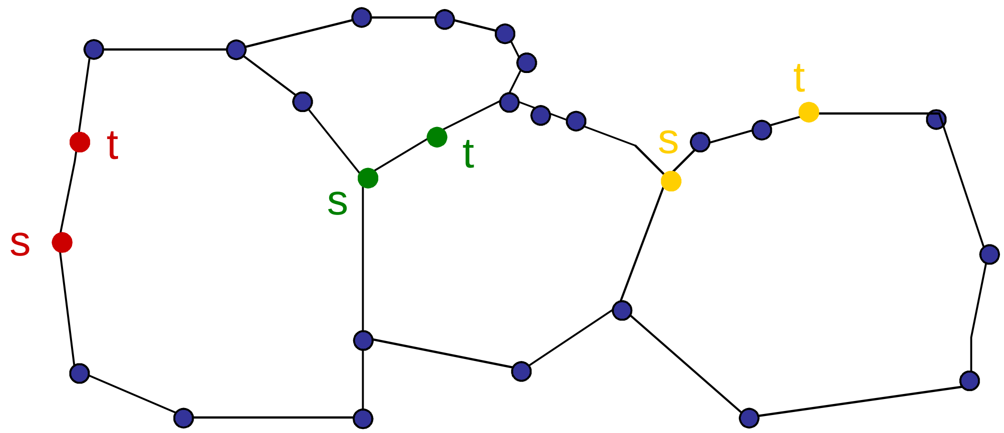
- Allows us to essentially carry independent set hardness to directed EDP even with congestion.
  - $n^{\Omega(1/c)}$-hardness for directed EDP with congestion c. [Andrews, Zhang '06] [Chuzhoy, Guruswami, K, Talwar '07]

Undirected Graphs

- No efficient labeling schemes known, and instead we rely on girth arguments.
- Girth of a graph = length of the shortest cycle.
- Canonical path + a non-canonical path = a cycle.
- So if girth is large and the canonical path is short, it follows that any cheating path must be large.

Undirected Graphs

- Each source-sink pair has a short canonical path.
- Path intersections are implemented using a "random process" to get a high girth graph: $\Theta(\log n)$ girth.



Pairs routed on non-canonical paths consume too much routing capacity.

Hardness of Undirected EDP

Simplified Analysis
(ignores implementation of girth property)

- Start with a degree d-bounded independent set instance where $d = \log^{1/2} n$.
- Hard to decide if max independent set size is $\Omega(n/d^{\epsilon})$ (Yes case) or $O(n/d^{1-\epsilon})$ (No case) for any $\epsilon > 0$.
- Create an $\Omega(\log n)$ girth undirected EDP instance:
  - Canonical paths have length $d = \log^{1/2} n$.
  - Non canonical paths have length $\Omega(\log n)$.
  - $O(nd)$ edges in total.

Hardness of Undirected EDP

Yes Case

- We can route $\Omega(n/d^{\epsilon})$ pairs in an edge-disjoint manner using canonical paths.

No Case

- Only $O(n/d^{1-\epsilon})$ pairs can be routed on canonical paths.

- Only $O(nd/\log n)$ pairs can be routed on non-canonical paths since girth is $\Omega(\log n)$.

Hardness of Undirected EDP

Yes Case

- $\Omega(n/\log^{\epsilon} n)$ pairs can be routed.

No Case

- $O(n/d^{1-\epsilon}) + O(nd/\log n) = O(n/\log^{1/2} n)$ pairs can be routed when $d = \log^{1/2} n$.

So we get a $\Omega(\log^{1/2-\epsilon} n)$ hardness for undirected EDP with no congestion.

So what remains to be done …

Approximability of undirected EDP with no congestion.

On the positive side …

$O(n^{1/2})$-approximation [Chekuri, K, Shepherd '06]

- Algorithm is based on rounding the multicommodity flow relaxation.
- Upper bound matches the integrality gap of the flow relaxation.

So what remains to be done …

On the negative side …

$\Omega(\log^{1/2-\epsilon} n)$ hardness [Andrews, Chuzhoy, Guruswami, K, Talwar, Zhang '05]

Approximability of undirected EDP remains wide open!

Undirected Congestion Minimization

A related open problem is congestion minimization in undirected graphs: minimize congestion needed to route all pairs.

- Randomized rounding of LP gives an $O(\log n/\log \log n)$ approximation [Raghavan and Thompson '87].
- A matching hardness result known in directed graphs. [Andrews, Zhang '06] [Chuzhoy, Guruswami, K, Talwar '07]
- But in undirected graphs, best known hardness is $\Omega(\log \log n / \log \log \log n)$ [Andrews and Zhang '07]

Concluding Remarks

- Several beautiful ideas composed together to obtain a constant congestion polylog-approximation for EDP.
- These ideas have already been used to obtain many other important results.
- With constant congestion, it is also possible to get a polylog-approximation for vertex-disjoint paths [Chekuri, Ene '13].

Thank You!