Report on paper:

Vertex Connectivity in Poly-logarithmic Max-flows

Presentation outline is in Appendix

Hansheng Liu (hl58)

December 16, 2024

1 Introduction

The **vertex connectivity** of a graph G is the minimum k s.t. for some $S \subseteq V(G)$ with $|S| = k$, graph $G - S$ either is disconnected or has at most one vertex. Connectivity is one of the most important and well-studies problems in graph theory. It can also be applied in many fields such as Computer Science and Tranportation. However, the algorithm to compute the vertex connectivity for a given graph is not an easy problem. For a long time, the best algorithm people have requires a running time of $\tilde{O}(mn)$ [3]. In 2021, Saranurak et al. proposed a new algorithm that reduced the vertec connectivity problem to polylog maxflow instances [4]. In 2022, Chen et al. gave an algorithm that compute the maxflow in $m^{1+O(1)}$ time [2], which can be used as a black box and make the vertex connectivity be computed in $\tilde{O}(m)$. In this report, we will focus on the idea of how to reduce vertex connectivity problems into polylog maxflow instances.

2 The main algorithm and its analysis

2.1 Basic settings and assumptions

Let $|V(G)| = n$, $|E(G)| = m$. Let S where $|S| = k$ denote the smallest cut. There will be at least two connected components (if G is not K_n) in $G - S$. Let the smallest one be L, others be R. So $|L| \leq |R|$. Let $V_{low} = \{v | d(v) \leq 8k\}$. $L_{low} = L \cap V_{low}$, and same for S_{low}, R_{low}.

Assumption 1: We already know $k, |L|, |S_{low}|$. We will prove later that we don't need exact true value to make the algorithm work. Actually, we just need the value to be inside a range between the exact true value, which can be done by binary search.

Assumption 2: $|R| = \Omega(n)$. This means G does not have a minimum vertex cut that covers most of the verticies. Examples include complete graph, or some highly connected graph.

Assumption 3: $m \leq nk$. This clearly not applies to many graphs. However, we can achieve this by computing a *k-connectivity certificate* [5] for G, which results in this.

We divide the algorithm into three cases depends on L and S_{low}.

2.2 Case 1: Large L

In this case, we assume $|L| > \frac{k}{polylog(n)}$. We independently sample each vertecx with probability $\frac{1}{k}$, let the sampled set be T.

Claim 1: With some probability, T contains exactly one vertex from L, no vertex from S, and all others (at least one) from R.

Proof:

$$\mathbb{P}[|L \cap T| = 1] \cdot \mathbb{P}[|S \cap T| = 0] \cdot \mathbb{P}[|R \cap T| \geq 1]$$

$$\geq c \cdot \mathbb{P}[|L \cap T| = 1] \cdot \mathbb{P}[|S \cap T| = 0]$$

$$\geq c \cdot (\frac{|L|}{k} \cdot (1 - \frac{1}{k})^{|L|-1}) \cdot (1 - \frac{1}{k})^{|S|}$$

$$\geq c \cdot \frac{|L|}{k} \cdot (1 - \frac{1}{k})^{|L|+|S|-1}$$

$$\geq \Omega(\frac{1}{\log n})$$

We can repeat polylog(n) times to increase probability.

After we obtained T, we can use modified *Isolating Cuts* [1] to compute the final results. We will skip the detail of this part as it will distract our main focus.

2.3 Case 2: Small L with small S_{low}

In this case, we assume $|L| < \frac{k}{polylog(n)}$ and $S_{low} < |L| \cdot polylog(n)$.

Claim 2: $|L \cup S| \leq 2k$

Proof: $|L| < \frac{k}{polylog(n)}$, $|S| = k$.

Claim 3: $L = L_{low} \subseteq V_{low}$

Proof: For every vertex x in L, all its neighbours must in $L \cup S$. So $d(x) \leq L + k \leq 2k < 8k$.

