Differential Privacy for Graphs and Social Networks

Sofya Raskhodnikova

Penn State University,

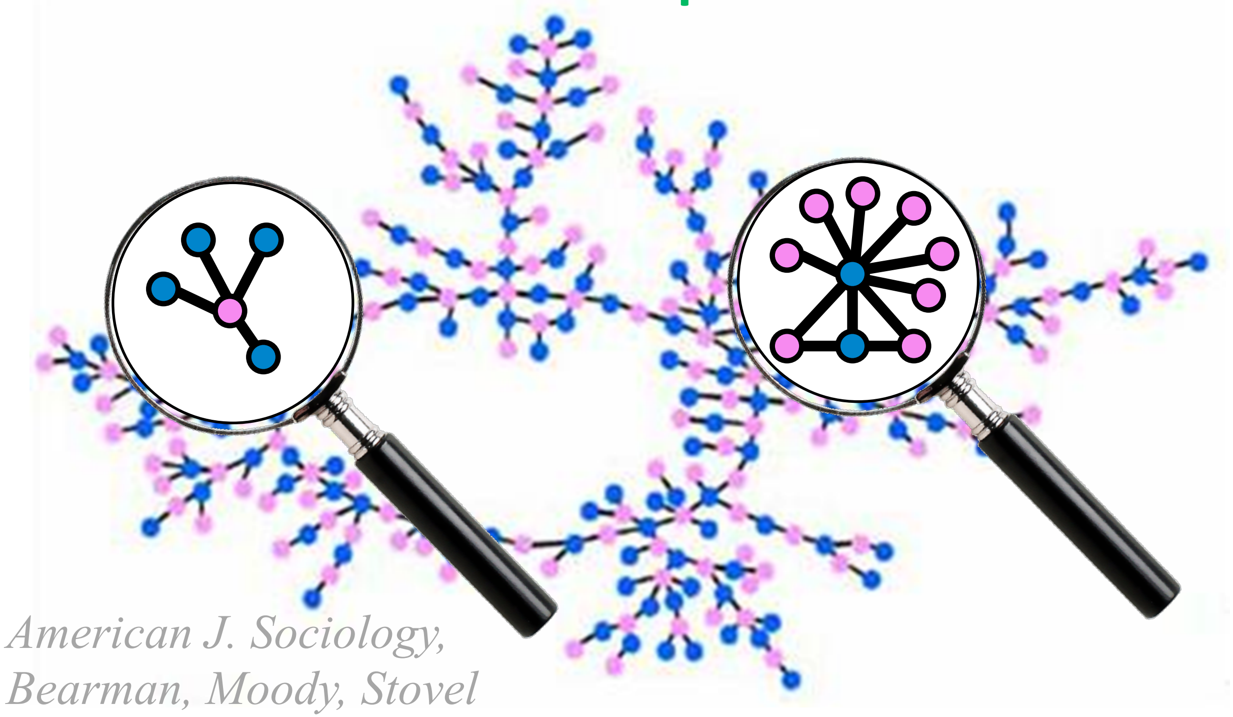
on sabbatical at BU for 2013-2014 ***privacy year***

Publishing information about graphs

Many types of data can be represented as graphs

- **"Friendships" in online social network**
- **Financial transactions**
- **Email communication**
- **Health networks (of doctors and patients)**
- **Romantic relationships**

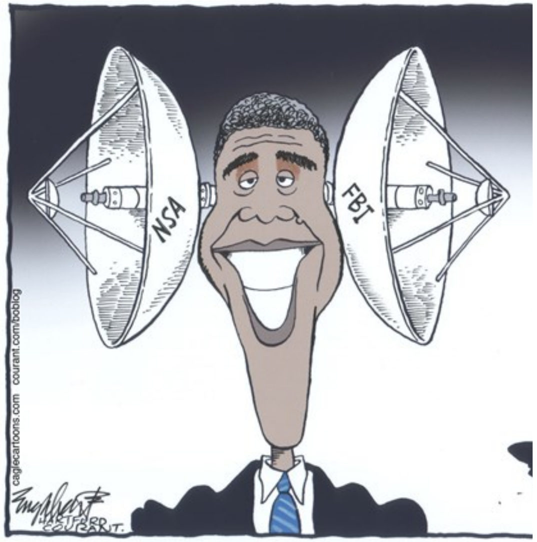
image source http://community.expressor-software.com/blogs/mtarallo/36-extracting-data-facebook-social-graph-expressor-tutorial.html



American J. Sociology, Bearman, Moody, Stovel

Privacy is a big issue!

