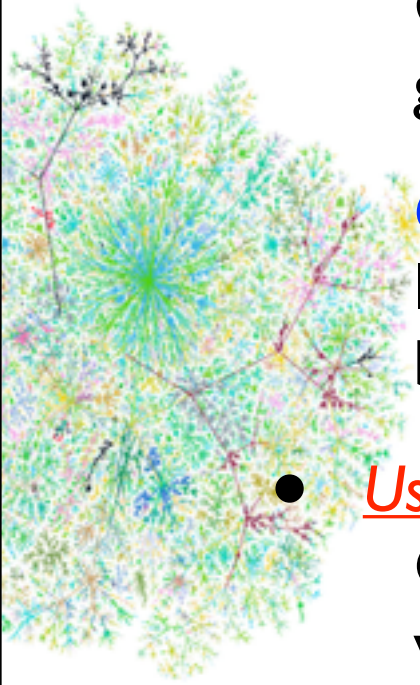
# Streaming, Sampling, Sketching, Small-Space Optimization

# Algorithmic Approaches *for* Analyzing Large Graphs

**Sudipto Guha**
*University of Pennsylvania*

**Andrew McGregor**
*University of Massachusetts*



- *Classic Big Graphs*

  Call graph ($5 \times 10^8$ nodes), web graph ($5 \times 10^9$ nodes), IP graph ($2^{32}$ nodes), social networks ($10^9$ nodes), ...

  *Challenge:* Can't use conventional algorithms on graphs this large. Often can't even store graph in memory. Graphs may be changing over time and data may be distributed.

- *Use Abstraction of Structure*

  Graphs are a natural way to encode structural information where we have data about both *basic entities* and their *relationships*. Examples include graphical networks, citation networks, protein interaction and metabolic networks.

- Want streaming, parallel, distributed algorithms…

- *Tutorial Goals and Caveats*

  Present some new algorithmic primitives for large graphs.

  Techniques are widely applicable; we'll be platform agnostic.

  Won't be comprehensive; will cherry pick illustrative results.

  Focus on arbitrary graphs rather than specific applications.

  Won't focus on proofs but will give basic outline when it helps convey why certain approaches are effective.

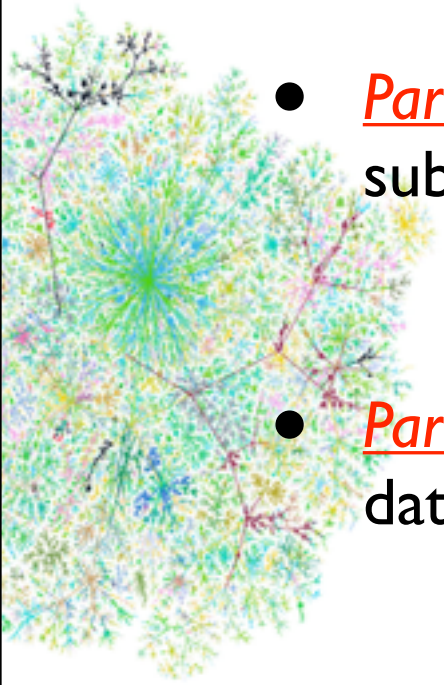
- *Resources*


Graph Stream Algorithms: A Survey

*Survey:* SIGMOD Record 2014
http://people.cs.umass.edu/~mcgregor/papers/13-graphsurvey.pdf

*Tutorial:* Slides and Bibliography
http://people.cs.umass.edu/~mcgregor/graphs

# Overview



- *Part I: Graph Sampling* Sampling for finding densest subgraphs, small matchings, triangles, spectral properties.

  *"Different sampling for different problems."*

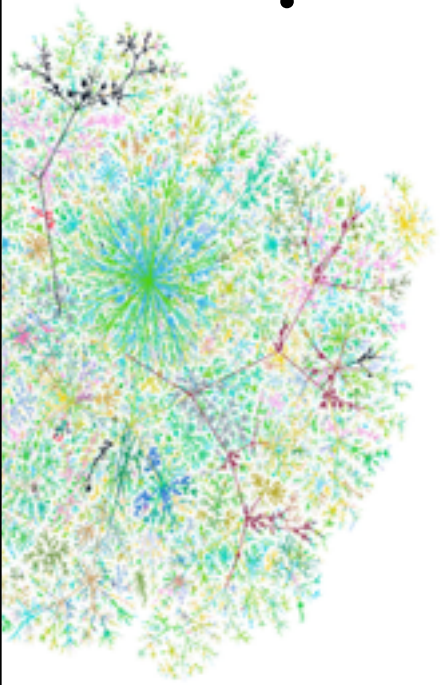
- *Part II: Graph Sketching* Dimensionality reduction for graph data. Examples include connectivity and sparsification.

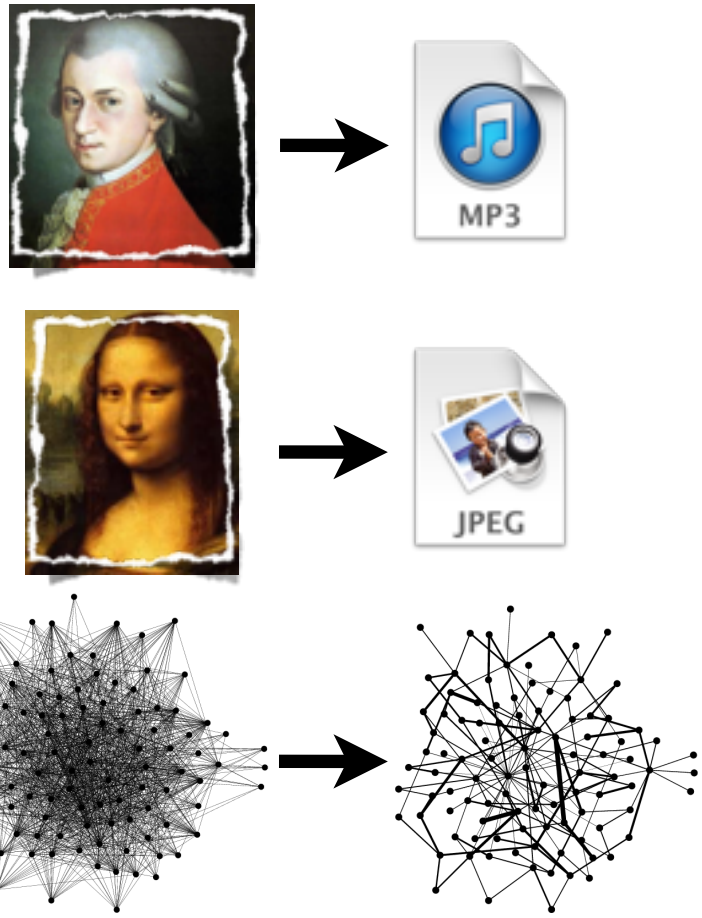
  *"Homomorphic compression: sketch first and then run algorithms on the sketched data."*