Claim 4: $|R_{low}| \geq |L_{low}|$

Proof: If $k \geq \frac{n}{8}$, then all verticies are in V_{low}, the claim is trivial. Therefore, assume $k < \frac{n}{8}$. Notice that

$$8k \cdot |V - V_{low}| \leq \sum_v d(v) = 2m \leq 2nk$$

Therefore, we have $|V - V_{low}| \leq \frac{n}{4}$, so $|V_{low}| \geq \frac{3}{4}n$. Also, by claim 2, we have $|R| \geq \frac{3}{4}n$. Therefore, $|R_{low}| \geq (1 - (1 - \frac{3}{4}) - (1 - \frac{3}{4}))n \geq \frac{n}{2} \geq |L_{low}|$.

We can independently sample vertex from V_{low} with probability $\frac{1}{|L| \cdot polylog(n)}$ and get in to the same situation as in case 1. The detailed analysis is similar.

2.4 Case 3: Small L with large S_{low}

In this case, we assume $|L| < \frac{k}{polylog(n)}$ and $S_{low} \geq |L| \cdot polylog(n)$. In this case, the sampling method we used in previous cases will not work as there is not a way to sample a desired vertex set with high probability. Therefore, we need to do something different.

A quick question: What information do we need to compute the vertex connectivity?

Answer: A vertex from L and at least one vertex from R. Then, the vertex connectivity can easily be computed by a maxflow.

Notice that if we do random sample from V, have verticies from R is with very high probability, but have a vertex from L is with very low probability. Actually, you need to sample at least $\tilde{O}(\frac{n}{|L|})$ to make

sure there is at least one vertex from L with high probability, call this set X. By constant number of sampling, we can assume we have at least one vertex from R, call it r.

Silly algorithm: For each vertex x in X, compute the maxflow from x to r. The minimum cut of the maxflow is the vertex connectivity.

This algorithm works, but it is too slow. The runtime is $\tilde{O}(\frac{n}{|L|} \cdot m)$. We need to improve it.

Improvement idea: If we are able to reduce the number of edges during each maxflow computation to $\tilde{O}(k|L|)$, then we can reduce the total runtime to $\tilde{O}(\frac{n}{|L|} \cdot k|L|) = \tilde{O}(nk) = \tilde{O}(m)$. (This m is the number of edges before we compute the k-connectivity certificate)

Lemma 1: For each vertex x in X, we can compute a new graph which keeps the vertex connectivity and has at most $\tilde{O}(k|L|)$ edges. We call this graph a *kernel*.

We now elaborate on how to construct the kernel. Assume we already have vertex $x \in L$, which is from X. Independently sample each vertex from V with probability $\frac{1}{|L|}$ into set T. Let $N[x] = N(x) \cup \{x\}$ and $T_x = T - N[x]$.

Claim 5: $|(L \cup S) - N[x]| < |L|$

Proof: Notice that $|L \cup S| \leq |L| + k$. Also, $|N[x]| = |N(x)| + 1 \geq k + 1 > k$. Therefore, $|(L \cup S) - N[x]| < |L| + k - k = |L|$.

If no verticies we sampled are from $|(L \cup S) - N[x]|$, then $T_x \subseteq R$. By claim 5, if we sample with probability $\frac{1}{|L|}$, we will have $T_x \subseteq R$ with high probability.

After this, we contract T_x into a single vertex t_x. Notice that if $x \in L$ and $T_x \subseteq R$, then a (x, t_x) maxflow will give us the vertex connectivity of G.

Observations:

1. $\forall\ v \in N(x) \cap N(t_x)$, it must be in S.
2. By Mengers theorem, there exists k vertex disjoint paths between x and t_x. Notice that if any path include more than one neighbour of x, then the path can be changed to directly connect to x without affect other paths. Same for t_x. Therefore, we can have each path only contains one neighbour of x and one neighbour of t_x.

By these observations, we can do following removals [1]:

1. Remove all $v \in N(x) \cap N(t_x)$ and add back later.
2. Remove all internal edges of $N(x)$ and $N(t_x)$.
3. After first two, remove all $v \in N(T_x)$ s.t. t_x is the only neighbour of v. This happens if $t \in N(v) \subseteq N[t]$.

[1] If reader find the following removal procedures hard to understand, there is a graph illustration in the appendix.

Claim 6: After these removals, the graph has $\tilde{O}(k|L|)$ edges, which is the kernel we want.

