# Latest on Linear Sketches for Large Graphs: Lots of Problems, Little Space, and Loads of Handwaving

*Andrew McGregor*
University of Massachusetts

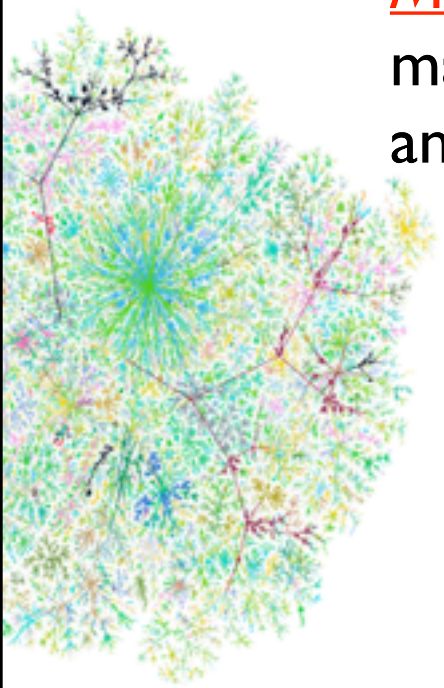
# Latest on Linear Sketches for Large Graphs: Lots of Problems, Little Space, and Loads of Handwaving

*Andrew McGregor*
University of Massachusetts

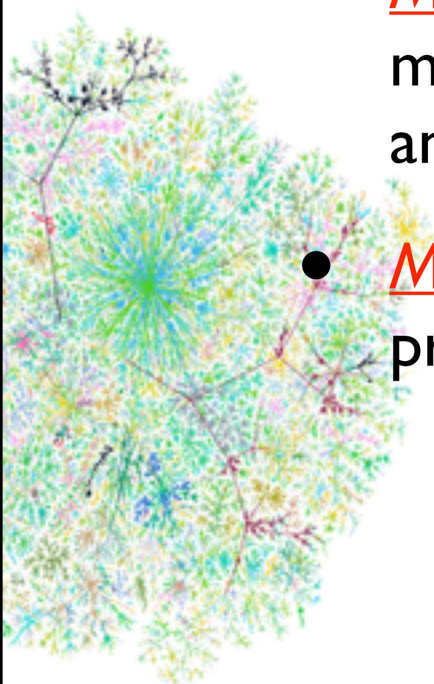
**Vertex Connectivity and Sparsification** — Guha, McGregor, Tench [PODS 2015]

**Densest Subgraphs** — McGregor, Tench, Vorotnikova, Vu [MFCS 2015]

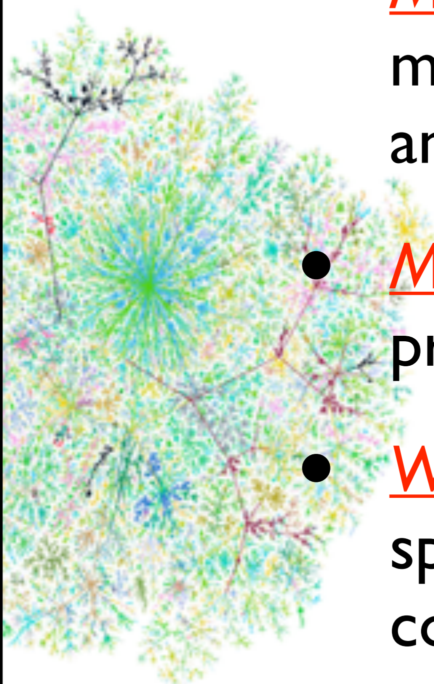
**Matching, Vertex Cover, Hitting Set**
Chitnis, Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova [TBA 2016]



- *Motivation:* Dynamic Graph Streams. Want to analyze a massive graph defined by a long sequence of edge insertions and deletions. Don't want to have to store the entire graph.



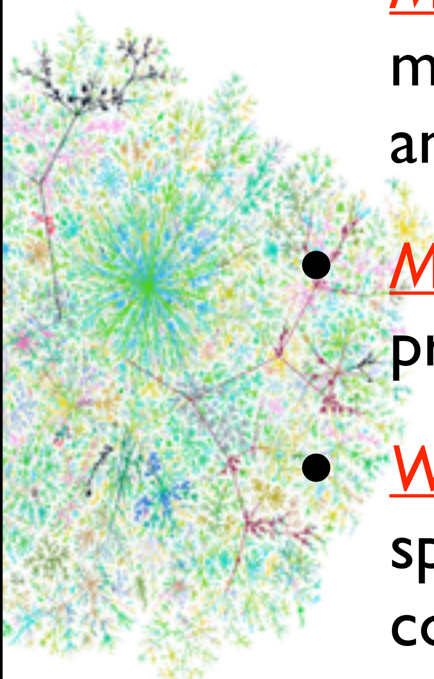
- *Motivation:* Dynamic Graph Streams. Want to analyze a massive graph defined by a long sequence of edge insertions and deletions. Don't want to have to store the entire graph.
- *Main Technique:* Linear Sketches. Maintain a random linear projections of vectors and matrices representing the graph.



- ***Motivation:*** Dynamic Graph Streams. Want to analyze a massive graph defined by a long sequence of edge insertions and deletions. Don't want to have to store the entire graph.

- ***Main Technique:*** Linear Sketches. Maintain a random linear projections of vectors and matrices representing the graph.

- ***What's Known:*** Lots and lots! Edge and vertex connectivity, spectral sparsification, matching, vertex cover, hitting set, correlation clustering, triangles, spanners, densest subgraph…



- *Motivation:* Dynamic Graph Streams. Want to analyze a massive graph defined by a long sequence of edge insertions and deletions. Don't want to have to store the entire graph.
- *Main Technique:* Linear Sketches. Maintain a random linear projections of vectors and matrices representing the graph.
- *What's Known:* Lots and lots! Edge and vertex connectivity, spectral sparsification, matching, vertex cover, hitting set, correlation clustering, triangles, spanners, densest subgraph…


Graph Stream Algorithms: A Survey

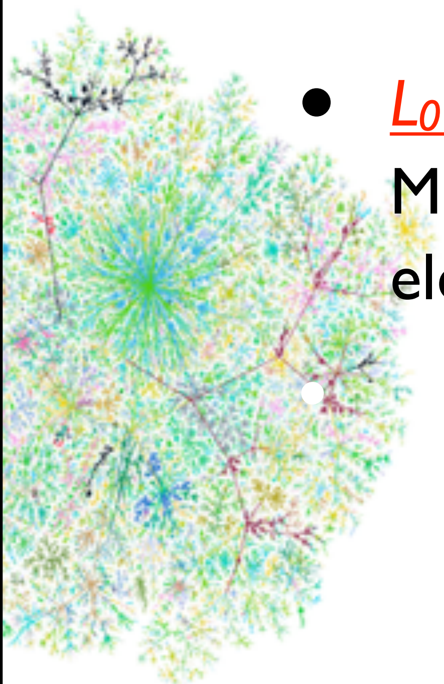
Andrew McGregor

ABSTRACT

1. INTRODUCTION

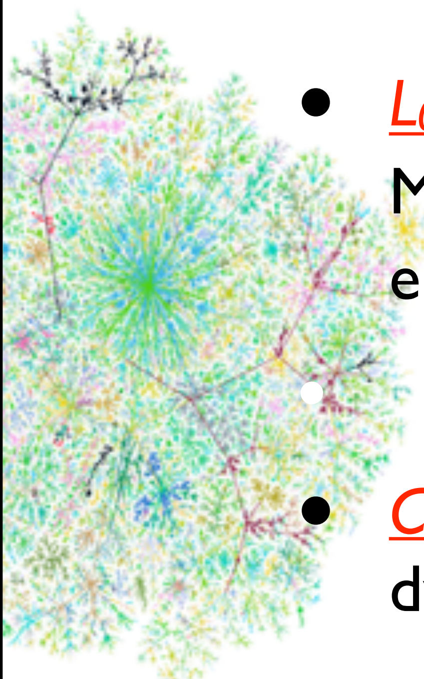
**Graph Streaming Survey**

**McGregor [SIGMOD Record 2014]**



- *$L_0$ Sampling Primitive* There's a distribution over matrices $M \in \Re^{polylog(N) \times N}$ such that for any $x \in \Re^N$, a random non-zero element of x can be reconstructed from Mx whp.