- *Part III: Small-Space Optimization* Combining sparsification and multiplicative weights for fast, small-space optimization. Examples include large matching and correlation clustering.

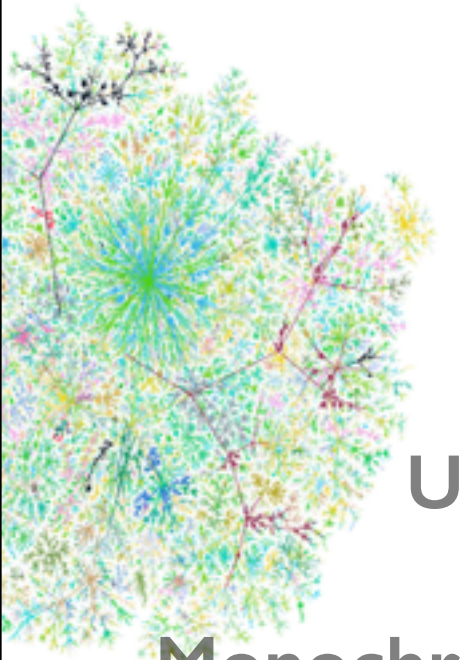
# Recurring Theme

? What's appropriate notion of lossy compression for graphs?






- If compression is easy, we get faster and more-space efficient algorithms by using existing algorithms on compressed graphs.



*Part I*

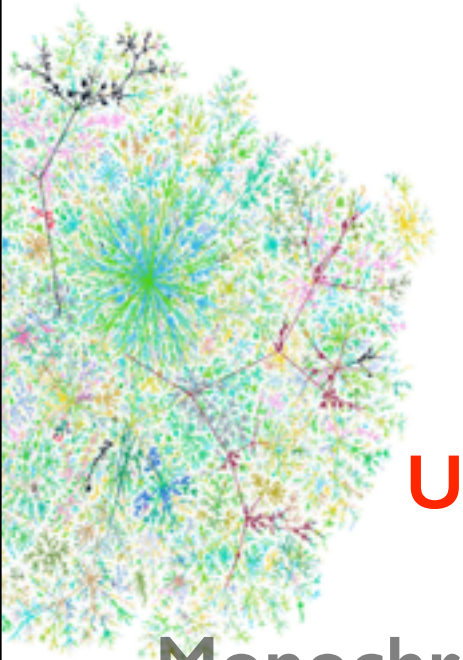
# Sampling

Uniform Sampling + Densest Subgraph

Snape Sampling + Matching

Monochromatic Sampling + Clustering Coefficient

Edge-Weighted Sampling + Cuts and Spectral Properties



*Part I*

# Sampling

**Uniform Sampling + Densest Subgraph**

Snape Sampling + Matching

Monochromatic Sampling + Clustering Coefficient

Edge-Weighted Sampling + Cuts and Spectral Properties

- Given a graph G, the *density* of a set of nodes $S \subset V$ is:

$$D_S = \frac{\#\text{ of edges with both endpoints in } S}{\#\text{ of nodes in } S}$$

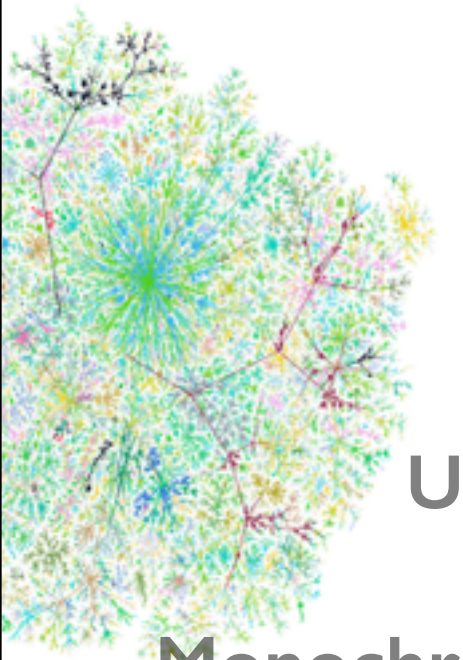
- *Problem* Estimating $\max_S D_S$ is a basic graph problem with numerous applications. Studied in a variety of models.

See tutorial *Gionis, Tsourakakis* [KDD 15]

- *Thm* Sample of $\tilde{O}(\varepsilon^{-2} n)$ edges uniformly and find the densest subgraph in sampled graph. This yields a $(1+\varepsilon)$-approx whp.

*McGregor et al.* [MFCS 15], *Esfandiari et al.* [15], *Mitzenmacher et al.* [KDD 15]

- *Proof Idea* Density of specific subgraph in sampled graph indicates whether actual density is large; if so, we get estimate of the density whp. Then union bound over $2^n$ subgraphs.