Proof: Let N_x be the remaining neighbours of x, N_t be the remaining neighbours of t_x, F' be all verticies left. There are three kinds of edges remain:

$E1 : N_x \times (F' \cup N_t)$

$E2 : F' \times (F' \cup N_t)$

$E3$: incident with x and t_x

We first prove $|E1 \cup E2| = \tilde{O}(k|L|)$ whp. We prove by showing the endpoints and their degree are bounded.

Claim 7: $\forall\ v \in N_x \cup F', d_{G-N_x}(v) \leq \tilde{O}(|L|)$ whp.

Proof: We prove a stronger claim $\forall\ v \in N[x] \cup F', d_{G-N[x]}(v) \leq 8k$ whp. Assume not, then $d_{G-N[x]}(v) > |L| \cdot polylog(n)$, then at least one of v's neighbour will be sampled into T_x whp, which makes $v \in N(T_x) = N_t$, thus $v \notin N_x \cup F'$. Contradiction.

Claim 8: $|N_x \cup F'| = O(k)$

Proof: Note that $|N_x| \leq |L| + k < 2k$, so we just need to show $|F'| = O(k)$. We go back to the origin graph G.

Claim 8.1: $|N(x)| < k + |L|$

Proof: $|N(x)| \leq |L| \cup |S|$.

Claim 8.2: $|S_{low} - N(x)| \leq |L|$

Proof: direct result of claim 5.

Claim 8.3: $\forall\ v \in F'$, whp, at least $k - |L| \cdot polylog(n)$ neighbours of v are in $N(x)$.

Proof: derect result of claim 7.

By claim 8.1, 8.2, 8.3, we can compute the number of neighbours of $v \in F'$ in S_{low}, which is

$$\geq \text{Total number of neighbours in } N[x] \text{ - verticies in } (N[x] - S_{low})^2$$

$$\geq k - |L| \cdot polylog(n) - (|N[x] - (|S_{low})| - |L|)$$

$$\geq \Omega(|S_{low}|)$$

We know that total numner of edges indident to S_{low} is $O(k|S_{low}|)$, each vertex in F' will take at least $|S_{low}|$ edges, so

$$|F'| = O(\frac{k \cdot |S_{low}|}{|S_{low}|}) = O(k)$$

This proves claim 8.

By claim 7 and 8, we have $|E1 \cup E2| = O(k|L|)$.

[2]If reader find this hard to follow, there is a graph illutration in appendix.

For $E3$, the numner of edges incident to x is bounded by $N[x] \leq 2k$. For edges indident to t_x, it must be incident to some edge in $E1 \cup E2$, or it would be removed in removal three. So it is also bounded by $O(k|L|)$. We proved lemma 1.

2.5 Algorithm aspect of Case 3

Building kernel can be done in sublinear time. We won't cover it in this report as it requires some more technical details in data structures and linear algebra.

Let's now go back to reality, where we don't have assumption 1. It turns out that we can we don't need the exact value of k and $|L|$ for the kernel to be constructed successfully. If we have κ and ℓ, we only need to make sure $\kappa \geq k$ and $\frac{|L|}{2} \leq \ell \leq |L|$, then it will work. Otherwise, it will either return no cut or a cut that is too large. This results in the algorithm for case 3 assume we know κ:

Algorithm 1 Case 3

```
Let X be the set of sampled verticies
for each x ∈ X do
    for i in 1 to log(n) do
        Let ℓ = 2^i
        Sample T with ℓ, κ
        Compute kernel and run maxflow
        Repeat log(n) times
    end for
end for
Return the smallest cut found
```

3 Putting things together

Algorithm 2 Vertex connectivity

```
Let k = n/2
Compute k-connectivity certificate for G
Do case 1, 2, 3
Verify and return the smallest cut found
Binary search for k and return the smallest cut found
```

References

[1] Chekuri, C., and Quanrud, K. Isolating cuts, (bi-)submodularity, and faster algorithms for global connectivity problems. *CoRR abs/2103.12908* (2021).

[2] Chen, L., Kyng, R., Liu, Y. P., Peng, R., Gutenberg, M. P., and Sachdeva, S. Maximum flow and minimum-cost flow in almost-linear time. *arXiv:2203.00671* (2022).