*Jowhari, Saglam, Tardos* [PODS 2011]



- *$L_0$ Sampling Primitive* There's a distribution over matrices $M \in \Re^{\text{polylog}(N) \times N}$ such that for any $x \in \Re^N$, a random non-zero element of x can be reconstructed from Mx whp.

*Jowhari, Saglam, Tardos* [PODS 2011]

- *Corollary* Can sample a uniform edge from a graph in the dynamic graph stream model using O(polylog n) bits of space.

# Densest Subgraph

- Density of node set S is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.

# Densest Subgraph

- Density of node set $S$ is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2}\, n)$ space.

*Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]

# Densest Subgraph

- Density of node set S is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]
- *Our Result* Single pass $(1+\varepsilon)$-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

# Densest Subgraph

- Density of node set S is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]
- *Our Result* Single pass $(1+\varepsilon)$-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

  Sample of $t=O(n \log n)$ edges using $L_0$ sampling. Let $\check{D}_S$ be density among sampled edge scaled by m/t. Return $\max_S \check{D}_S$

# Densest Subgraph

- Density of node set $S$ is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]

- *Our Result* Single pass $(1+\varepsilon)$-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

  Sample of $t=O(n \log n)$ edges using $L_0$ sampling. Let $\check{D}_S$ be density among sampled edge scaled by $m/t$. Return $\max_S \check{D}_S$

- *Analysis* With probability $1-n^{-2k}$ for any subset $S$ of size $k$,

# Densest Subgraph

- Density of node set $S$ is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]
- *Our Result* Single pass $(1+\varepsilon)$-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

  Sample of $t=O(n \log n)$ edges using $L_0$ sampling. Let $\check{D}_S$ be density among sampled edge scaled by $m/t$. Return $\max_S \check{D}_S$
- *Analysis* With probability $1-n^{-2k}$ for any subset $S$ of size $k$,

$$\check{D}_S \approx_\varepsilon D_S \text{ if } D_S \approx D^* \quad \text{and} \quad \check{D}_S \ll D^* \text{ if } D_S \ll D^*$$

# Densest Subgraph

- Density of node set S is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* 2+ε approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]

- *Our Result* Single pass (1+ε)-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

  Sample of $t=O(n \log n)$ edges using $L_0$ sampling. Let $\check{D}_S$ be density among sampled edge scaled by m/t. Return $\max_S \check{D}_S$

- *Analysis* With probability $1-n^{-2k}$ for any subset S of size k,

  $$\check{D}_S \approx_\varepsilon D_S \text{ if } D_S \approx D^* \quad \text{and} \quad \check{D}_S \ll D^* \text{ if } D_S \ll D^*$$

  Use union bound over $O(n^k)$ subsets of size k for each k.

# Densest Subgraph

- Density of node set S is $D_S = |E_S|/|S|$. Estimate $D^* = \max_S D_S$.
- *Previous Result* $2+\varepsilon$ approximations using $\tilde{O}(\varepsilon^{-2} n)$ space.

  *Bhattycharya et al.* [STOC 2015], *Bahmani et al.* [PVLDB 2012]

- *Our Result* Single pass $(1+\varepsilon)$-approx. using $\tilde{O}(\varepsilon^{-2} n)$ space:

  Sample of $t=O(n \log n)$ edges using $L_0$ sampling. Let $\check{D}_S$ be density among sampled edge scaled by $m/t$. Return $\max_S \check{D}_S$

- *Analysis* With probability $1-n^{-2k}$ for any subset S of size k,

  $$\check{D}_S \approx_\varepsilon D_S \text{ if } D_S \approx D^* \quad \text{and} \quad \check{D}_S \ll D^* \text{ if } D_S \ll D^*$$

  Use union bound over $O(n^k)$ subsets of size k for each k.

see also *Mitzenmacher et al.* [KDD 2015], *Esfandiari et al.* [ArXiv 2015]

*What other types of sampling are there that a) are useful for solving graph problems and b) can be supported on dynamic graph streams?*

*I*. Graph Matching
via **SNAPE** Sampling


*II*. Graph Connectivity
via **DEALS** Sampling

# Graph Matchings

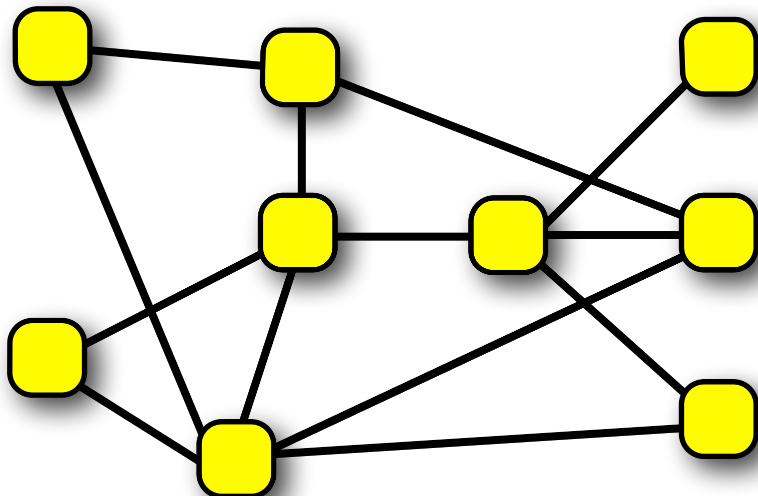
- *1st Result* If max matching has size $\leq k$, can find optimal matching in dynamic stream model using $\tilde{O}(k^2)$ space.
  - *Optimal & Simple.* Extends to hypergraph matching, vertex cover, hitting set… but gets a lot more complicated.
  - *Basic Idea:* "SNAPE" sampling primitive.

# Graph Matchings

- *1st Result* If max matching has size $\leq k$, can find optimal matching in dynamic stream model using $\tilde{O}(k^2)$ space.
  - *Optimal & Simple.* Extends to hypergraph matching, vertex cover, hitting set… but gets a lot more complicated.
  - *Basic Idea:* "SNAPE" sampling primitive.
- *2nd Result* If max matching has size $\geq k$, can find matching of size $\Omega(k/t)$ in the dynamic stream model using $\tilde{O}(k^2/t^3)$ space.
  - *Application:* Guessing k gives O(t)-approx for max matching using $\tilde{O}(n^2/t^3)$ space. This is also optimal; see Sanjeev's talk.