Who'd want to de-anonymize a social network graph?



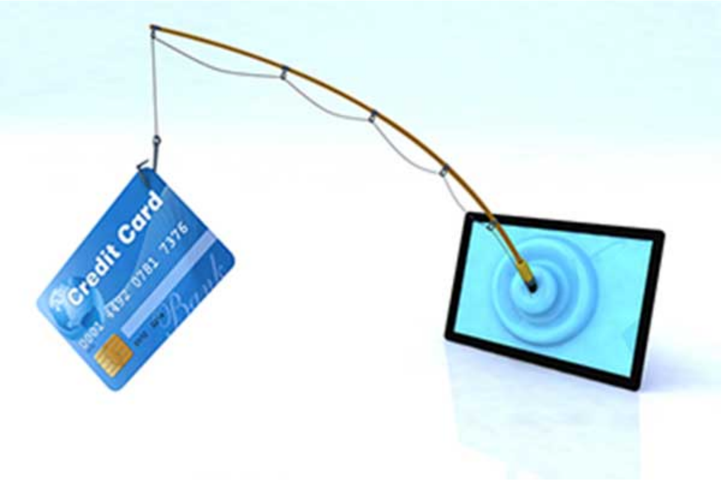


image sources © Depositphotos.com/fabioberti.it, Andrew Joyner, http://dukeromkey.com/

Some published attacks

- Social networks

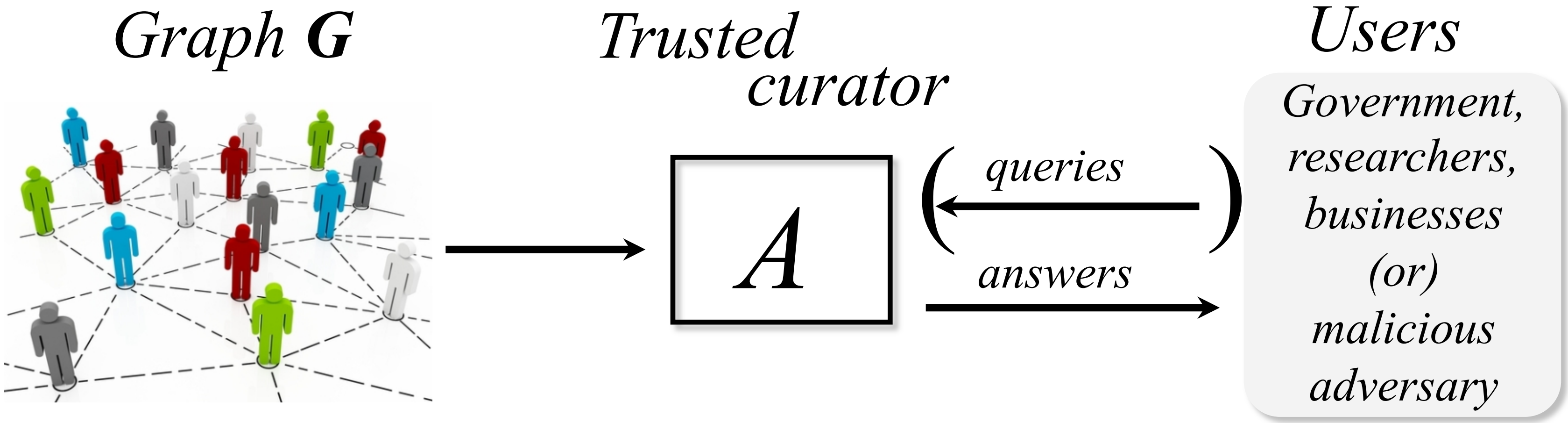
  [Backstrom Dwork Kleinberg 07,
  Narayanan Shmatikov 09, Narayanan Shi Rubinstein 12]

- Computer networks

  [Coull Wright Monrose Collins Reiter 07,
  Ribeiro Chen Miklau Townsley 08]

Can reidentify individuals based on external sources.