[3] Henzinger, M. R., Rao, S., and Gabow, H. N. Computing vertex connectivity: New bounds from old techniques. *Journal of Algorithms 34*, 2 (2000), 222–250. Announced at FOCS'96.

[4] Li, J., Nanongkai, D., Panigrahi, D., Saranurak, T., and Yingchareonthawornchai, S. Vertex connectivity in poly-logarithmic max-flows. *arXiv:2104.00104* (2021).

[5] Nagamochi, H., and Ibaraki, T. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica 7*, 5&6 (1992), 583–596.

Vertex Connectivity in polylog Max-Flows (2021)

Present by Hansheng Liu 2024/12/10

Results: Reduce vertex conn. to polylog maxflow instances

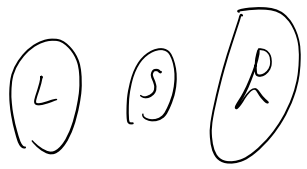
m^{α} maxflow $\rightarrow \tilde{O}(m^{\alpha})$ for vertex conn.

current best maxflow: $m^{1+o(1)}$ (2022), $m^{4/3+o(1)}$ (before published)

Previous best: $\tilde{O}(mn)$ in 1996

I will not focus on probability computation due to time

Basic Settings:



$|S| = k$ is the cut, $|L| \le |R|$, $|R| = \Omega(n)$

$V_{low} = \{v \mid d(v) \le 8k\}$

$m \le nk$ by compute a

k-connectivity certificate black box

Assume we know k, $|L|$, $|S_{low}| = S \cap V_{low}$

Case 1: Large $L \rightarrow |L| > k/\text{polylog}(n)$

Sample each vertex with Prob. $1/k$ to T

With some prob., T contains exactly 1 vertex from L, none from S, all other from R

Use adapted Isolating Cuts

(will cover if time is enough)

Case 2: Small L, small S_{low}

$|L| < k/\text{polylog}(n)$, $|S_{low}| < |L| \cdot \text{polylog}(n)$

$\forall x \in L$, $N(x) \subseteq L \cup S \to d(x) \leq |L| + |S| < 2k \in V_{low}$

$\delta k |V \setminus V_{low}| \leq \sum_v d(v) = 2m \leq 2nk$ $\boxed{k < n/\delta}$

So $|V \setminus V_{low}| \leq \frac{n}{4}$, $|V_{low}| \geq \frac{3}{4}n \Rightarrow |R \cap V_{low}| \geq \frac{n}{2} \geq |L_{low}|$

Sample from V_{low} with Prob. $1/(|L| \cdot \text{polylog}(n))$

Then same as Case 1

Case 3: Small L, large S_{low}

$|L| < k/\text{polylog}(n)$, $|S_{low}| > |L| \cdot \text{polylog}(n)$

What info we need to get a vertexcut?

- know $l \in L$, $r \in R$, easily compute mincut by maxflow

Obtain r is easy, but obtain l is hard

Require $\tilde{O}(n/L)$ samples to get high prob.

- Run maxflow on these is too slow

Method: Run maxflow on a graph with $\tilde{O}(kL)$ edges

We want to construct a graph H with $\tilde{O}(kL)$ edges which contains the info. of vertex conn. of G

$$\tilde{O}\left(\frac{n}{L} \cdot kL\right) = \tilde{O}(nk) \leq \tilde{O}(m)$$

We call this graph a kernel

Workflow: ① sample $\tilde{O}(n/L)$ vertices to make at least one of them be in L whp.

② construct kernel

③ run maxflow on kernel

④ pick the minimum one

How to construct the kernel?

Sample vertices with prob $1/|L|$ into set T

$N_G[x] = N_G(x) \cup \{x\}$

$T_x = T \setminus N_G[x]$

Claim 1: $|(L \cup S) \setminus N[x]| < |L|$

Proof: $|L \cup S| \leq |L| + k$

$|N[x]| = |N(x)| + 1 \geq k + 1 > k$

$|(L \cup S) \setminus N[x]| < |L| + k - k < |L|$

So $T_x \subseteq R$ whp as sample prob. is $1/|L|$

Contract T_x into single vertex t_x

Important: an (x, t_x) maxflow call will return the vertex conn. of G

Still too many edges.

Need to further reduces edges w\ losing info. about vertex conn.

