Streaming Algorithms for Matchings in Low Arboricity Graphs

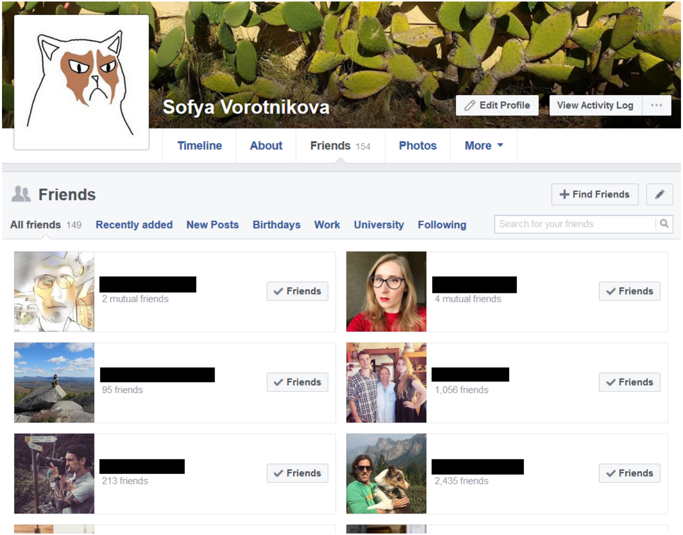
Sofya Vorotnikova

University of Massachusetts Amherst

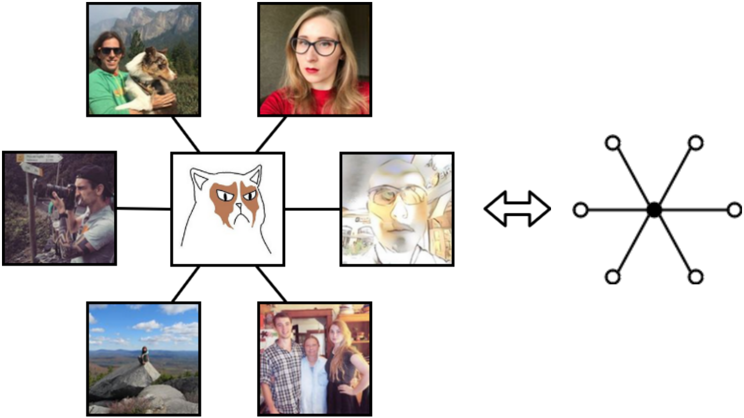
Joint work with Andrew McGregor

Representing Data as a Graph
Example: Social Network

Representing Data as a Graph
Example: Social Network

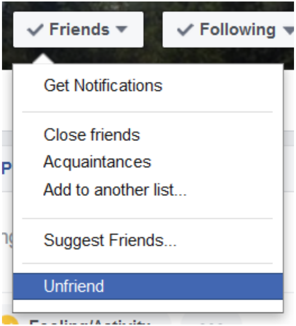


Representing Data as a Graph
Example: Social Network



List of edges incident to a vertex

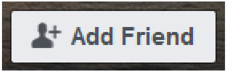
Representing Data as a Graph
Example: Social Network



users can friend and unfriend others $\Longleftrightarrow$ edges of the graph get added and deleted

Updates are not grouped by user/vertex — arbitrary order

Representing Data as a Graph
Example: Social Network



Simpler model: arbitrary order, but only adding edges

Streaming Model(s)

- Vertex set is fixed
- Start with no edges
- Edge updates arrive in a sequence
- One pass

Streaming Model(s)

- Vertex set is fixed
- Start with no edges
- Edge updates arrive in a sequence
- One pass

	insertions	deletions	arbitrary order
dynamic	✓	✓	✓
insert-only	✓	✗	✓
adjacency-list	✓	✗	✗

edges incident to the same vertex arrive together; see every edge twice

Streaming Model: Objectives

- Compute some function of the graph defined by the stream
  - maximum matching, connectivity, number of triangles, etc
- Minimize amount of space: cannot store the entire graph
- Fast update time is generally encouraged
- Solution extraction (postprocessing) time can be large

Why Streaming?