Differential privacy (for graph data)



Graph **G**
Trusted curator
A
queries
answers
Users
Government, researchers, businesses (or) malicious adversary

Differential privacy [Dwork McSherry Nissim Smith 06]

An algorithm A is ϵ-differentially private if

for all pairs of **neighbors** G, G' and all sets of answers **S**:

$$Pr[A(G)\in S] \leq e^{\epsilon} Pr[A(G')\in S]$$

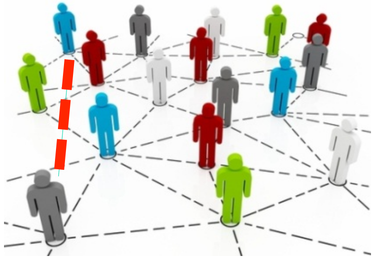
image source http://www.queticointernetmarketing.com/new-amazing-facebook-photo-mapper/

Two variants of differential privacy for graphs

- **Edge** differential privacy

***G*:**

***G′*:**

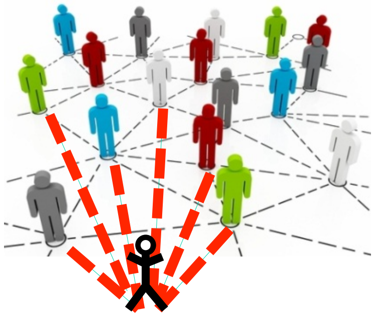


Two graphs are **neighbors** if they differ in ***one edge***.

- **Node** differential privacy

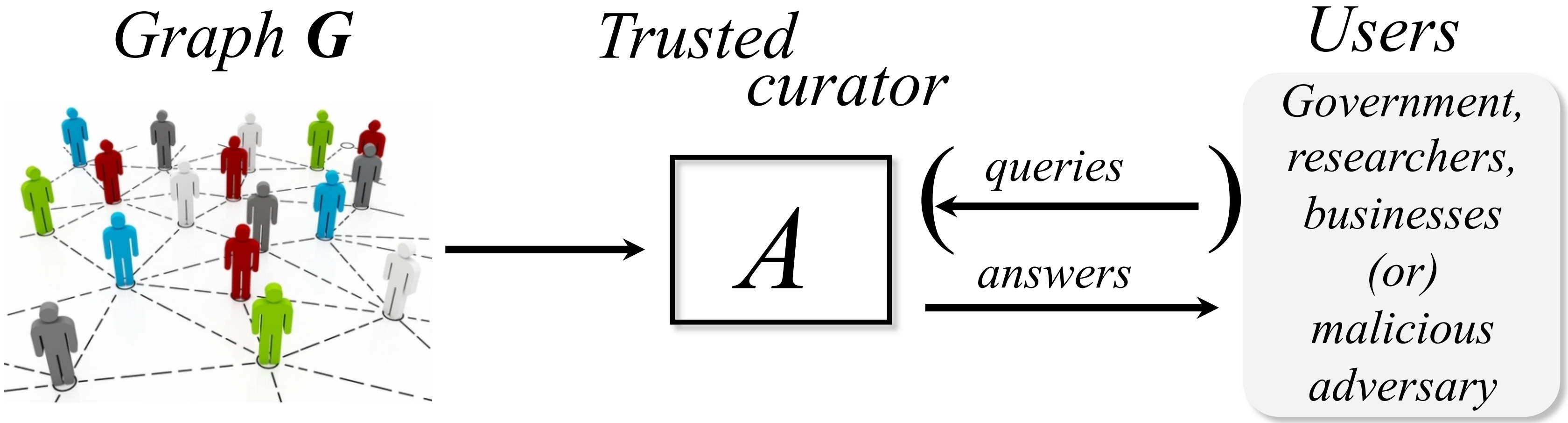
***G*:**

***G′*:**



Two graphs are **neighbors** if one can be obtained from the other by deleting ***a node and its adjacent edges***.