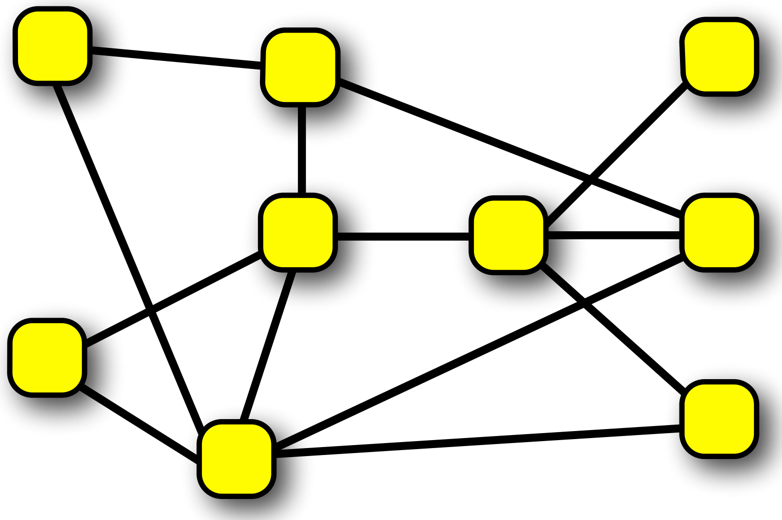
# SNAPE Sampling

## Sample-Nodes-And-Pick-Edge



# SNAPE Sampling

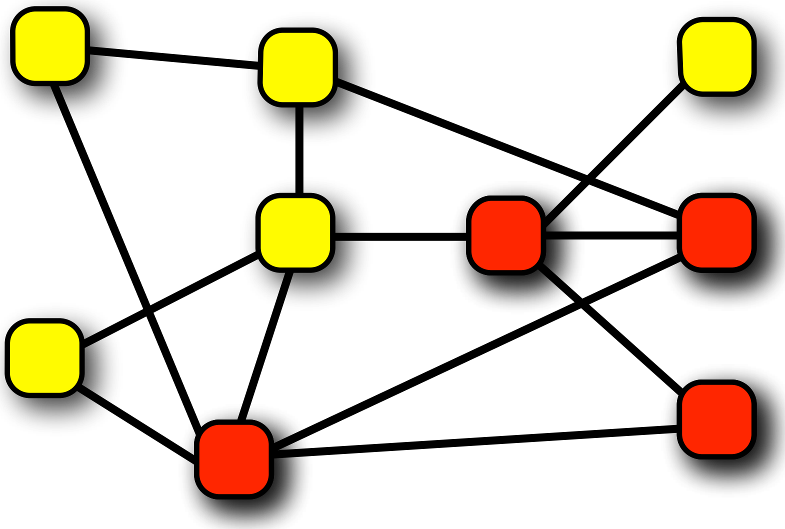
## Sample-Nodes-And-Pick-Edge



- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest

# SNAPE Sampling

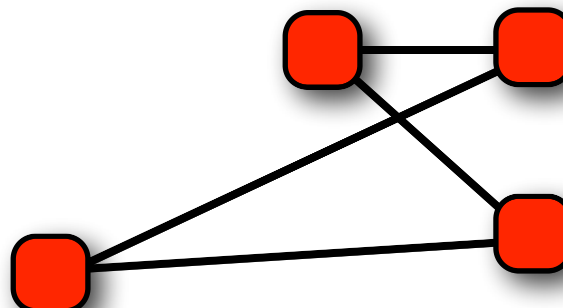
## Sample-Nodes-And-Pick-Edge



- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest

# SNAPE Sampling

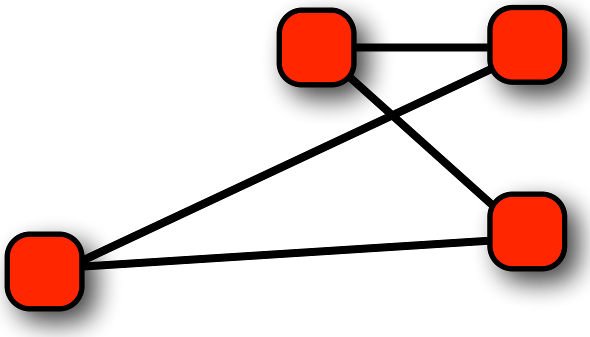
## Sample-Nodes-And-Pick-Edge



- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest

# SNAPE Sampling

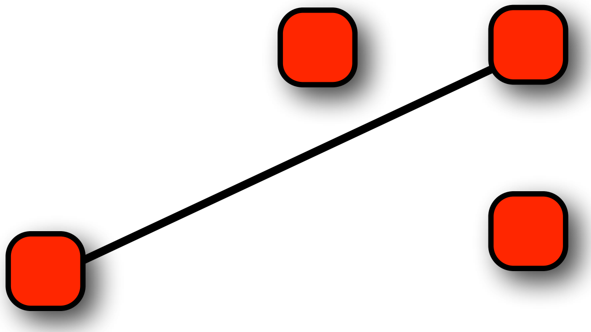
## Sample-Nodes-And-Pick-Edge



- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest
- RETURN a random edge amongst those that remain. If no edges remain, return NULL.

# SNAPE Sampling

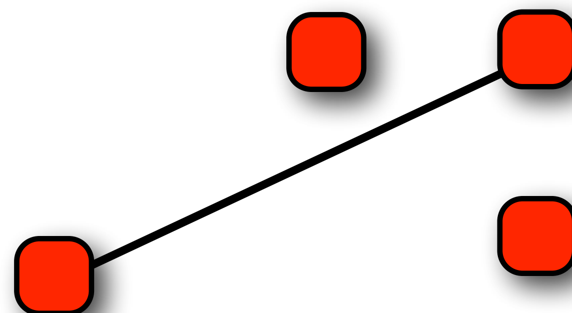
## Sample-Nodes-And-Pick-Edge



- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest
- RETURN a random edge amongst those that remain. If no edges remain, return NULL.

# SNAPE Sampling

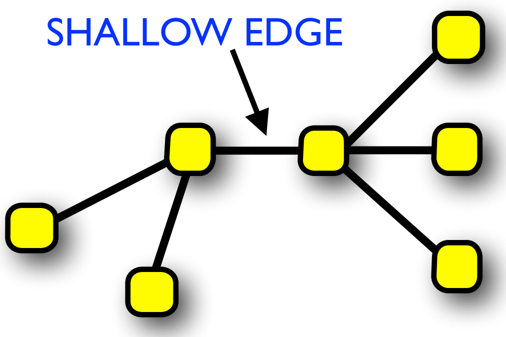
## Sample-Nodes-And-Pick-Edge

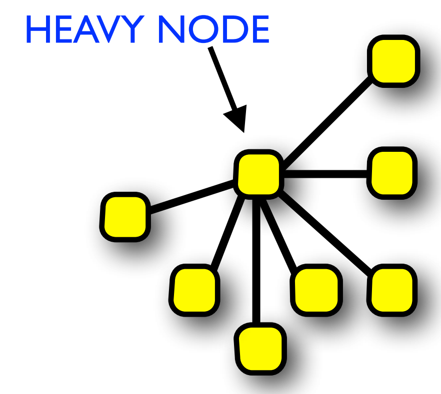


- SAMPLE each node with prob. $\Theta(1/k)$ and DELETE the rest
- RETURN a random edge amongst those that remain. If no edges remain, return NULL.
- *Theorem* If G has max matching size $\leq k$ then $O(k^2 \log k)$ SNAPE samples will include a max matching from G.

# *Small Matching Analysis: Basic Idea*

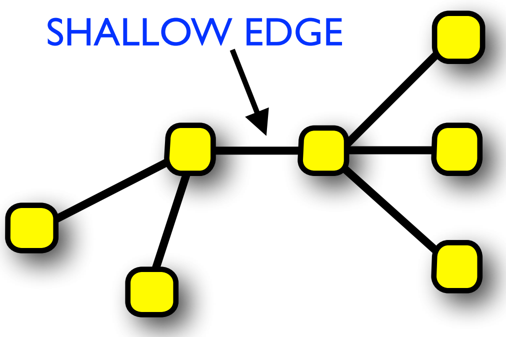
- Let G have max matching of size $\leq$k. Say node is heavy if degree is $\geq$10k and edge is shallow if both endpoints aren't heavy.

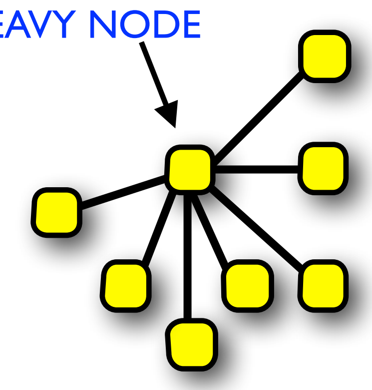


# *Small Matching Analysis: Basic Idea*

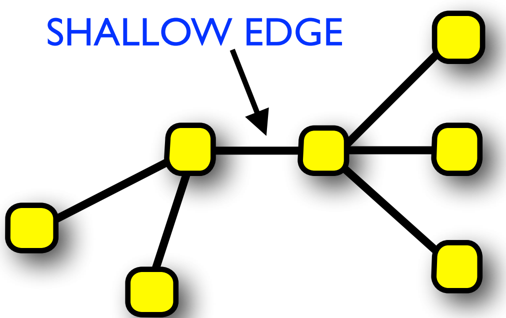
- Let G have max matching of size $\leq$k. Say node is heavy if degree is $\geq$10k and edge is shallow if both endpoints aren't heavy.

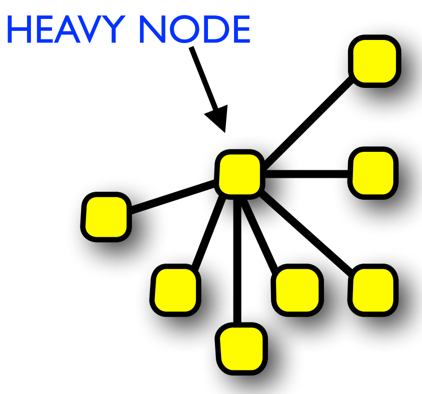


# *Small Matching Analysis: Basic Idea*

- Let G have max matching of size ≤k. Say node is heavy if degree is ≥10k and edge is shallow if both endpoints aren't heavy.