Problem	Streaming Advantage
graph is too large to be stored in main memory	sequential reading from external memory device
graph is distributed across multiple machines	edge-by-edge is an extreme version of batch-by-batch
graph is changing over time	store/update the summary of data

Why Streaming?

Problem	Streaming Advantage
graph is too large to be stored in main memory	sequential reading from external memory device
graph is distributed across multiple machines	edge-by-edge is an extreme version of batch-by-batch
graph is changing over time	store/update the summary of data

restricted model

$+$

general problems

$=$

techniques that extend to other models and can be used in a variety of real-life applications

What Can Be Done in Graph Streams?

Sampling!

- Sample edges uniformly
- Sample edges non-uniformly
- Sample vertices, then collect incident edges

Other things:

- Compute degrees of vertices or other quantities depending on degrees
- Using stream ordering as part of the algorithm

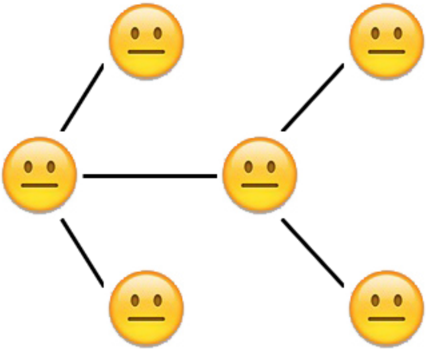
How Can It Be Done?

Sampling a random edge (uniformly)

- Insertions only: reservoir sampling
  - for e_i, the i-th edge in the stream, replace currently stored edge with e_i with probability $1/i$
- Insertions and deletions: L_0-sampling
  - fails with probability δ
  - uses space $O(\log^2 n \log \delta^{-1})$

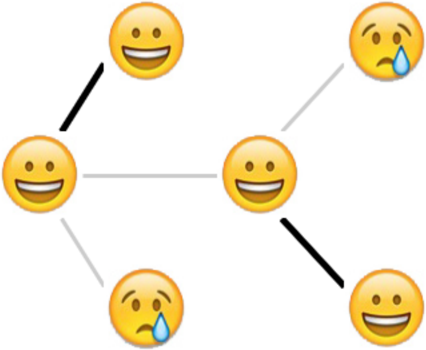
For sampling vertices use hash functions

Problem: Maximum Matching



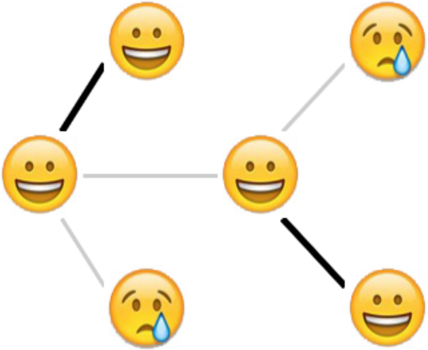
- Department event
- Each grad student can bring a "plus one"

Problem: Maximum Matching



- Department event
- Each grad student can bring a "plus one"
- Want to maximize the number of pairs

Problem: Maximum Matching

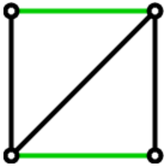


- Department event
- Each grad student can bring a “plus one”
- Want to maximize the number of pairs

List of pairs is then a **matching**.

Approximating Size of Maximum Matching

Matching is a set of edges that don't share endpoints.



In insert-only stream can run greedy algorithm to obtain *maximal* matching, which is a 2-approximation of *maximum* matching.

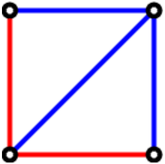
Maximum matching can be as large as $n/2$.

By approximating the **size** of the matching without finding the matching itself, we can use smaller space.

Low Arboricity Graphs

We concentrate on the class of graphs of arboricity α.