Differentially private analysis of graphs



*Graph **G***

Trusted curator

A

queries

answers

Users

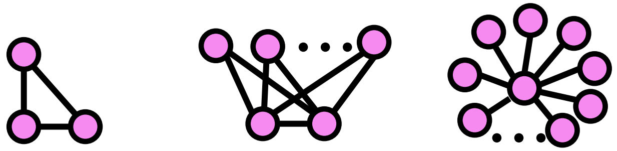
Government, researchers, businesses (or) malicious adversary

- **Two conflicting goals:** utility and privacy
  - Impossible to get both in the worst case
- **Want:** differentially private algorithms that are accurate on realistic graphs
  - ***differentially private*** (***for all*** *graphs)*
  - ***accurate for a subclass*** *of graphs*

image source http://www.queticointernetmarketing.com/new-amazing-facebook-photo-mapper/

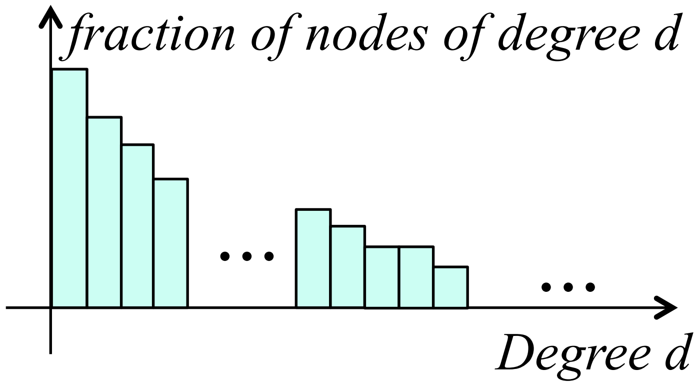
Graph statistics

- Number of edges
- Counts of small subgraphs



(e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

- Degree distribution



- Cut sizes
- Distance to nearest graph with a certain property
- Joint degree distribution

Edge differentially private algorithms pre-2013:

graph statistics and techniques

- **number of triangles**, **MST cost** [Nissim Raskhodnikova Smith 07]
  - Smooth sensitivity
- **degree distribution** [Hay Rastogi Miklau Suciu 09, Hay Li Miklau Jensen 09, Karwa Slavkovic 12, Kifer Lin 13]
  - Global sensitivity and postprocessing
- **small subgraph counts** [Karwa Raskhodnikova Smith Yaroslavtsev 11]
  - Smooth sensitivity; Propose-Test-Release [Dwork Lei 09]
- **cuts**
  - Random projections, global sensitivity [Blocki Blum Datta Sheffet 12]
  - Iterative updates [Hardt Rothblum 10, Gupta Roth Ullman 12]
- **Kronecker graph model parameters** [Mir Wright 12]
  - Postprocessing of [KRSY'11]

Other definitions

Edge private against Bayesian adversary (*weaker* privacy)

- **small subgraph counts** [Rastogi Hay Miklau Suciu 09]

Node zero-knowledge private (*stronger* privacy than DP)

- **average degree, distances to nearest connected, Eulerian, cycle-free graphs (privacy only for bounded-degree graphs)** [Gehrke Lui Pass 12]
  - Sublinear-time algorithms + global sensitivity

Today: 2013

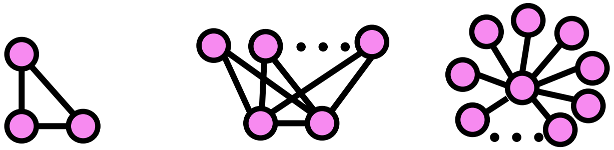
New techniques [Blocki Blum Datta Sheffet 13, Kasiviswanathan Nissim Raskhodnikova Smith 13, Chen Zhou 13, Raskhodnikova Smith]