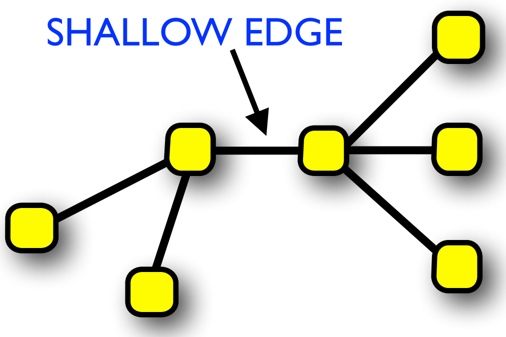


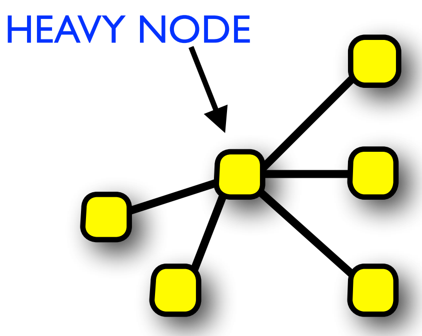




- *Lemma* Let G' contains a max matching of G if:
  - G' includes all shallow edges in G.
  - Every heavy node in G has degree at least 5k in G'.

# *Small Matching Analysis: Basic Idea*

- Let G have max matching of size ≤k. Say node is heavy if degree is ≥10k and edge is shallow if both endpoints aren't heavy.

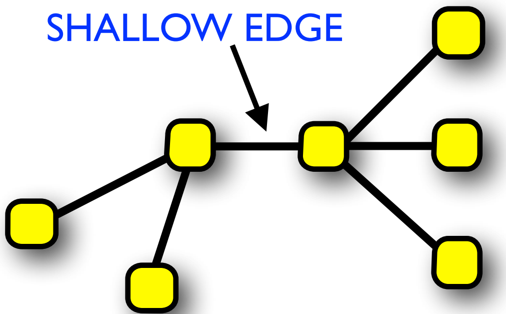


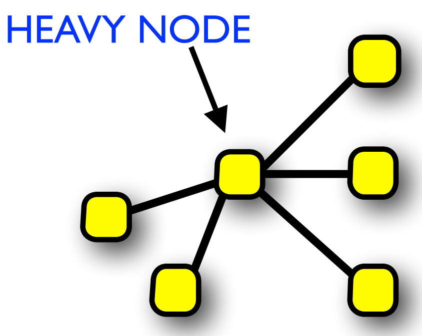




- *Lemma* Let G' contains a max matching of G if:
  - G' includes all shallow edges in G.
  - Every heavy node in G has degree at least 5k in G'.

# *Small Matching Analysis: Basic Idea*

- Let G have max matching of size $\leq$k. Say node is heavy if degree is $\geq$10k and edge is shallow if both endpoints aren't heavy.

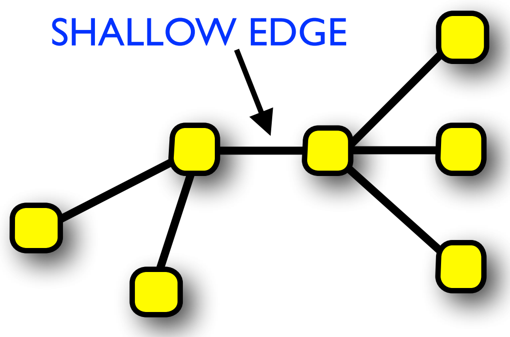


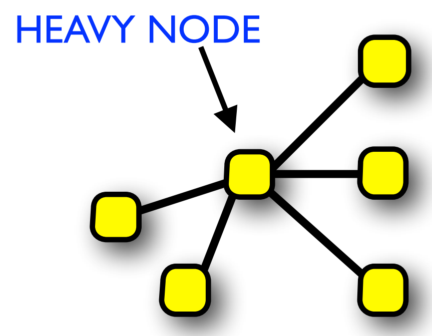




- *Lemma* Let G' contains a max matching of G if:
  - G' includes all shallow edges in G.
  - Every heavy node in G has degree at least 5k in G'.
- *Proof* Each missing edge is incident to some heavy node but you still have plenty of other edges on that node.

# *Small Matching Analysis: Basic Idea*

- Let G have max matching of size ≤k. Say node is heavy if degree is ≥10k and edge is shallow if both endpoints aren't heavy.

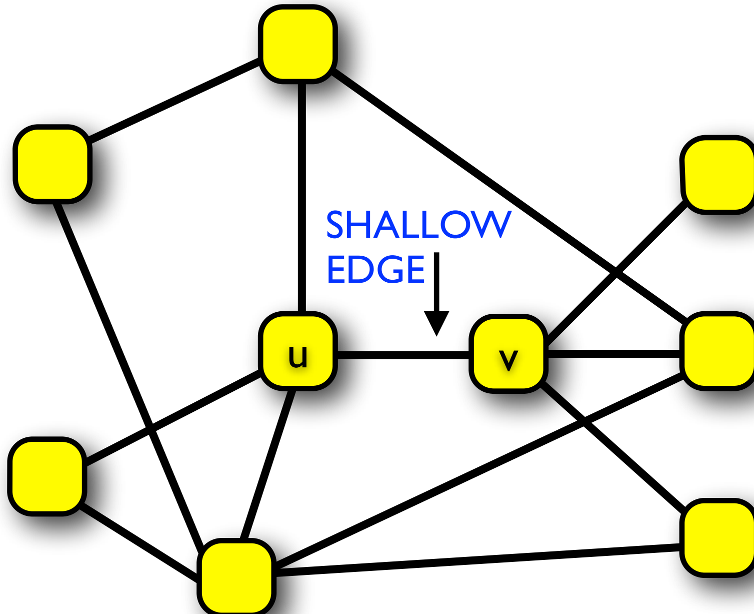


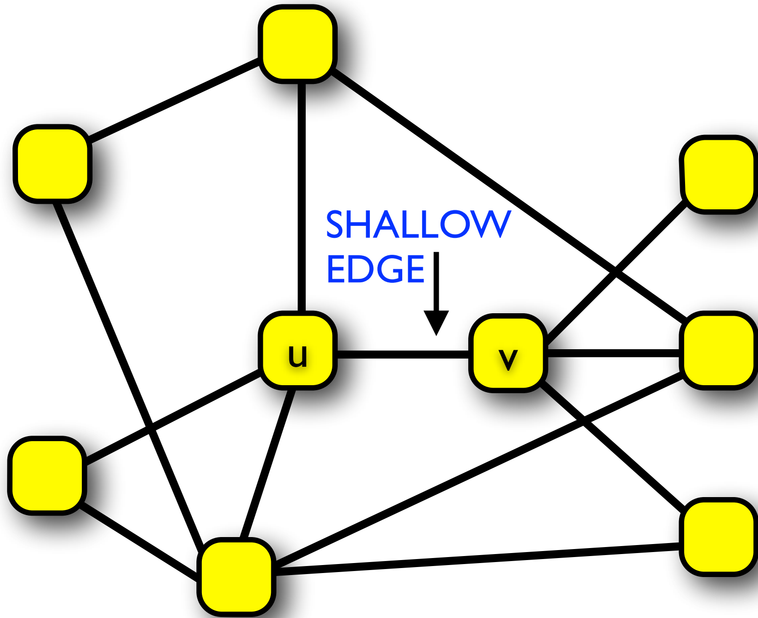

- *Lemma* Let G' contains a max matching of G if:
    - G' includes all shallow edges in G.
    - Every heavy node in G has degree at least 5k in G'.
- *Proof* Each missing edge is incident to some heavy node but you still have plenty of other edges on that node.

- *Useful Fact* G has a vertex cover W of size at most 2k.