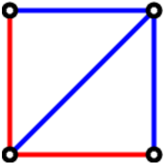
Arboricity is the minimum number of forests into which the edges of the graph can be partitioned.



Low Arboricity Graphs

We concentrate on the class of graphs of arboricity α.

Arboricity is the minimum number of forests into which the edges of the graph can be partitioned.



No dense subgraphs $\Leftrightarrow$ low arboricity.

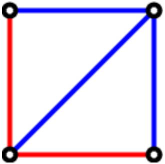
Property: Every subgraph on r vertices has at most αr edges.

Planar graphs have arboricity at most 3.

Low Arboricity Graphs

We concentrate on the class of graphs of arboricity α.

Arboricity is the minimum number of forests into which the edges of the graph can be partitioned.



No dense subgraphs $\Leftrightarrow$ low arboricity.

Property: Every subgraph on r vertices has at most αr edges.

Planar graphs have arboricity at most 3.

In dynamic stream, intermediate graphs can have high arboricity.

Results

	space	approx factor	work
dynamic	$\tilde{O}(\alpha n^{4/5})$	$(5\alpha+9)(1+\epsilon)$	CCEHMMV16
	$\tilde{O}(\alpha n^{4/5})$	$(\alpha+2)(1+\epsilon)$	MV16
	$\tilde{O}(\alpha^{10/3} n^{2/3})$	$(22.5\alpha+6)(1+\epsilon)$	CJMM17*
	$\Omega(\sqrt{n}/\alpha^{2.5})$	$O(\alpha)$	AKL17
insert-only	$\tilde{O}(\alpha n^{2/3})$	$(5\alpha+9)(1+\epsilon)$	EHLMO15
	$\tilde{O}(\alpha n^{2/3})$	$(\alpha+2)(1+\epsilon)$	MV16
	$O(\alpha\epsilon^{-3}\log^2 n)$	$(22.5\alpha+6)(1+\epsilon)$	CJMM17
	$O(\epsilon^{-2}\log n)$	$(\alpha+2)(1+\epsilon)$	MV18
adj	$O(1)$	$\alpha+2$	MV16

*Restriction: $O(\alpha n)$ deletions.

Space is specified in words. An edge or a counter = one word.

Approach

All our results have the following two parts:

- **Structural result**: define Σ that is an $(\alpha + 2)$ approximation of match(G)
- **Algorithm**: $(1 + \epsilon)$ approximation of Σ in streaming (exact computation in adjacency list stream)

Approach

All our results have the following two parts:

- **Structural result**: define Σ that is an $(\alpha + 2)$ approximation of match(G)
- **Algorithm**: $(1 + \epsilon)$ approximation of Σ in streaming (exact computation in adjacency list stream)

Dynamic: Σ_{dyn}

- $(1 + \epsilon)$-approximation in $\tilde{O}(\alpha n^{4/5})$ space
- Also gives $\tilde{O}(\alpha n^{2/3})$ space algorithm in insert-only streams

Insert-only: Σ_{ins}

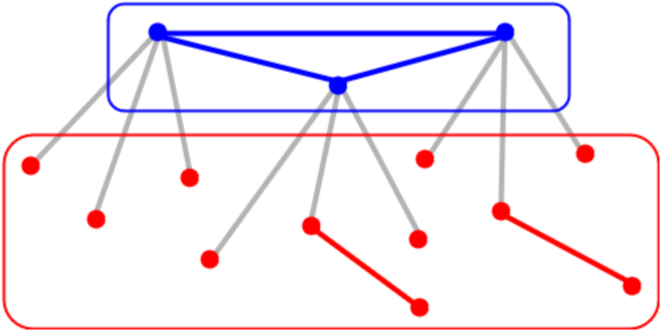
- $(1 + \epsilon)$-approximation in $O(\epsilon^{-2} \log n)$ space