*Part I*

# Sampling

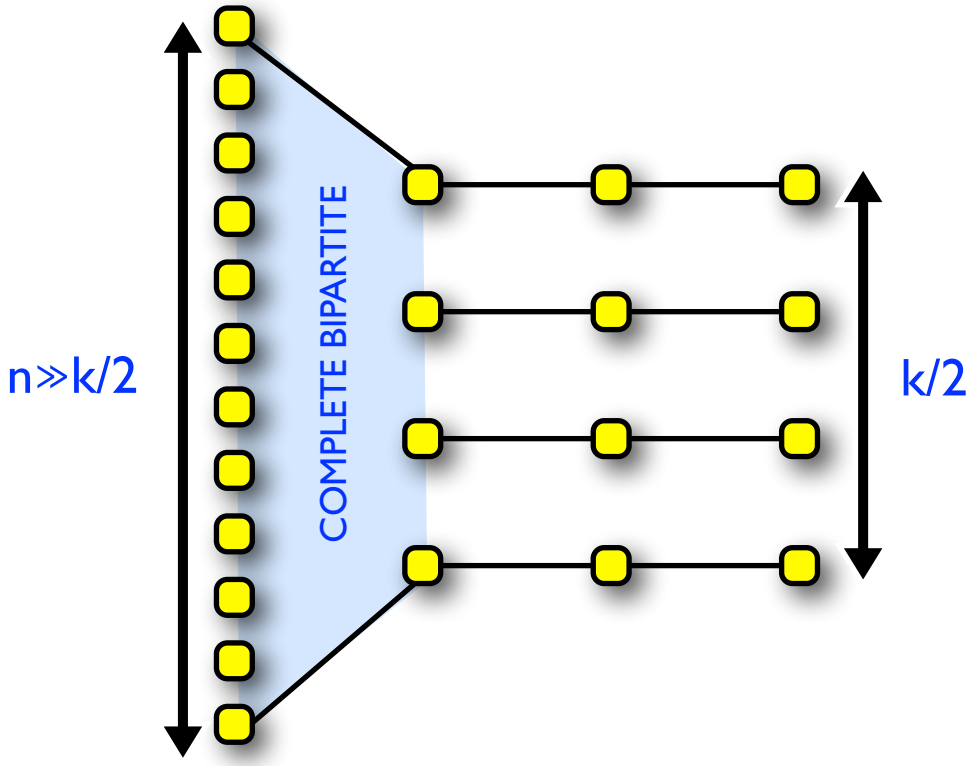
Uniform Sampling + Densest Subgraph

**Snape Sampling + Matching**

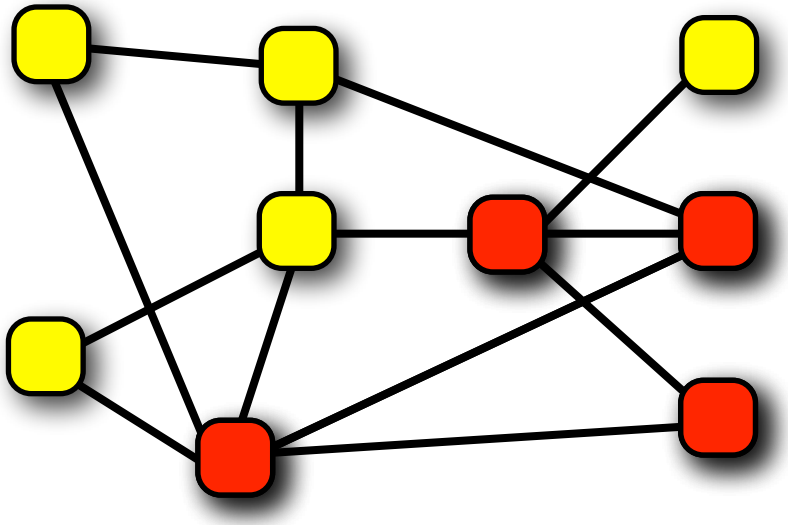
Monochromatic Sampling + Clustering Coefficient

Edge-Weighted Sampling + Cuts and Spectral Properties

- *Matching Problem* Find large set of edges such that no two edges share an endpoint.
- How many "samples" are needed to find a matching of size k?
- Sampling uniformly can be very inefficient…



- *SNAPE "Sample Nodes And Pick Edge" Sampling:*
  - Sample each node with probability $\Theta(k^{-1})$ and delete rest
  - Pick a random edge amongst those that remain.

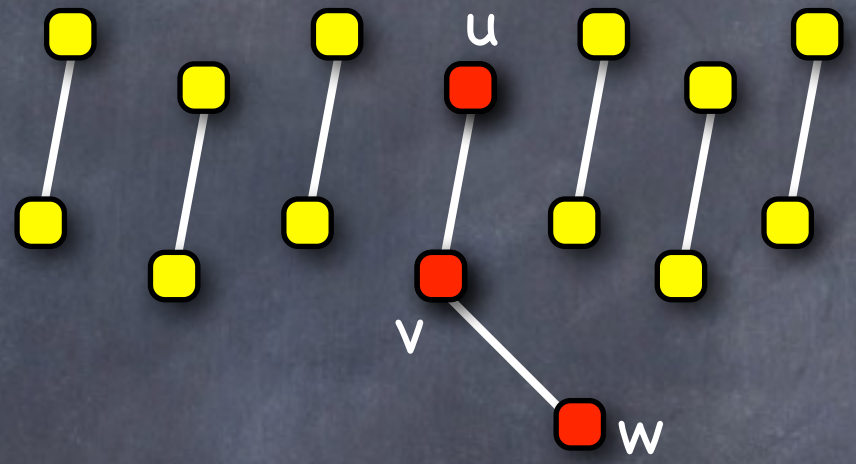


- *Theorem* If G has max matching size k, then $O(k^2 \log k)$ SNAPE samples will include a max matching from G.