# *Small Matching Analysis: Shallow Edges*

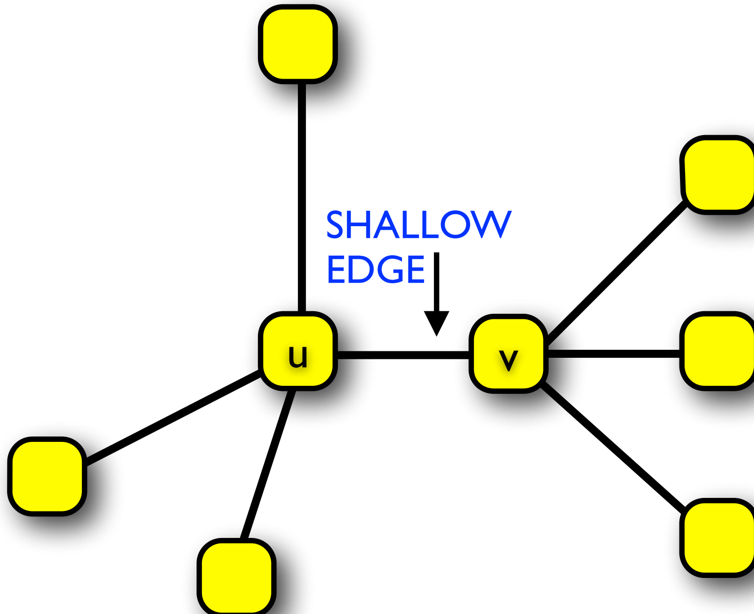


# *Small Matching Analysis: Shallow Edges*



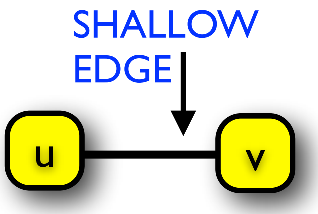

- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.

# *Small Matching Analysis: Shallow Edges*



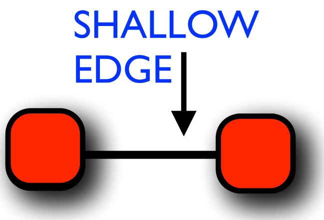

- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.

# *Small Matching Analysis: Shallow Edges*



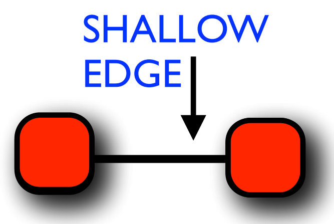

- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.

# *Small Matching Analysis: Shallow Edges*




- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.

# *Small Matching Analysis: Shallow Edges*

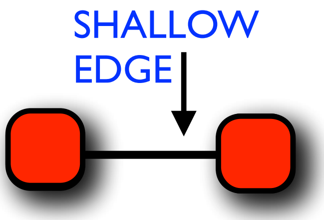



- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.

- Hence, if uv is shallow:

$$\Pr[\text{uv is only remaining edge}] \geq p^2(1-p)^{|\Gamma(u)|+|\Gamma(v)|+|W|} = \Omega(k^{-2})$$

# *Small Matching Analysis: Shallow Edges*

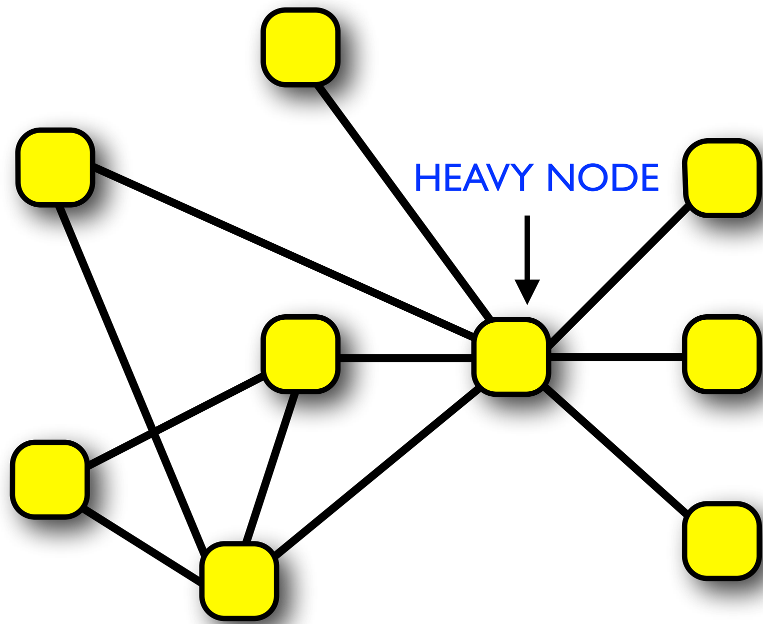



- If we delete nodes (other than u and v) in hitting set W and neighbors of u, v leaves exactly the edge uv if u and v sampled.
- Hence, if uv is shallow:

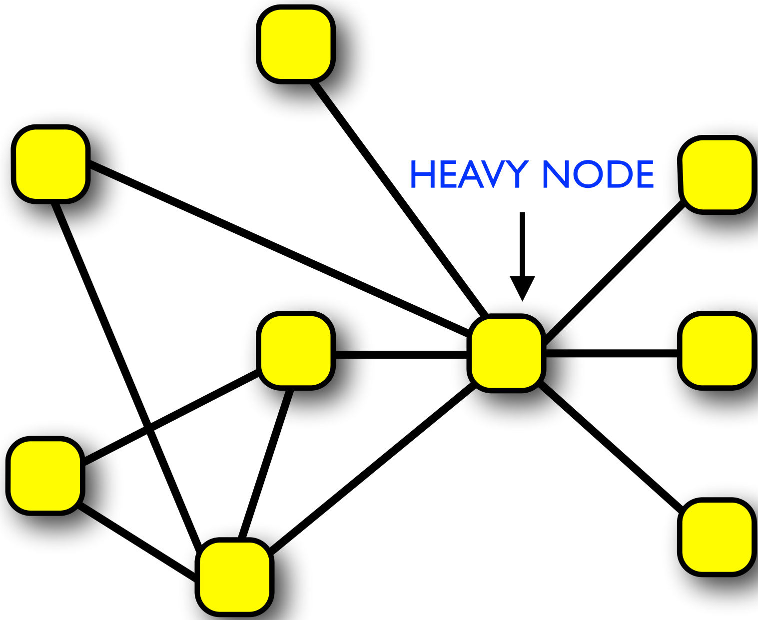
$$\Pr[\text{uv is only remaining edge}] \geq p^2(1-p)^{|\Gamma(u)|+|\Gamma(v)|+|W|} = \Omega(k^{-2})$$

- After $O(k^2 \log k)$ repetitions, have sampled edge uv whp.

# *Small Matching Analysis: Edges on Heavy Nodes*

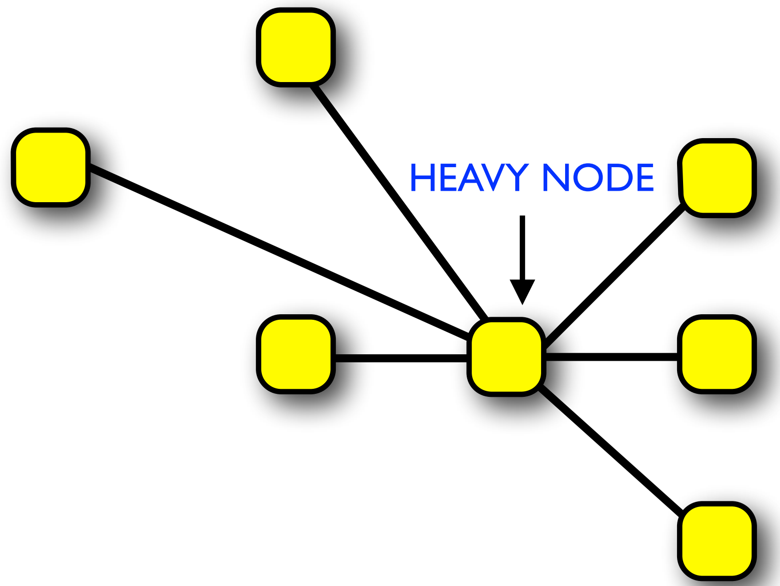


# *Small Matching Analysis: Edges on Heavy Nodes*



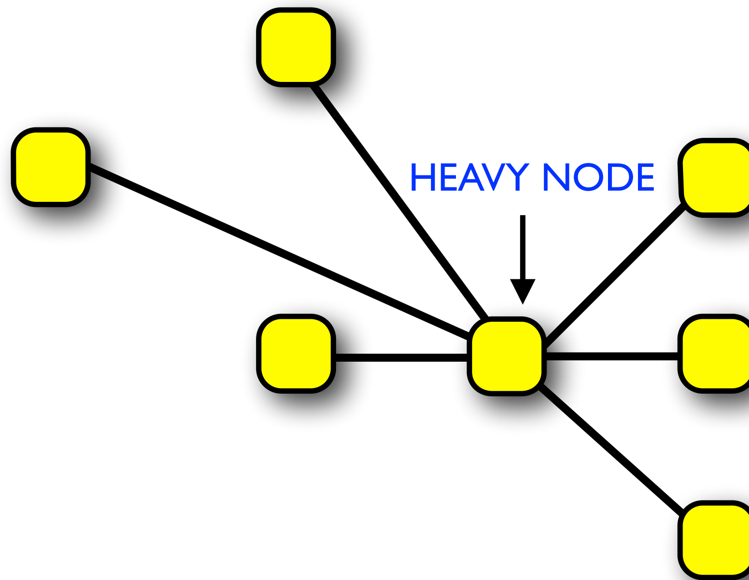

- For heavy u, deleting W\{u} leaves star on u with ≥ 8k leaves.