Adjacency list: Σ_{adj}

- Exact computation in $O(1)$ space

Structural Results

Structural Results: Definitions



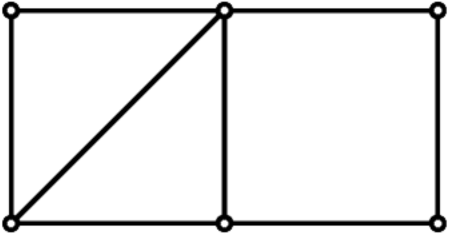
V^H = heavy vertices of degree $\geq \alpha + 2$
E^H = heavy edges with 2 heavy endpoints
V^L = light vertices
E^L = light edges

Structural Results: Definitions: Σ_{adj}

$$\Sigma_{adj} = |E^L| + |V^H|(\alpha + 1) - |E^H|$$

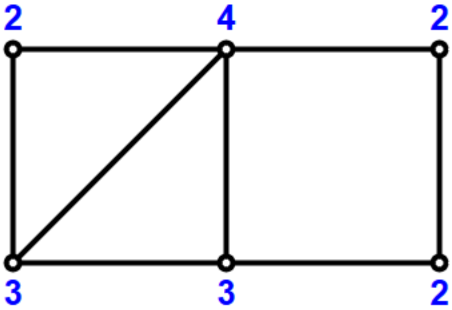
Structural Results: Definitions: Σ_{dyn}

$$x_e = x_{uv} = \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha + 1}\right)$$



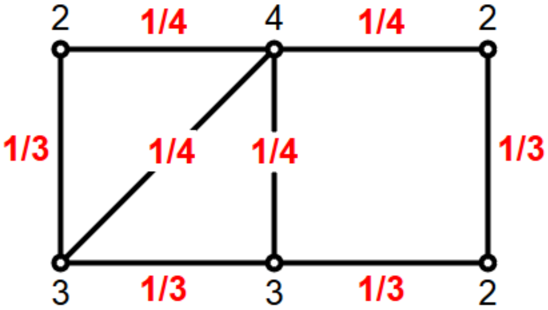
Structural Results: Definitions: Σ_{dyn}

$$x_e = x_{uv} = \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$



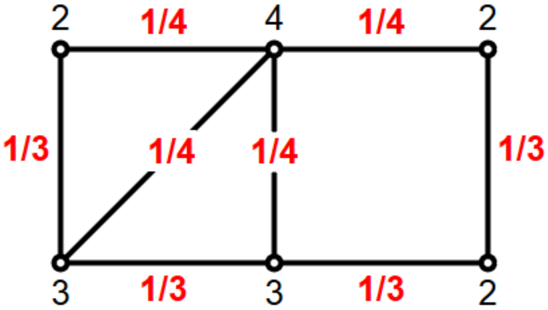
Structural Results: Definitions: Σ_{dyn}

$$x_e = x_{uv} = \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$



Structural Results: Definitions: Σ_{dyn}

$$x_e = x_{uv} = \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$



$$\Sigma_{dyn} = (\alpha+1)\sum_e x_e$$

Structural Results: Σ_{dyn} and Σ_{adj}

$\mathsf{match}(G) \leq \lvert E^L \rvert + \lvert V^H \rvert$	since a matched edge is either light or incident to a heavy vertex
$\leq \lvert E^L \rvert + \lvert V^H \rvert (\alpha + 1) - \lvert E^H \rvert = \Sigma_{adj}$	since $\lvert E^H \rvert \leq \alpha \lvert V^H \rvert$
$\leq (\alpha + 1) \sum_e x_e = \Sigma_{dyn}$	Lemma 1
$\leq (\alpha + 2)\, \mathsf{match}(G)$	Lemma 2

Structural Results: Σ_{dyn} and Σ_{adj}

Lemma 1:

$$\Sigma_{adj} = |E^L| + |V^H|(\alpha + 1) - |E^H| \leq (\alpha + 1) \sum_e x_e = \Sigma_{dyn}$$

- Split $\sum_e x_e$ into 3 sums for $e \in E^L$, $e \in E^H$, and $e \notin E^L, E^H$
- Bound x_e in each case