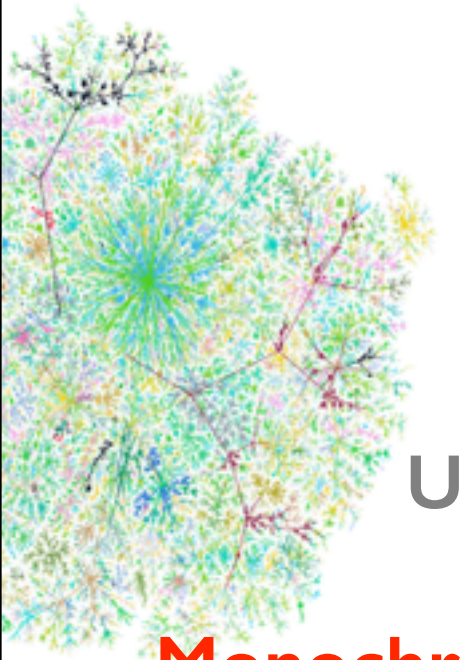
*Chitnis et al.* [SODA 16], *Bury, Schwiegelshohn* [ESA 15]

# Why SNAPE Sampling Works…

- Pick a maximum matching M of size k and pick arbitrary edge uv in this matching.




- With $\Omega(k^{-2})$ probability u and v are only endpoints of M that aren't deleted.
- Hence, when we pick one of the remaining edges it's either uv or another edge that's equally useful.
- Take $O(k^2 \log k)$ samples; apply analysis to all edges.



*Part I*

# Sampling

Uniform Sampling + Densest Subgraph

Snape Sampling + Matching

**Monochromatic Sampling + Clustering Coefficient**

Edge-Weighted Sampling + Cuts and Spectral Properties

- Given a graph G, the *global clustering coefficient* is

$$\kappa = \frac{3 \times \text{number of triangles}}{\text{number of length 2 paths}}$$

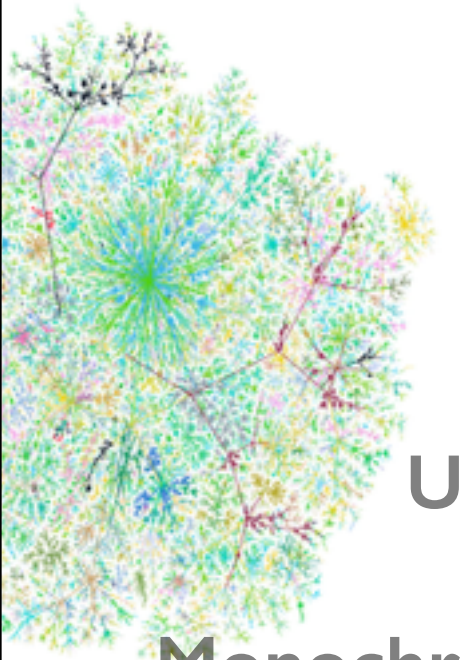
  A measure how much nodes tend to cluster together.

- *Monochromatic Sampling* Randomly color each node from a set of colors. Store all edges with monochromatic endpoints.

- *Thm* Can additively estimate κ from $\tilde{O}(\sqrt{n})$ samples.

  *Pagh, Tsourakakis* [IPL 12], *Jha, Seshadhri, Pinar* [KDD 15]

- *Proof Idea* Compute expectation and variance of number of triangles amongst sampled edges and apply Chebyshev bound.



*Part I*

# Sampling

Uniform Sampling + Densest Subgraph

Snape Sampling + Matching

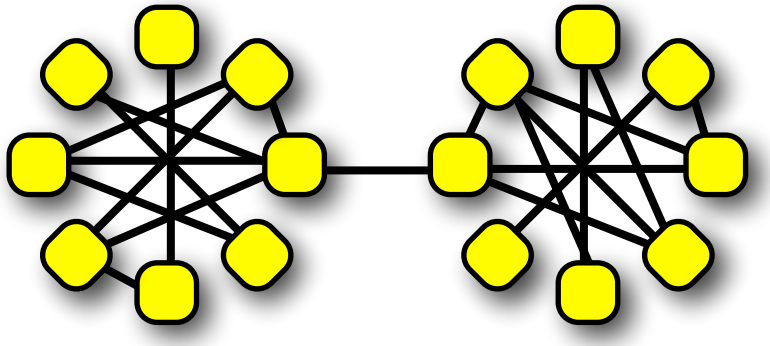
Monochromatic Sampling + Clustering Coefficient

**Edge-Weighted Sampling + Cuts and Spectral Properties**

- *Defn* A *sparsifier* of graph G is a weighted subgraph H with:

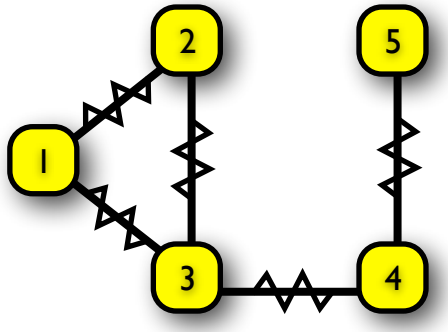
  $\forall$ cuts: "size of cut in G" = $(1\pm\varepsilon)$ "size of cut in H"

- *Basic Approach* Sample each edge uv with probability $p_{uv}$ and reweight by $1/p_{uv}$. Probabilities depend on edge properties…



- *Thm* If $p_{uv} \approx \varepsilon^{-2}/\lambda_{uv}$ or $p_{uv} \approx \varepsilon^{-2} r_{uv}$ then result is sparsifier with $\tilde{O}(\varepsilon^{-2}\, n)$ edges. *Fung et al.* [STOC 11], *Spielman, Srivastava* [STOC 08]

$\lambda_{uv}$ is the min number of edges whose removal disconnects u and v




$r_{uv}$ is potential difference when unit of flow injected at u and extracted at v