# *Small Matching Analysis: Edges on Heavy Nodes*




- For heavy u, deleting W\{u} leaves star on u with $\geq$ 8k leaves.

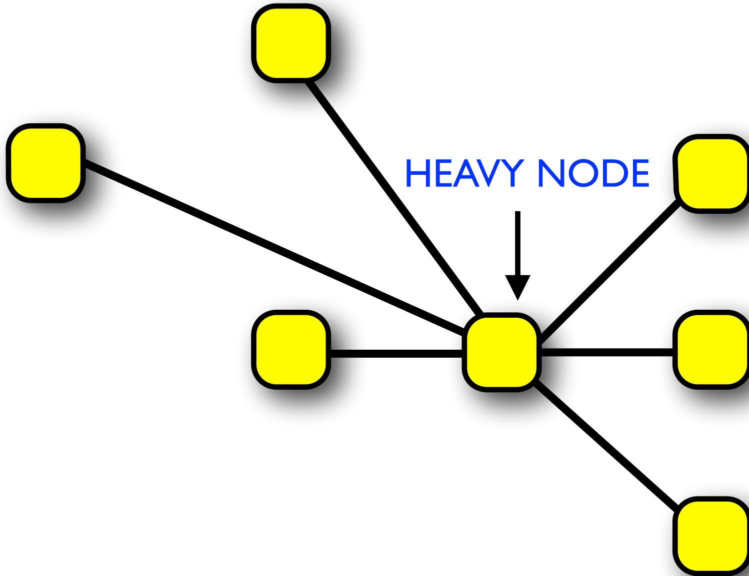
# *Small Matching Analysis: Edges on Heavy Nodes*




- For heavy u, deleting W\{u} leaves star on u with ≥ 8k leaves.
- Hence,

$$\Pr[\text{edge incident to u is sampled}] \geq p(1-p)^{|W|} = \Omega(k^{-1})$$

# *Small Matching Analysis: Edges on Heavy Nodes*




- For heavy u, deleting W\{u} leaves star on u with ≥ 8k leaves.
- Hence,

$$\Pr[\text{edge incident to u is sampled}] \geq p(1-p)^{|W|} = \Omega(k^{-1})$$

- After $O(k^2 \log k)$ repetitions, have sampled 5k edges on u.

# *Approximate Matching: Basic Idea*

- *Theorem* If G has matching $\geq k$ then $O(k/t^3)$ SNAPE samples with $p=\Theta(t/k)$ has matching of size $\Omega(k/t)$ with high probability.

# *Approximate Matching: Basic Idea*

- *Theorem* If G has matching $\geq k$ then $O(k/t^3)$ SNAPE samples with $p=\Theta(t/k)$ has matching of size $\Omega(k/t)$ with high probability.
- *Proof*
  - Let $e_1, e_2, e_3, e_4, \ldots$ be sequence of SNAPE samples and consider constructing greedy matching M.

# *Approximate Matching: Basic Idea*

- *Theorem* If G has matching $\geq$k then $O(k/t^3)$ SNAPE samples with $p=\Theta(t/k)$ has matching of size $\Omega(k/t)$ with high probability.
- *Proof*
  - Let $e_1, e_2, e_3, e_4, \ldots$ be sequence of SNAPE samples and consider constructing greedy matching M.
  - Assuming $|M|=o(k/t)$ then

$$\Pr[e_i \text{ added to } M] \approx \Pr[e_i \text{ isn't a NULL}] \cdot \Pr[\text{all endpoints in } M \text{ are deleted}]$$
$$= \Omega(kp^2) \cdot (1-p)^{o(k/t)} = \Omega(t^2/k)$$

# *Approximate Matching: Basic Idea*

- *Theorem* If G has matching $\geq$k then $O(k/t^3)$ SNAPE samples with $p=\Theta(t/k)$ has matching of size $\Omega(k/t)$ with high probability.
- *Proof*
  - Let $e_1, e_2, e_3, e_4, \ldots$ be sequence of SNAPE samples and consider constructing greedy matching M.
  - Assuming $|M|=o(k/t)$ then

$$\begin{aligned}\Pr[e_i \text{ added to } M] &\approx \Pr[e_i \text{ isn't a NULL}] \cdot \Pr[\text{all endpoints in } M \text{ are deleted}]\\ &= \Omega(kp^2) \cdot (1-p)^{o(k/t)} = \Omega(t^2/k)\end{aligned}$$

  - After $O(k/t^2)$ SNAPE samples we have $|M|= \Omega(k/t)$

*I*. Graph Matching
via **SNAPE** Sampling


*II*. Graph Connectivity
via **DEALS** Sampling

# Graph Connectivity

# Graph Connectivity

- *1st Result* Test if graph is k-edge-connected in $\tilde{O}(kn)$ space.
  - Basic Idea: "DEALS" sampling primitive.

# Graph Connectivity

- *1st Result* Test if graph is k-edge-connected in $\tilde{O}(kn)$ space.
  - Basic Idea: "DEALS" sampling primitive.
- *2nd Result* Distinguish node connectivity $\leq k$ from $\geq (1+\varepsilon)k$ using $\tilde{O}(\varepsilon^{-1}kn)$ space.
  - Basic Idea: Combine node sampling and DEALS sampling.
  - Open: Testing exactly exact node connectivity?

# Graph Connectivity

- *1st Result* Test if graph is k-edge-connected in $\tilde{O}(kn)$ space.
  - Basic Idea: "DEALS" sampling primitive.
- *2nd Result* Distinguish node connectivity $\leq k$ from $\geq (1+\varepsilon)k$ using $\tilde{O}(\varepsilon^{-1}kn)$ space.
  - Basic Idea: Combine node sampling and DEALS sampling.
  - Open: Testing exactly exact node connectivity?
- *3rd Result* $(1+\varepsilon)$-approx every cut using $\tilde{O}(\varepsilon^{-2}n)$ space.
  - Basic Idea: Combine edge sampling and DEALS sampling.
  - Hypergraph Sparsifiers: Extends *Kogan, Krauthgamer* [ITCS 2015]


# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches


# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use Õ(n) space.


# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

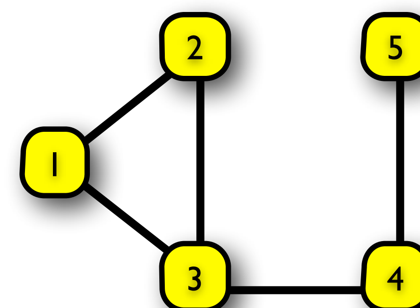
- ***Problem*** Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.
- ***Algorithm*** Construct $M\mathbf{a}_1, M\mathbf{a}_2, \ldots, M\mathbf{a}_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.


# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.

- *Algorithm* Construct $M\mathbf{a}_1, M\mathbf{a}_2, \ldots, M\mathbf{a}_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.



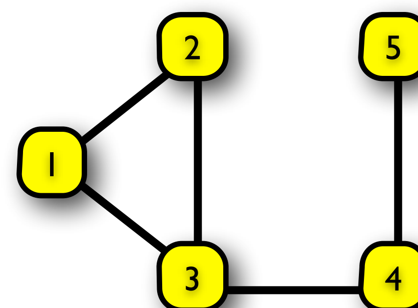

# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.