Structural Results: Σ_{dyn} and Σ_{adj}

Lemma 1:

$$\Sigma_{adj} = |E^L| + |V^H|(\alpha+1) - |E^H| \leq (\alpha+1)\sum_e x_e = \Sigma_{dyn}$$

- Split $\sum_e x_e$ into 3 sums for $e \in E^L$, $e \in E^H$, and $e \notin E^L, E^H$
- Bound x_e in each case

Lemma 2:

$$\Sigma_{dyn} = (\alpha+1)\sum_e x_e \leq (\alpha+2)\,\mathrm{match}(G)$$

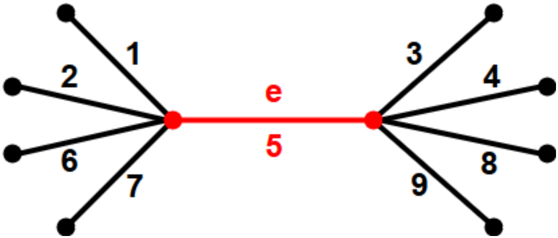
- $\{x_e\}_{e\in E}$ is a fractional matching with max weight $1/(\alpha+1)$
- Use Edmond's thm to relate $\sum_e x_e$ to match(G)

Structural Results: Definitions: Σ_{ins}

Let E_α be the set of edges uv where the number of edges incident to u or v that appear in the stream after uv are both at most α.

Structural Results: Definitions: Σ_{ins}

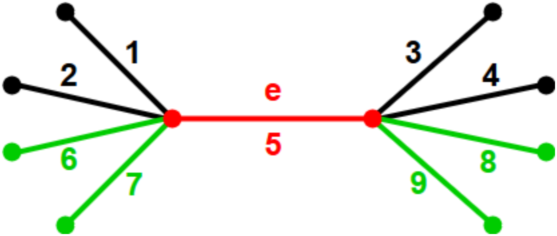
Let E_α be the set of edges *uv* where the number of edges incident to *u* or *v* that appear in the stream after *uv* are both at most α.



$\alpha = 3$

Structural Results: Definitions: Σ_{ins}

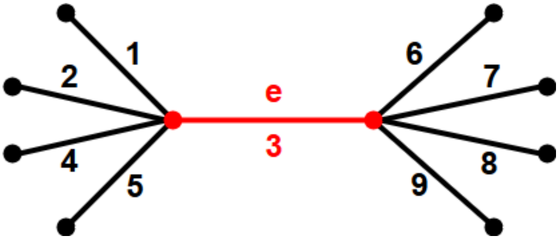
Let E_α be the set of edges uv where the number of edges incident to u or v that appear in the stream after uv are both at most α.



$\alpha = 3$
$e \in E_\alpha$

Structural Results: Definitions: Σ_{ins}

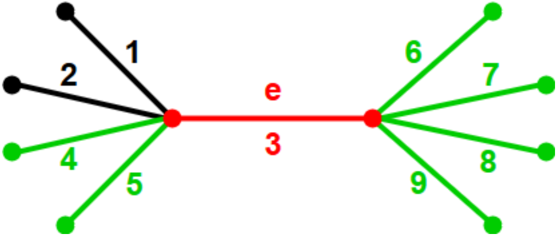
Let E_α be the set of edges uv where the number of edges incident to u or v that appear in the stream after uv are both at most α.



$\alpha = 3$

Structural Results: Definitions: Σ_{ins}

Let E_α be the set of edges uv where the number of edges incident to u or v that appear in the stream after uv are both at most α.



$\alpha = 3$
$e \notin E_\alpha$
E_α depends on stream ordering

Structural Results: Definitions: Σ_{ins}

Lemma 3

$$\text{match}(G) \leq |E_\alpha| \leq (\alpha + 2)\,\text{match}(G)$$

Structural Results: Definitions: Σ_{ins}

Lemma 3

$$\text{match}(G) \leq |E_\alpha| \leq (\alpha + 2)\,\text{match}(G)$$

Let G_t be the graph defined by the first t edges in the stream.