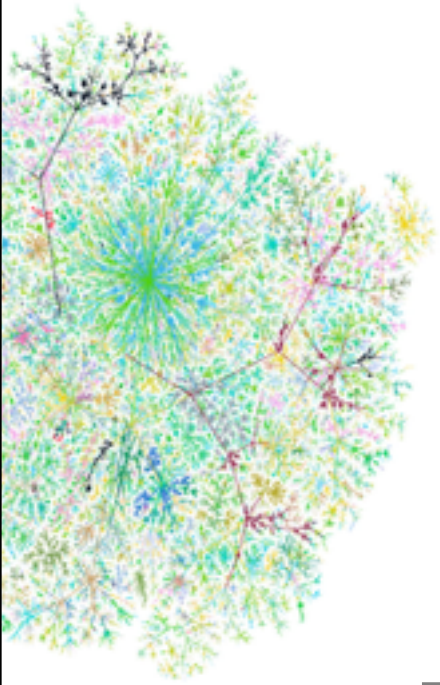
- *Simpler Thm* If min-cut is $\gg \varepsilon^{-2} \log n$ then $p_e = 1/2$ works.

# Proof Idea of Simpler Theorem ...

- Lemma (Chernoff) Let k′ be the number of edges that were sampled across some cut of size k. Then

$$\Pr[k' = (1 \pm \varepsilon)k/2] < \exp(-\varepsilon^2 k/6)$$

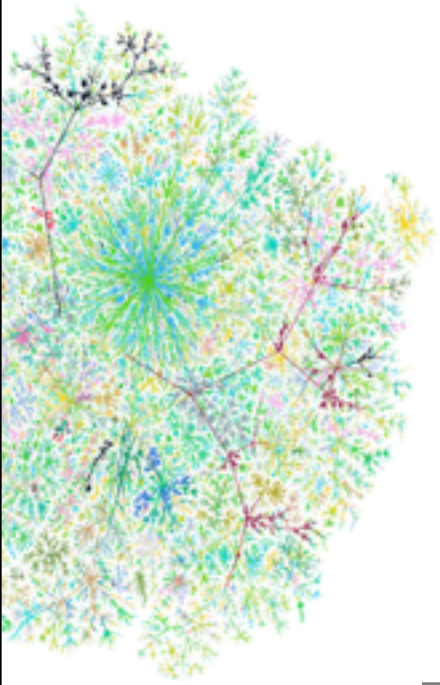
- Lemma (Karger) The number of cuts with k edges is $< \exp(2k \log n / \lambda)$ where $\lambda$ is size of min-cut.
- Result then follows by substituting bound for $\lambda$ and applying union bound over all cuts.



*Part II*

# Sketching

What is sketching?
Surprising connectivity example
Revisiting graph cuts and sparsification



*Part II*

# Sketching

**What is sketching?**

Surprising connectivity example

Revisiting graph cuts and sparsification

- *Random linear projection* M: $\mathbb{R}^N \rightarrow \mathbb{R}^D$ where $D \ll N$ that preserves properties of any $v \in \mathbb{R}^N$ with high probability.

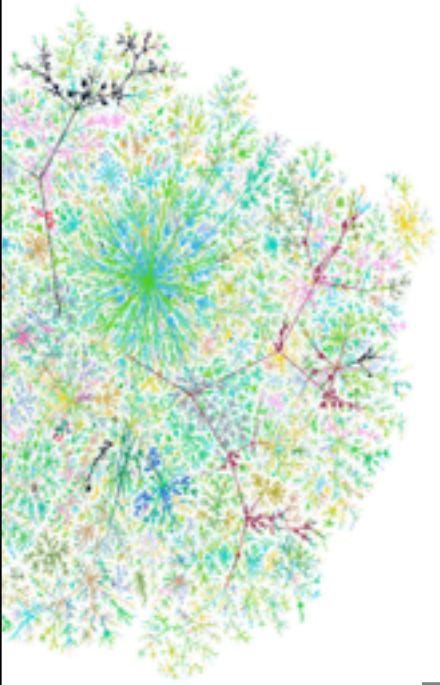
$$\begin{bmatrix} & M & \end{bmatrix} \begin{bmatrix} \\ v \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} Mv \\ \\ \end{bmatrix} \longrightarrow \text{answer}$$

- *Many results* for numerical statistics and geometric properties... *extensive theory* with connections to hashing, compressed sensing, dimensionality reduction, metric embeddings... *widely applicable* since parallelizable and suitable for stream processing.

- *Example "$l_0$ Sampling" Sketch* Can be used to sample uniformly from non-zero entries of the vector where D=polylog(N).

*Jowhari, Saglam, Tardos* [PODS 11]

? *Question* What about analyzing massive graphs via sketches?