- *Algorithm* Construct $Ma_1, Ma_2, \ldots, Ma_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.

$$\begin{array}{cccccccccccc} & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\ \mathbf{a_1} = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \end{array}$$

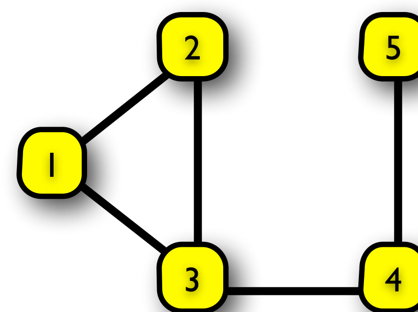



# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use Õ(n) space.
- *Algorithm* Construct $Ma_1, Ma_2, \ldots, Ma_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.

$$
\begin{array}{rcccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a_1} = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a_2} = ( & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0)
\end{array}
$$



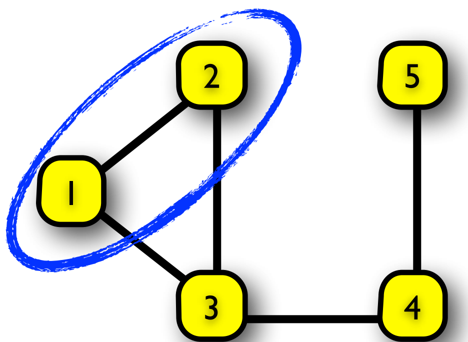

# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use Õ(n) space.

- *Algorithm* Construct $M\mathbf{a}_1, M\mathbf{a}_2, \ldots, M\mathbf{a}_n$ where M is $L_0$-sampling sketch and $\mathbf{a}_i$ encodes neighborhood of node i.

$$\begin{array}{rcccccccccc} & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\ \mathbf{a}_1 = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\ \mathbf{a}_2 = ( & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0) \end{array}$$



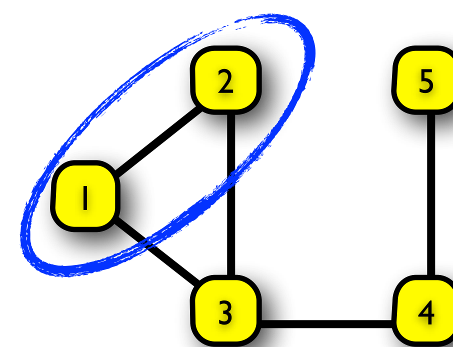

# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.

- *Algorithm* Construct $Ma_1, Ma_2, \ldots, Ma_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.

$$\begin{array}{rcccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_2 = ( & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_1 + \mathbf{a}_2 = ( & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0)
\end{array}$$

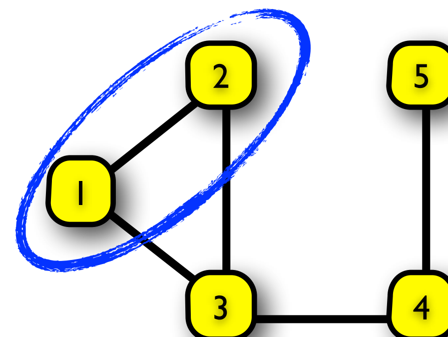



# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.
- *Algorithm* Construct $Ma_1, Ma_2, \ldots, Ma_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.

$$
\begin{array}{rcccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_2 = ( & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_1 + \mathbf{a}_2 = ( & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0)
\end{array}
$$



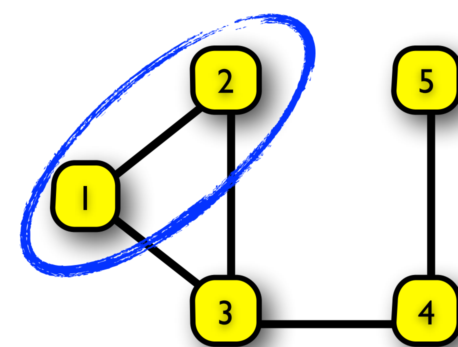

- *Lemma* Non-zero entries of $\sum_{i \in S} \mathbf{a}_i$ = edges across (S,V\S) and hence, $\sum_{i \in S} M\mathbf{a}_i = M(\sum_{i \in S} \mathbf{a}_i)$ yields random edge across (S,V\S).

# DEALS Sampling

## Direct-Edges-Add-$L_0$-Sketches

- *Problem* Sample edge across cut (S,V\S) where cut is specified at end of the stream. May use $\tilde{O}(n)$ space.
- *Algorithm* Construct $Ma_1, Ma_2, \ldots, Ma_n$ where M is $L_0$-sampling sketch and $a_i$ encodes neighborhood of node i.

$$
\begin{array}{rcccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = ( & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_2 = ( & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0) \\
\mathbf{a}_1 + \mathbf{a}_2 = ( & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0)
\end{array}
$$




- *Lemma* Non-zero entries of $\sum_{i \in S} \mathbf{a}_i$ = edges across (S,V\S) and hence, $\sum_{i \in S} M\mathbf{a}_i = M(\sum_{i \in S} \mathbf{a}_i)$ yields random edge across (S,V\S).
- *Application* Find spanning trees and edges in light cuts.

# Application to Node Connectivity...

# *Application to Node Connectivity...*

- *Simplified Result* Can answer queries of form "are u and v connected after removal of set of k nodes S" using $\tilde{O}(kn)$ space.

# *Application to Node Connectivity...*

- *Simplified Result* Can answer queries of form "are u and v connected after removal of set of k nodes S" using Õ(kn) space.
- *Algorithm*
  - Sample some edges and answer "no" iff there's no S-avoiding path between u and v amongst sampled edges.

# *Application to Node Connectivity…*

- *Simplified Result* Can answer queries of form "are u and v connected after removal of set of k nodes S" using $\tilde{O}(kn)$ space.
- *Algorithm*
  - Sample some edges and answer "no" iff there's no S-avoiding path between u and v amongst sampled edges.
  - How to sample: Pick each node with probability 1/k and find spanning forest on these nodes. Repeat $\tilde{O}(k^2)$ times.

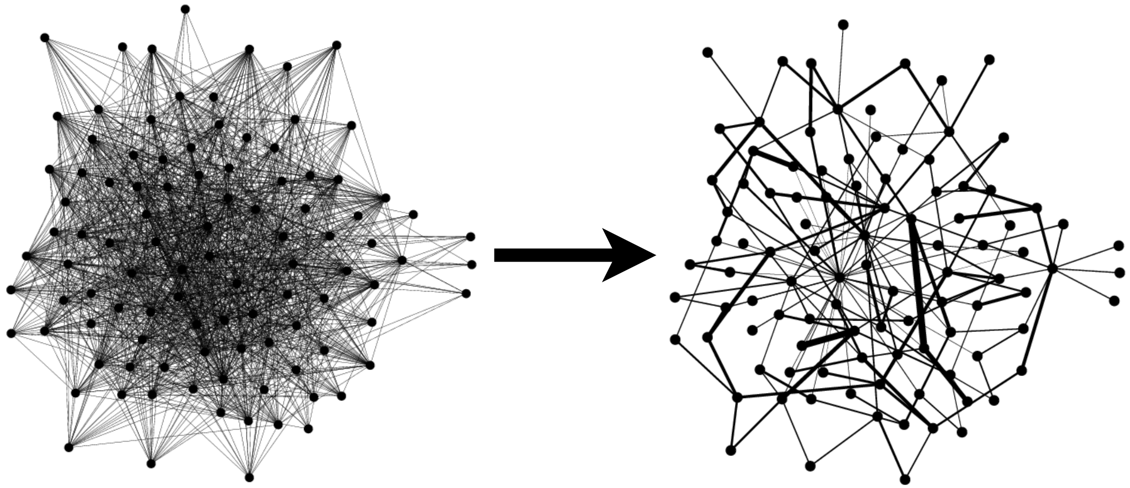
# *Application to Node Connectivity…*

- *Simplified Result* Can answer queries of form "are u and v connected after removal of set of k nodes S" using $\tilde{O}(kn)$ space.
- *Algorithm*
  - Sample some edges and answer "no" iff there's no S-avoiding path between u and v amongst sampled edges.
  - How to sample: Pick each node with probability 1/k and find spanning forest on these nodes. Repeat $\tilde{O}(k^2)$ times.
- *Analysis* Let $u$-$x_1$-$x_2$-….-$x_t$-$v$ be S-avoiding path in input graph.