Let E_α^t be $E_\alpha(G_t)$. Then

$$\text{match}(G_t) \leq |E_\alpha^t| \leq (\alpha + 2)\,\text{match}(G_t)$$

Structural Results: Definitions: Σ_{ins}

Lemma 3

$$\text{match}(G) \le |E_\alpha| \le (\alpha + 2)\,\text{match}(G)$$

Let G_t be the graph defined by the first t edges in the stream.

Let E_α^t be $E_\alpha(G_t)$. Then

$$\text{match}(G_t) \le |E_\alpha^t| \le (\alpha + 2)\,\text{match}(G_t)$$

Let $\Sigma_{ins} = \max_t |E_\alpha^t| = |E_\alpha^T|$.

Since $\text{match}(G_t)$ is non-decreasing function of t,

$$\text{match}(G) \le |E_\alpha| \le \Sigma_{ins} = |E_\alpha^T| \le (\alpha{+}2)\,\text{match}(G_T) \le (\alpha{+}2)\,\text{match}(G)$$

Structural Results: Σ_{ins}: Lemma 3

Upper bound:

$$|E_\alpha| \leq (\alpha + 2)\,\text{match}(G)$$

- Let
$$y_e = \begin{cases} 1/(\alpha+1) & \text{if } e \in E_\alpha \\ 0 & \text{otherwise} \end{cases}$$
- $\{y_e\}_{e \in E}$ is a fractional matching with max weight $1/(\alpha + 1)$
- $\sum_e y_e = |E_\alpha|/(\alpha + 1)$
- Use Edmond's thm to relate $\sum_e y_e$ to match(G)

Structural Results: Σ_{ins}: Lemma 3

Upper bound:

$$|E_\alpha| \leq (\alpha + 2)\,\text{match}(G)$$

- Let
$$y_e = \begin{cases} 1/(\alpha+1) & \text{if } e \in E_\alpha \\ 0 & \text{otherwise} \end{cases}$$
- $\{y_e\}_{e \in E}$ is a fractional matching with max weight $1/(\alpha + 1)$
- $\sum_e y_e = |E_\alpha|/(\alpha + 1)$
- Use Edmond's thm to relate $\sum_e y_e$ to match(G)

Lower bound:

$$|E_\alpha| \geq \text{match}(G)$$

- Count light edges and edges on heavy vertices in E_α to show $|E_\alpha| \geq |E^L| + |V^H| \geq \text{match}(G)$

Algorithms

Algorithms: Dynamic Stream

$$\Sigma_{dyn} = (1+\alpha)\sum_e x_e = (1+\alpha)\sum_e \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$

In parallel:

If matching is small: $\leq \boldsymbol{n^{2/5}}$

- Use algorithm for bounded size matchings [CCEHMMV16]: $\tilde{O}(n^{4/5})$ space

If matching is large: $> \boldsymbol{n^{2/5}}$

- Estimate Σ_{dyn} by computing x_e for a particular set of edges
- Accurate since matching and thus Σ_{dyn} are large

Algorithms: Dynamic Stream

$$\Sigma_{dyn} = (1+\alpha)\sum_e x_e = (1+\alpha)\sum_e \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$

In parallel:

If matching is small: $\leq n^{2/5}$

- Use algorithm for bounded size matchings [CCEHMMV16]: $\tilde{O}(n^{4/5})$ space

If matching is large: $> n^{2/5}$

- Estimate Σ_{dyn} by computing x_e for a particular set of edges
- Accurate since matching and thus Σ_{dyn} are large

Note: In insert-only streams, can use greedy algorithm for approximating small matching. Reduces total space to $\tilde{O}(\alpha n^{2/3})$.