Observations:

① $v \in N(x) \cap N(t_x)$ must be in S

② Mengers: $\exists k$ vertex disjoint path between x and t_x s.t. each path contains exactly one nb of x and one nb of t_x

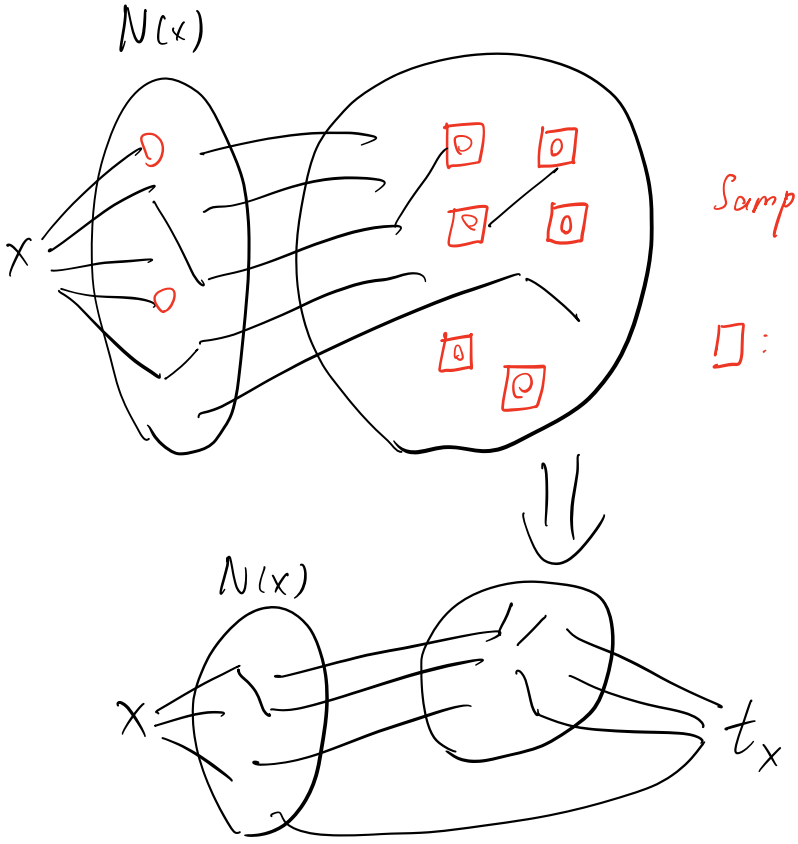
What we can do now?

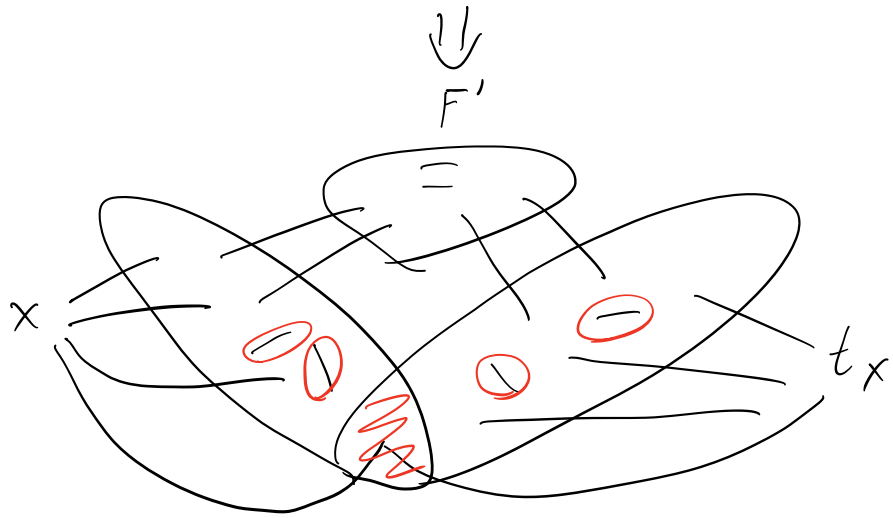
By ①, remove all $v \in N(x) \cap N(t_x)$ as must in S