# *Application to Node Connectivity…*

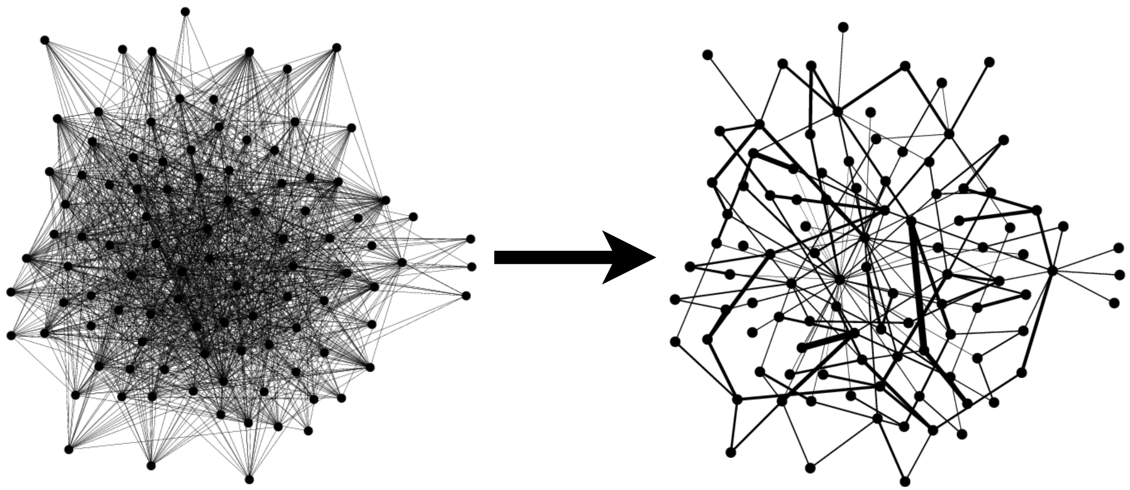
- *Simplified Result* Can answer queries of form "are u and v connected after removal of set of k nodes S" using $\tilde{O}(kn)$ space.
- *Algorithm*
  - Sample some edges and answer "no" iff there's no S-avoiding path between u and v amongst sampled edges.
  - How to sample: Pick each node with probability 1/k and find spanning forest on these nodes. Repeat $\tilde{O}(k^2)$ times.
- *Analysis* Let u-$x_1$-$x_2$-….-$x_t$-v be S-avoiding path in input graph.
  - Spanning forest on sampled nodes contains an S-avoiding path between $x_i$ and $x_{i+1}$ with prob. $p^2(1-p)^k \approx k^{-2}$. After $\tilde{O}(k^2)$ repeats we have S-avoiding path in E' with high probability.

# *Application to Cut Sparsification…*



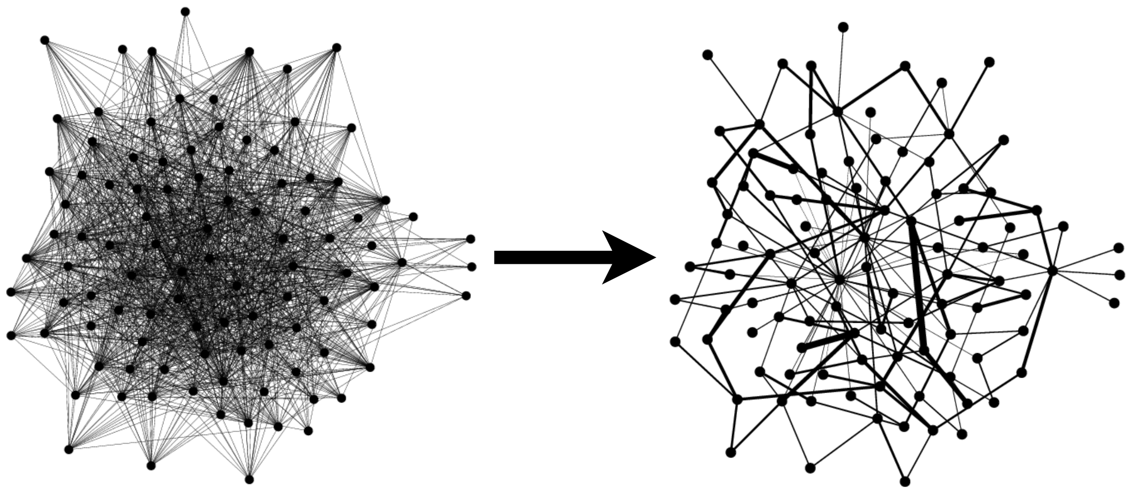
- ***Result*** Can $(1+\varepsilon)$ approximate all cuts using $O(\varepsilon^{-2}n)$ space.

# *Application to Cut Sparsification…*



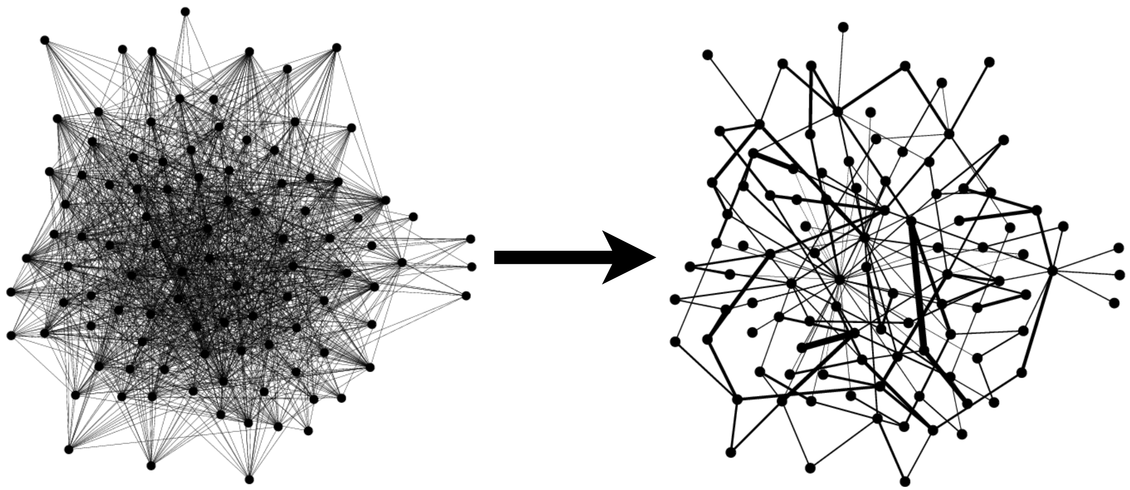
- ***Result*** Can $(1+\varepsilon)$ approximate all cuts using $O(\varepsilon^{-2}n)$ space.
- ***Basic Idea***
    - Sampling edges with probability $\geq (c\varepsilon^{-2} \log n)/\lambda_e$ preserves all cut sizes where $\lambda_e$ is the edge connectivity. *Fung et al.* [STOC 2011]

# *Application to Cut Sparsification...*



- *Result* Can $(1+\varepsilon)$ approximate all cuts using $O(\varepsilon^{-2}n)$ space.
- *Basic Idea*
  - Sampling edges with probability $\geq (c\varepsilon^{-2} \log n)/\lambda_e$ preserves all cut sizes where $\lambda_e$ is the edge connectivity. *Fung et al.* [STOC 2011]
  - Use DEALS sampling to pick all edges with $\lambda_e \leq 2c\varepsilon^{-2} \log n$ and sample each remaining edge with probably 1/2.

# *Application to Cut Sparsification…*



- *Result* Can $(1+\varepsilon)$ approximate all cuts using $O(\varepsilon^{-2}n)$ space.
- *Basic Idea*
  - Sampling edges with probability $\geq (c\varepsilon^{-2} \log n)/\lambda_e$ preserves all cut sizes where $\lambda_e$ is the edge connectivity. *Fung et al.* [STOC 2011]
  - Use DEALS sampling to pick all edges with $\lambda_e \leq 2c\varepsilon^{-2} \log n$ and sample each remaining edge with probably 1/2.
  - Recurse $O(\log n)$ times in parallel until we have sparse graph.

*Thanks!*

Graph Streaming Survey McGregor [SIGMOD Record 2014]

Vertex Connectivity and Sparsification. Guha, McGregor, Tench [PODS 2015]

Densest Subgraphs. McGregor, Tench, Vorotnikova, Vu [MFCS 2015]

Matching, Vertex Cover, Hitting Set.

Chitnis, Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova [TBA 2016]

# *Thanks!*

Graph Streaming Survey McGregor [SIGMOD Record 2014]

Vertex Connectivity and Sparsification. Guha, McGregor, Tench [PODS 2015]

Densest Subgraphs. McGregor, Tench, Vorotnikova, Vu [MFCS 2015]

Matching, Vertex Cover, Hitting Set.

Chitnis, Cormode, Esfandiari, Hajiaghayi, McGregor, Monemizadeh, Vorotnikova [TBA 2016]