- achieve node differential privacy
- give better edge differentially private algorithms

- Guarantees for resulting algorithms
  - **node differentially private** ***for all*** graphs
  - **accurate** ***for a subclass*** of graphs, which includes
    - **graphs with sublinear (not necessarily constant) degree bound**
    - **graphs where the tail of the degree distribution is not too heavy**
    - **dense graphs**
  - good performance in experiments ***on real graphs*** for simple statistics

Today

- Node differentially private algorithms for releasing
  - number of edges
  - counts of small subgraphs
  - degree distribution



Today

- New techniques
  1. **Truncation + smooth sensitivity** [BBDS’13, KNRS’13]
  2. **Lipschitz extensions** [BBDS’13, KNRS’13]
  3. **Recursive mechanism** [Chen Zhou 13]

- Unifying idea: ``projections’’ on ``graphs’’ with low sensitivity
  - Generic reduction to privacy over bounded-degree graphs

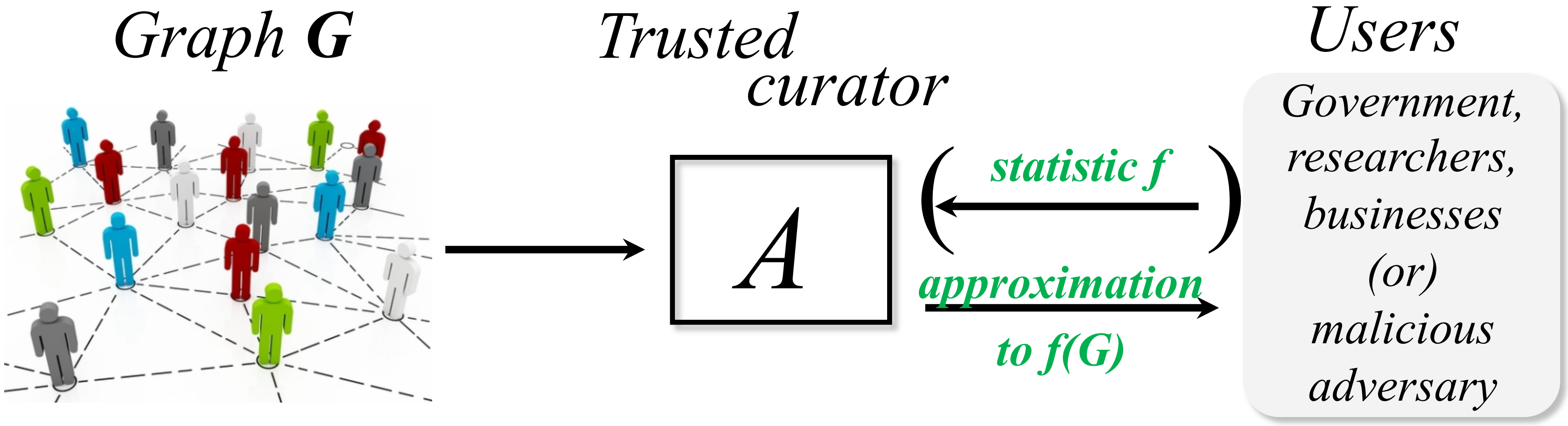
    truncation + smooth sensitivity [BBDS’13,KNRS’13]
  - Releasing number of edges and subgraph counts

    Lipschitz extensions via max flow and LP [KNRS’13]
  - Releasing degree distribution

    Lipschitz extension via convex programming [Raskhodnikova Smith]
  - Releasing subgraph counts

    Recursive mechanism [Chen Zhou 13]

Basic question



*Graph **G***

Trusted curator

A

statistic f

approximation to f(G)

Users

Government, researchers, businesses (or) malicious adversary

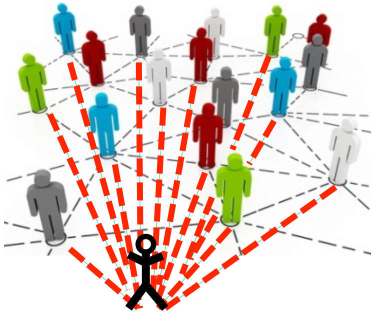
How accurately

can an ϵ-differentially private algorithm release f(G)?