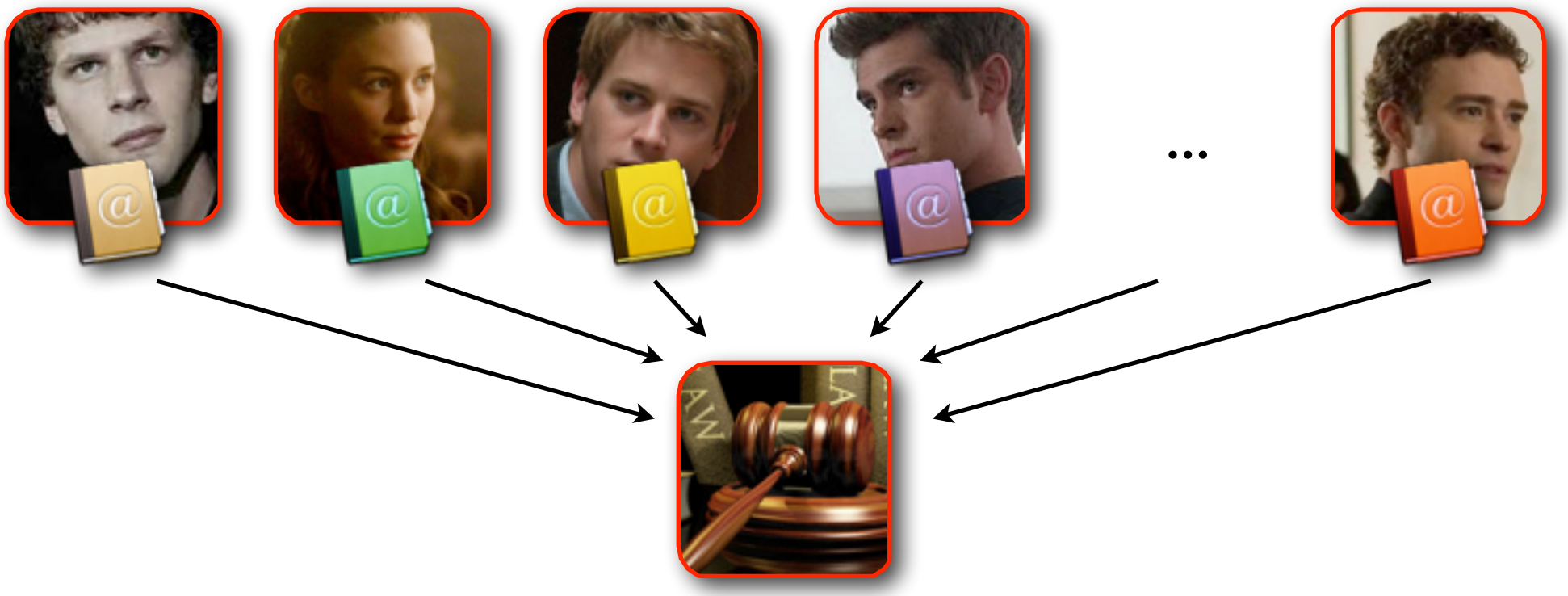
*Part II*

# Sketching

What is sketching?

**Surprising connectivity example**

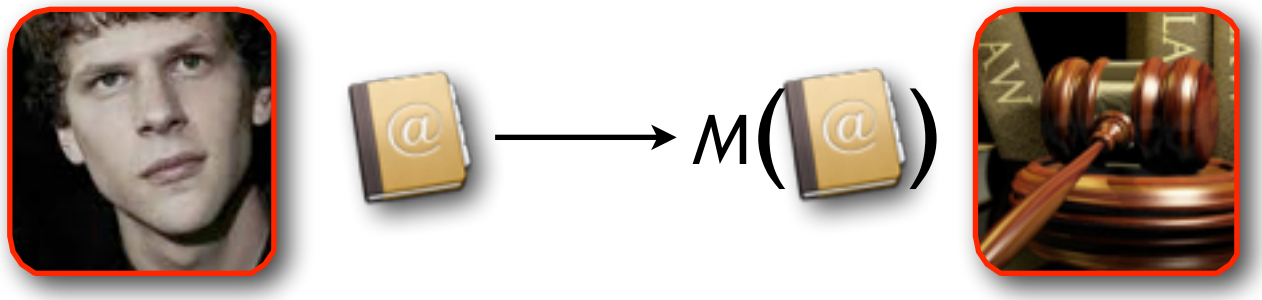
Revisiting graph cuts and sparsification



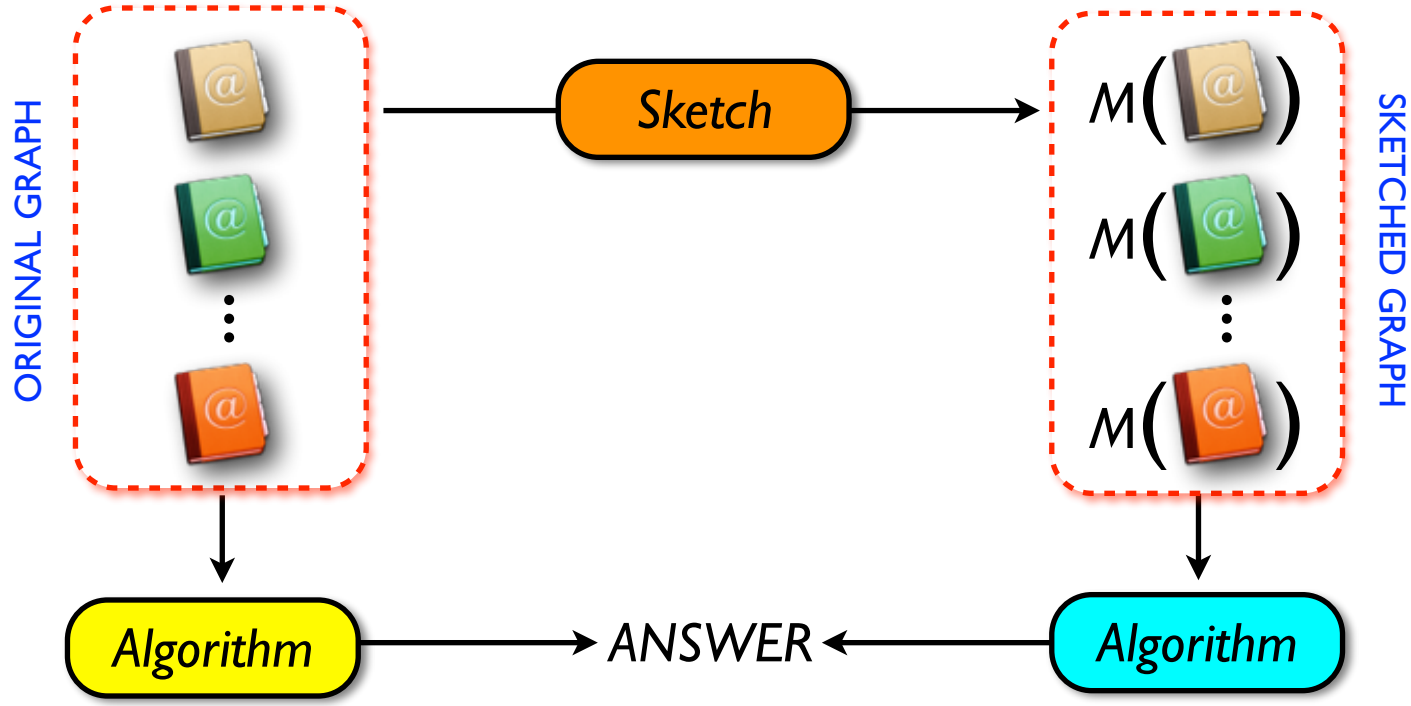
- *Communication Problem* n players each have a book listing their friends. Simultaneously, they each send a message to a central player who deduces if underlying graph is connected.
- *Thm* O(polylog n) bit message from each player suffices.

