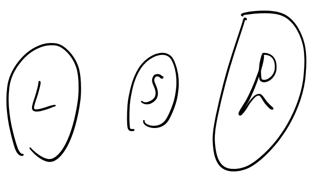
Vertex Connectivity in polylog Max-flows (2021)

Present by Hansheng Liu 2024/12/10

Results: Reduce vertex conn. to polylog maxflow instances
m^{α} maxflow $\longrightarrow$ $\tilde{O}(m^{\alpha})$ 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 probabity 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 $\boxed{\text{k-connectivity certificate}}$ 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 $\boxed{\text{adapted Isolating Cuts}}$

(Will cover if time is enough)

Case 2: Small L, small S_{low}

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

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

$8k|V \setminus V_{low}| \leq \sum_v d(v) = 2m \leq 2nk$ $\quad \boxed{k < n/8}$

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

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

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 furthur 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?

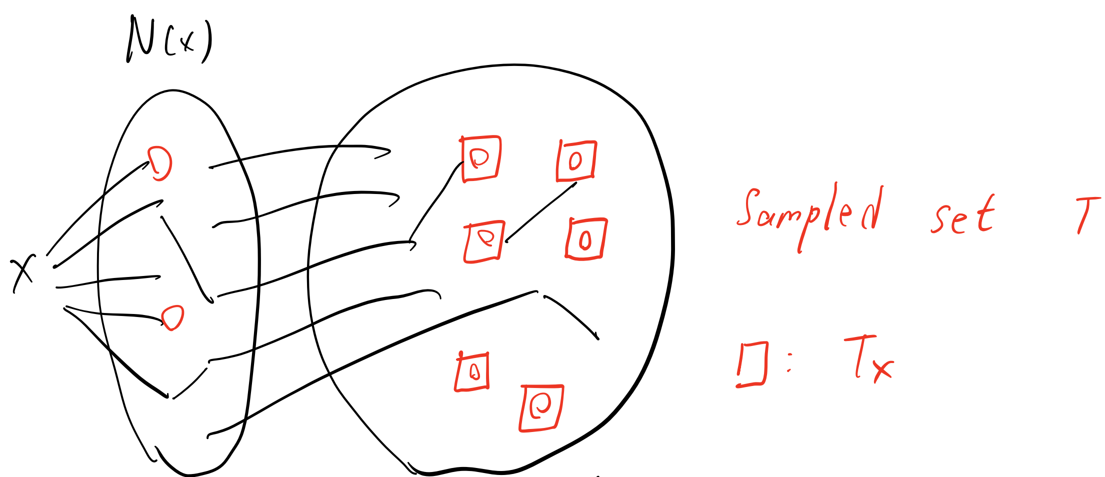
By ①, remove all $v \in N(x) \cap N(t_x)$ as must in S

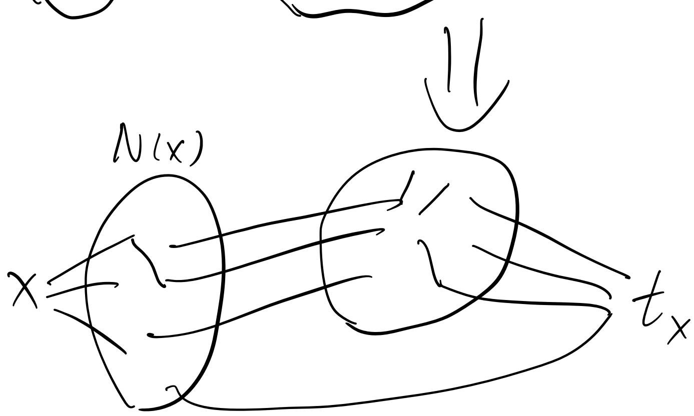
By ②, remove all internal edges of $N(x)$ and $N(t_x)$

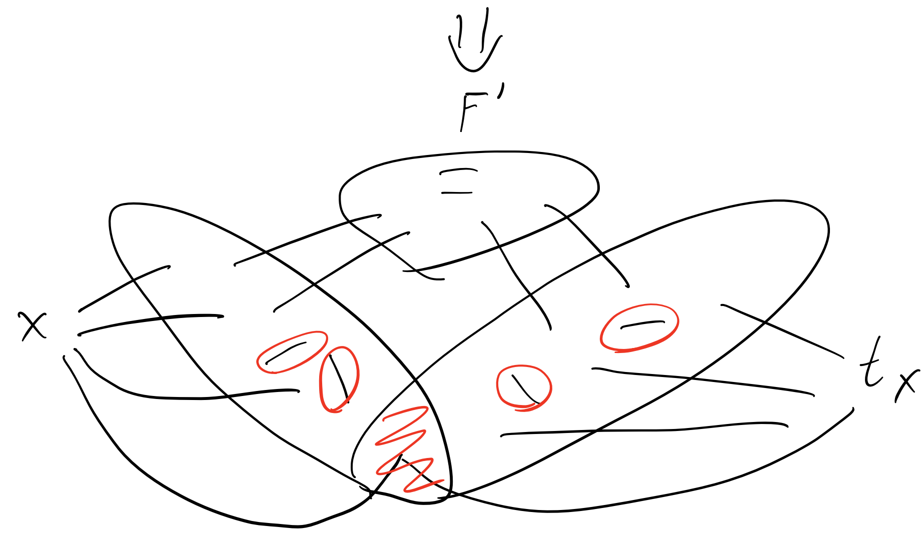
Also, $v \in N(t_x)$ which only connect to t_x