Challenge for node privacy: high sensitivity

- **Global sensitivity** of a function f is

$\boldsymbol{\partial f} = \max┬(\mathbf{node})\mathbf{neighbor}s\ G,G' \ |f(G) - f(G\uparrow ')|$

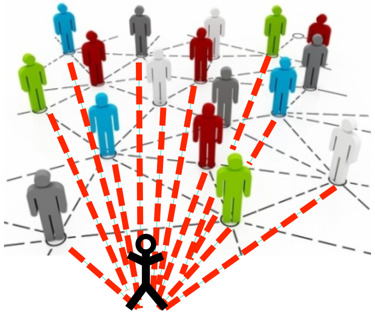


- **Examples:**

➢ $f\downarrow -$ (G) is the number of edges in G. $\boldsymbol{\partial f}\downarrow - \ = n.$

➢ $f\downarrow\triangle$ (G) is the number of triangles in G. $\boldsymbol{\partial f}\downarrow\triangle = (\boldsymbol{n}¦\mathbf{2}\ ).$

Challenge for node privacy: high sensitivity



- **Global sensitivity** of a function f is

  $\boldsymbol{\partial f} = \max_{(\textbf{node})\textbf{neighbor}s\ G,G'} |f(G)-f(G')|$

- **Local sensitivity,** $\max_{G'\,:\,\textbf{neighbor of } G} |f(G)-f(G')|$**,** is also high.

- New measure of sensitivity [Chen Zhou 13]

  Down sensitivity is $\max_{G'\,:\textbf{subgraph neighbor of } G} |f(G)-f(G')|$.

Idea: project onto graphs with low down sensitivity.

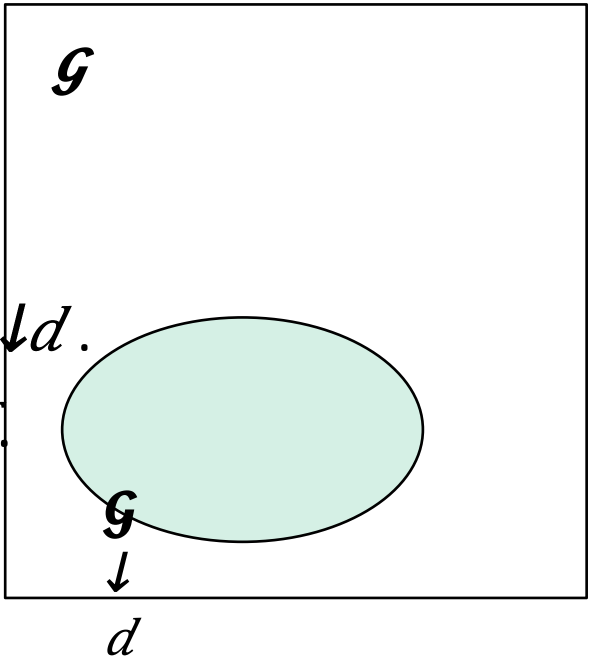
“Projections” on graphs of small degree [BBDS’13,KNRS’13]

Let $\mathcal{G}$ = family of all graphs,

$\mathcal{G}_d$ = family of graphs of degree $\leq d$.

Notation. ∂f = **global sensitivity of** f over $\mathcal{G}$.

$\partial_d f$ = **global sensitivity of** f over $\mathcal{G}_d$.



Observation. $\partial_d f$ is low for many useful f.

Examples:

- $\partial_d f_{-} = d$ **(compare to $\partial f_{-} = n$)**
- $\partial_d f_{\triangle} = \binom{d}{2}$ **(compare to $\partial f_{\triangle} = \binom{n}{2}$)**

Goal: privacy for all graphs

Idea: ``Project’’ on graphs in $\mathcal{G}_d$ for a carefully chosen d << n.