By ②, remove all internal edges of $N(x)$ and $N(t_x)$

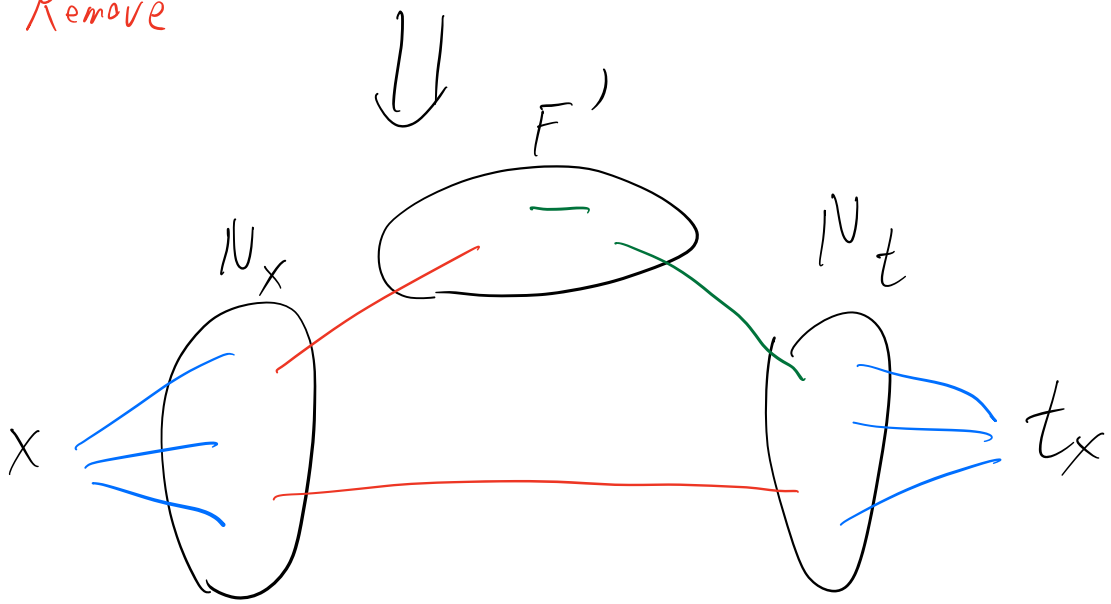
Also, $v \in N(t_x)$ which only connect to t_x
— happens when $t \in N(v) \subseteq N[t]$

Claim 2: after these removal, there are only $\tilde{O}(kL)$ edges





Remove



E_1 E_2 E_3

E_1 : $N_x \times (F' \cup N_t)$
E_2 : $F' \times (F' \cup N_t)$
E_3 : incident to x and t_x

We need $E_1, E_2, E_3 = \tilde{O}(kL)$

☆ Distinguish between
$N(x)$
$N[x]$
N_x

We first prove $|E_1 \cup E_2| = \tilde{O}(kL)$ whp. We prove by showing the endpoint's degree is bounded

Claim 3: $\forall v \in N_x \cup F'$, $d_{G \setminus N_x}(v) \leq \tilde{O}(|L|)$ whp

Proof: We prove stronger:

$\forall v \in N[x] \cup F'$, $d_{G \setminus N[x]}(v) \leq \tilde{O}(|L|)$ whp

Assume not, that is $d_{G \setminus N[x]}(v) > |L| \cdot \text{polylog}(n)$, then at least one of v's neighbour will be sampled into T_x, which makes $v \in N(t_x)$. The stronger result easily leads to our claim

Claim 4: $|V(N_x \cup F')| = O(k)$

Proof: Note that $|N_x| \leq |L| + |S| \leq 2k$, so we just need to focus on F'

From here, we go back to G (before the common neighbours of x and t_x be deleted)