*Ahn, Guha, McGregor* [SODA 12]

- *Can't be possible!* What if there's a *bridge* (u,v) in the graph, i.e., a friendship that is critical to ensuring the graph is connected.
- It *appears* like at least one player needs to send Ω(n) bits.
  - Central player needs to know about the special friendship.
  - Participant doesn't know which friendships are special.
  - Participants may have Ω(n) friends.

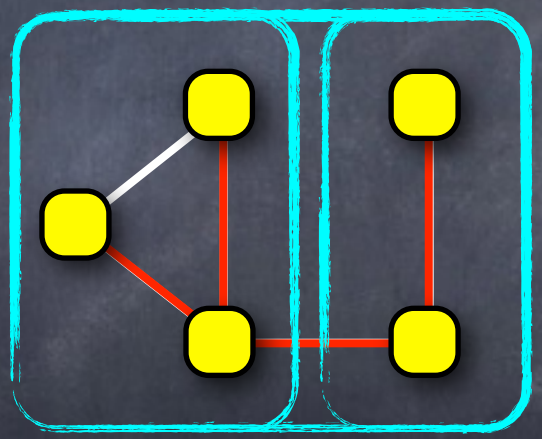



- Players send carefully-designed sketches of address books.
- *Homomorphic Compression* Instead of running algorithm on original data, run algorithm on sketched data.



# Ingredient 1: Basic Algorithm

- Algorithm (Spanning Forest):
  - For each node: pick incident edge
  - For each connected comp: pick incident edge
  - Repeat until no edges between connected comp.

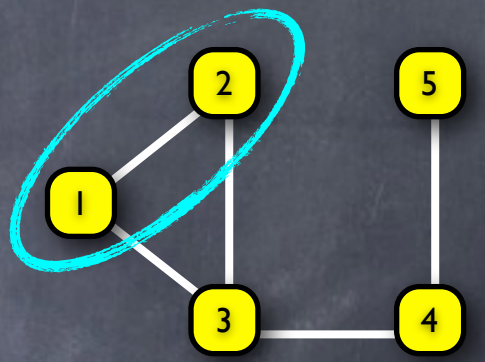


- Lemma After O(log n) rounds selected edges include spanning forest.

# Ingredient 2: Sketching Neighborhoods

- For node i, let $a_i$ be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.

$$
\begin{array}{rc}
 & \begin{matrix}\{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\}\end{matrix} \\
\mathbf{a}_1 = & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{a}_2 = & \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{a}_1 + \mathbf{a}_2 = & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}
$$