— happens when $t \in N(v) \subseteq N[t]$

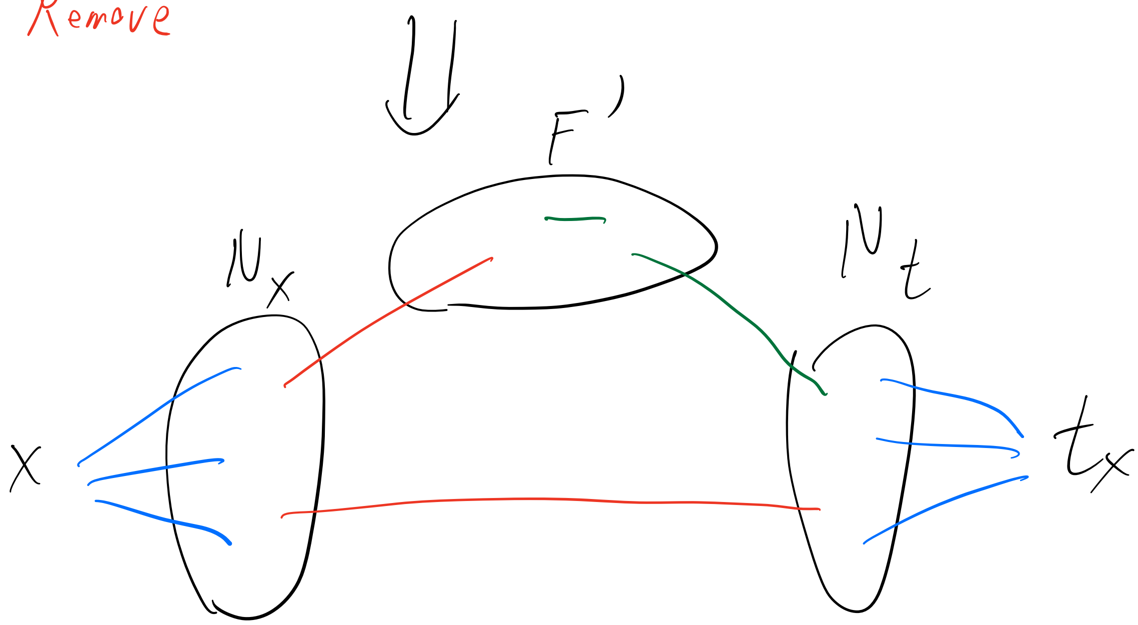
Claim 2: after these removal, there are only $\tilde{O}(kL)$ edges







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 $|N_x| \leq |L| + |S| < 2k$, 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: $|N(x)| < k + |L|$

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

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

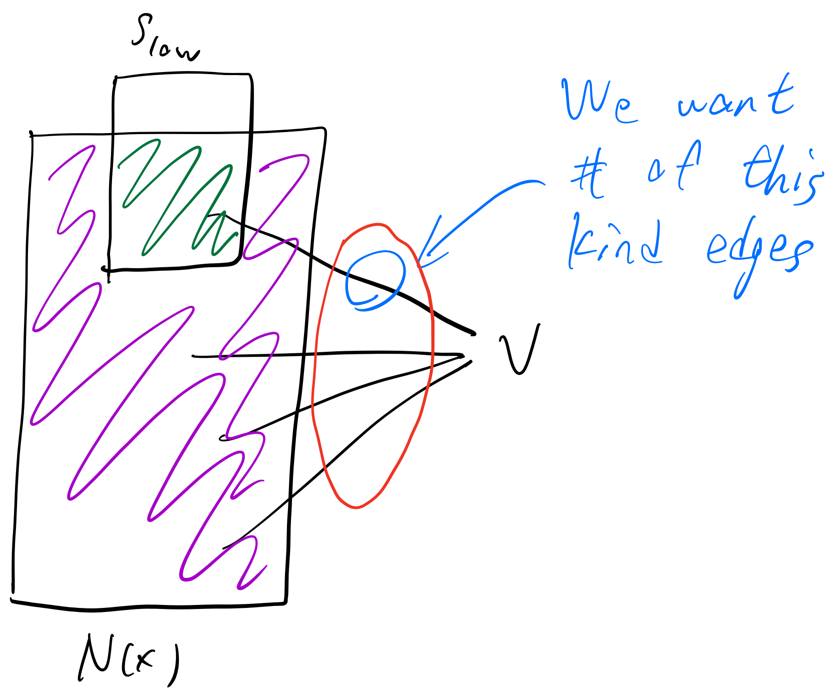
Proof: $|(L \cup S) \setminus N[x]| < |L|$

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

Proof: follows Claim 3

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

Proof:



$$\underbrace{k - |L|\,\mathrm{polylog}(n)} - \underbrace{\left(|N(x)| - (|S_{low}| - |L|)\right)}$$

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

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

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 ℓ, 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: $N(S_v)$