Claim 4.1: $|N(x)| \leq k + |L|$

Proof: $|N(x)| \leq |L| \cup |S|$

Claim 4.2: $|S_{low} \setminus N(x)| \leq |L|$

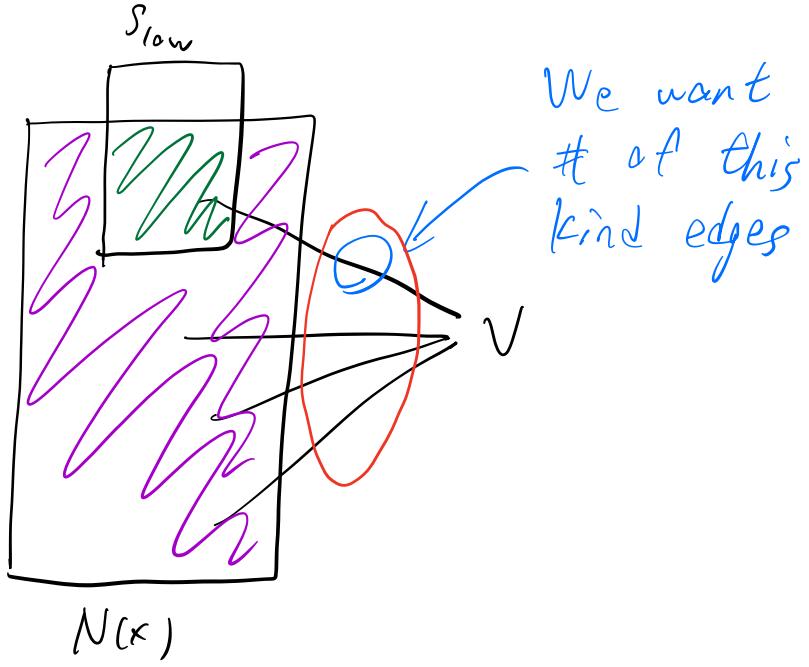
Proof: $|(L \cup S) \setminus N[x]| \leq |L|$

Claim 4.3: $\forall v \in F'$, whp, at least $k - |L| \cdot \text{polylog}(n)$ neighbours in $N(x)$

Proof: follows Claim 3

Claim 4.4: $\forall v \in F'$, $\#$ $|N(v) \subseteq S_{low}| = \Omega(|S_{low}|)$

Proof:



$$k - |L| \text{polylog}(n) - (|N(x)| - (|S_{low}| - |L|))$$

$$\geq k - |L| \text{polylog}(n) - (k + |L|) + (|S_{low}| - |L|)$$

$$\geq |S_{low}| - 2|L| - |L| \text{polylog}(n)$$

$\curvearrowleft$ dominated

We know that total # of edges incident to S_{low} is $O(k \cdot |S_{low}|)$, we have

$$|F'| = O\left(\frac{k \cdot |S_{low}|}{|S_{low}|}\right) = O(k)$$

By Claim 3 and Claim 4, we get

$$|E_1 \cup E_2| = \tilde{O}(kL)$$

For E_3, number of edges incident to x is bounded by $N[x] < 2k$. For edges incident to t_x, it must be incident to some edges in $E_1 \cup E_2$, or it will be deleted in the third step.

So $|E_3| \leq 2k + |E_1 \cup E_2| = \tilde{O}(kL)$

Claim 2 proved

Build kernel in sublinear time:

Sparse recovery sketches

(Focus on data structure, skip as we focus on graph)

Go back to reality: we don't know $|L|, k, |S_{low}|$

Let ℓ, K be our guess of $|L|, k$

Claim 5: When $K \geq k$, the kernel will output correct k whp. Else, output nothing

Claim 6: We don't need exact $\ell = |L|$. Whenever $\frac{|L|}{2} \leq \ell \leq |L|$, the kernel will output correct result whp. Else, output no cut.

Algorithm:

Binary search on k:

Compute k-conn. certificate

Case 3:

For i in 1 to $\lg(k/100 \log n)$:

$\ell = 2^i$

Sample $T^{(i,1)}, \dots, T^{(i,\log n)}$ by prob. $1/8\ell$

Sample X with prob. $n \log n / \ell$

For j in 1 to $\log n$:

Compute kernel $(\ell, X, T^{(i,j)})$

Run maxflow to get k

Among all, verify the smallest, and return

Case 1: sample and do modified isolating cut
Case 2: try different t, sample, isolating cut
Return min of three cases

Isolating Cuts (If time permit)

Let U_v be the connected component of v
Remove all edges with both endpoints in $N_G(U_v)$
Add t connect to all $N_G(U_v)$
Min s-t cut, get S_v
Vertex-cut: $N(S_v)$