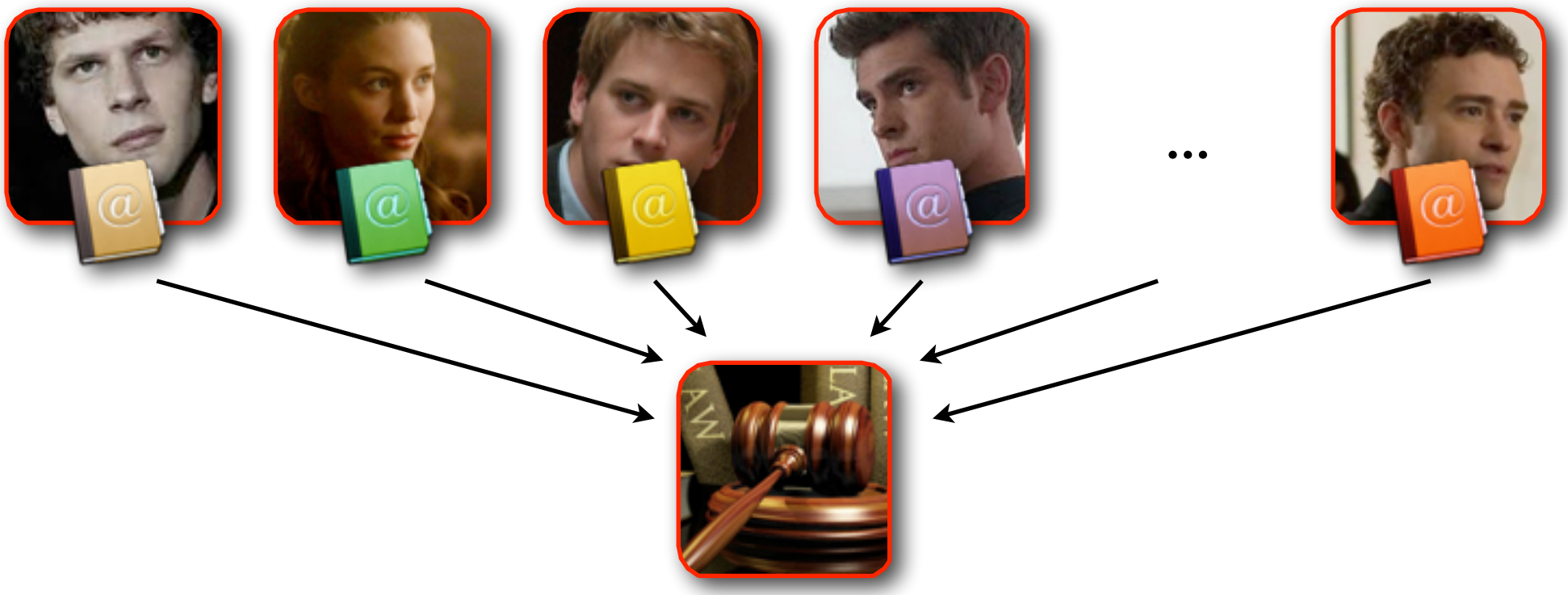

- Lemma For any subset of nodes $S \subset V$, non-zero entries of $\sum_{j \in S} a_j$ are edges across cut $(S, V \setminus S)$
- Player j sends $M(a_j)$ where M is "$l_0$ sampling" sketch.

# Recipe: Sketch & Compute on Sketches

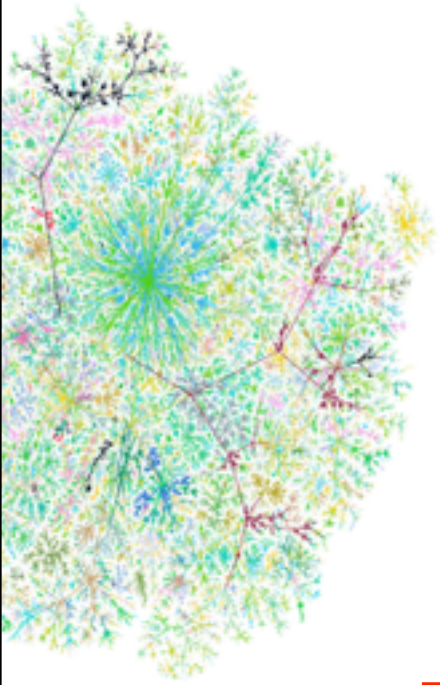
- Player with Address Books: Player j sends $Ma_j$
- Central Player: "Runs Algorithm in Sketch Space"
  - Use $Ma_j$ to get incident edge on each node j
  - For i=2 to log n:
    - To get incident edge on component $S \subset V$ use:

$$\sum_{j \in S} M\mathbf{a}_j = M(\sum_{j \in S} \mathbf{a}_j) \longrightarrow \text{non-zero elt. of } \sum_{j \in S} a_j = \text{edge across cut}$$

Detail: Actually each player sends log n independent sketches $M_1a_j$, $M_2a_j$, ... and central player uses $M_ia_j$ when emulating $i^{th}$ iteration of the algorithm.



- *Thm* O(polylog n) bit message from each player suffices.
- *Various extensions* E.g., with $\tilde{O}(k)$ bit messages, can find all edges that participate in cuts of size less than k.



*Part II*

# Sketching

What is sketching?

Surprising connectivity example

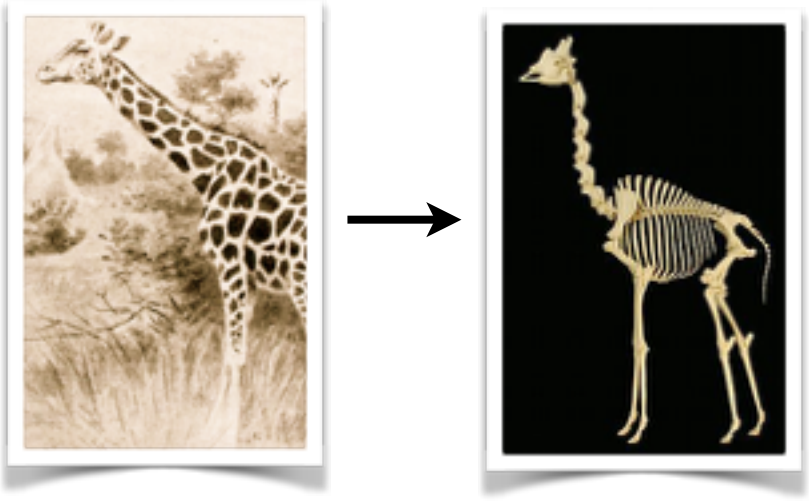
**Revisiting graph cuts and sparsification**

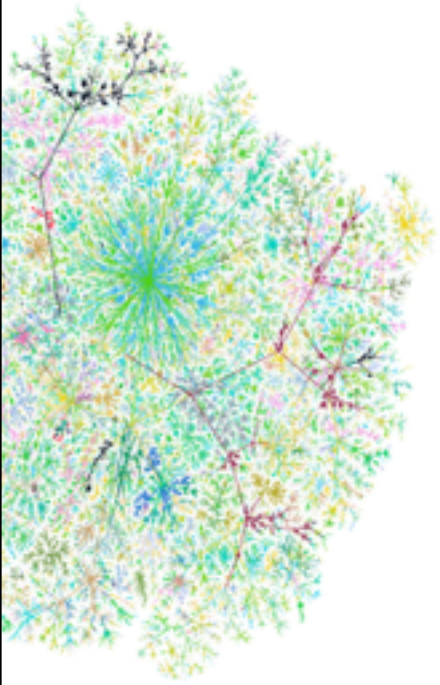
- *Thm* $O(\varepsilon^{-2}$ polylog n) bit messages suffice for central player to construct sparsifier and approx all graph cuts.

  *Guha, McGregor, Tench* [PODS 15], Kapralov et al. [STOC 14]

- *Main Ideas:*

  1. For a graph G, can find set all edges in small cuts.
  2. For large cuts, suffices to sample edges with prob. 1/2.
  3. So, sparsifying G reduces to sparsifying sampled graph G'.
  4. To sparsify G' recurse… Can do recursion in parallel.





*Thanks! Over to Sudipto…*