Algorithms: Dynamic Stream

$$\Sigma_{dyn} = (1+\alpha)\sum_e x_e = (1+\alpha)\sum_e \min\left(\frac{1}{d(u)}, \frac{1}{d(v)}, \frac{1}{\alpha+1}\right)$$

Estimating Σ_{dyn}

- Sample a set of vertices T with probability $p = \tilde{\Theta}(1/n^{1/5})$
  - $|T| = \tilde{\Theta}(n^{4/5})$
- Compute degrees of vertices in T
- Let E_T be edges with both endpoints in T
  - $|E_T| = \tilde{O}(\alpha n^{4/5})$ at the end of the stream
  - $|E_T|$ can be larger in the middle of the stream
- Sample $\min(|E_T|, \tilde{\Theta}(\alpha n^{4/5}))$ edges in E_T
- Use $(\alpha+1)/p \cdot \sum_{e \in E_T} x_e$ as estimate

Algorithms: Insert-only Stream

$$\Sigma_{ins} = \max_t |E_\alpha^t|$$

where E_α^t is the set of edges uv, s.t. the number of edges incident to u or v between arrival of uv and time t is at most α.

Idea: keep a sample of edges in E_α^t by sampling with probability that allows us to

- keep an accurate approximation of $|E_\alpha^t|$
- use small amount of space

Algorithms: Insert-only Stream

$$\Sigma_{ins} = \max_t |E_\alpha^t|$$

where E_α^t is the set of edges uv, s.t. the number of edges incident to u or v between arrival of uv and time t is at most α.

1. Set $p \leftarrow 1$
2. Start sampling each edge with probability p
3. If e is sampled:
   - store e
   - store counters for degrees of endpoints in the rest of the stream
   - if later we detect $e \notin E_\alpha^t$, it is deleted
4. If the number of stored edges $> 40\epsilon^{-2} \log n$
   - $p \leftarrow p/2$
   - delete every edge currently stored with probability $1/2$
5. Return $\max_t \frac{\#\text{ samples at time } t}{p \text{ at time } t}$

Algorithms: Insert-only Stream

$$\Sigma_{ins} = \max_t |E^t_\alpha|$$

where E^t_α is the set of edges uv, s.t. the number of edges incident to u or v between arrival of uv and time t is at most α.

Let k be s.t. $(20\epsilon^{-2}\log n)2^{k-1} \le \Sigma_{ins} < (20\epsilon^{-2}\log n)2^k$.

We show that whp:

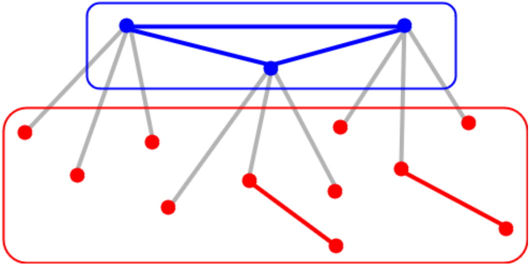
1. If sampling probability is high enough ($\ge 1/2^k$), can compute $|E^t_\alpha| \pm \epsilon\Sigma_{ins}$ for all t. From Chernoff and union bounds.

2. We do not switch to probability that is too low ($< 1/2^k$), since the $\#$ edges sampled wp $1/2^k$ does not exceed $(1+\epsilon)\Sigma_{ins}/2^k < (1+\epsilon)(20\epsilon^{-2}\log n) \le 40\epsilon^{-2}\log n$.

Algorithms: Adjacency List Stream

$$\Sigma_{adj} = |E^L| + |V^H|(\alpha + 1) - |E^H|$$

Treat adjacency stream as a degree sequence of the graph.
$|V^H|$ can be computed easily.



$$|E^L| - |E^H| = |E| - \sum_{h \in V^H} d(h)$$

which is also easy to compute.

Conclusion

Summary:

- There are quantities that provide good approximation of the size of maximum matching in graphs of arboricity α.
- Computing those quantities can be done efficiently.

Open questions:

- Better than $\alpha + 2$ approximation.
- Closing the gap between upper and lower bounds for dynamic streams.

Thank you for your attention!