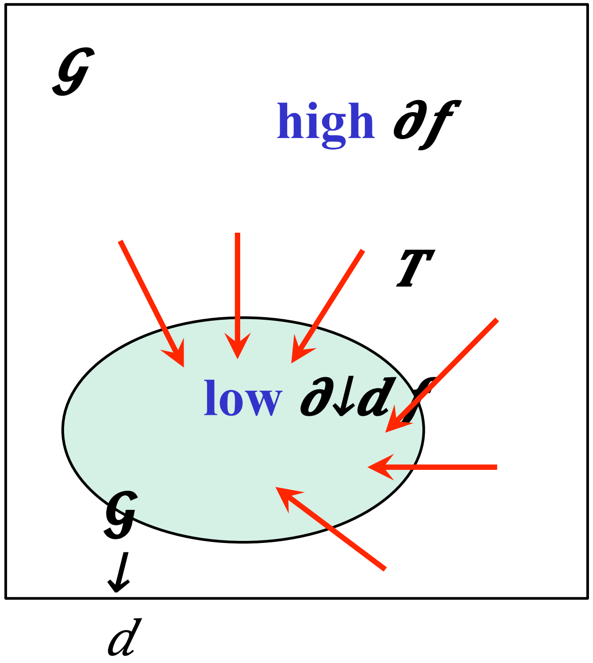
Method 1

Truncation + smooth sensitivity

Method 1: reduction to privacy over $\mathcal{G}\downarrow d$ [KNRS'13]

Input: Algorithm B that is node-DP over $\mathcal{G}\downarrow d$

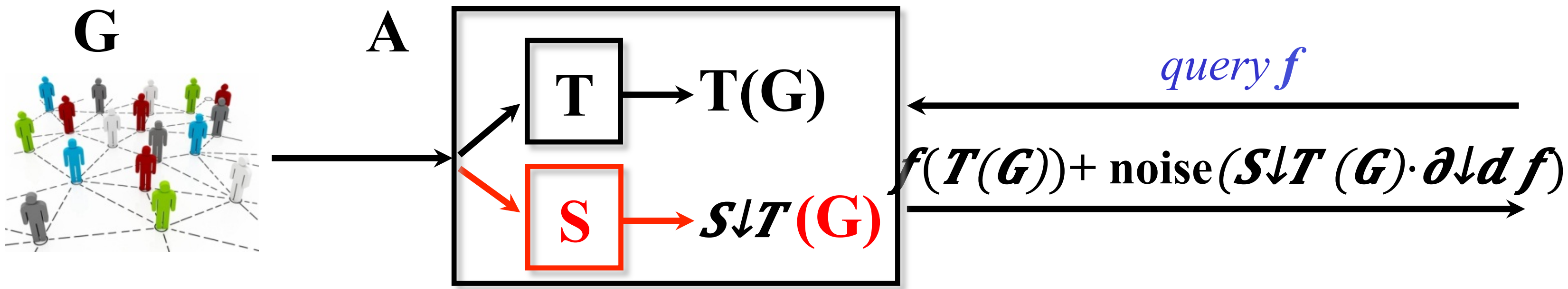
Output: Algorithm A that is node-DP over $\mathcal{G}$,
has accuracy similar to B on "nice" graphs



- Time(A) = Time(B) + O(m+n)
- Reduction works for all functions f

How it works: Truncation T(G) outputs G with nodes of degree $>d$ removed.

- Answer queries on T(G) instead of G
  - via Smooth Sensitivity framework [NRS'07]
  - via finding a DP upper bound ℓ on local sensitivity [Dwork Lei 09, KRSY'11] and running any algorithm that is (ϵ/ℓ)-node-DP over $\mathcal{G}\downarrow d$


G
A
T
T(G)
S
$S\downarrow T$ (G)
query f
$f(T(G))$+ noise$(S\downarrow T\,(G)\cdot\partial\downarrow d\, f)$

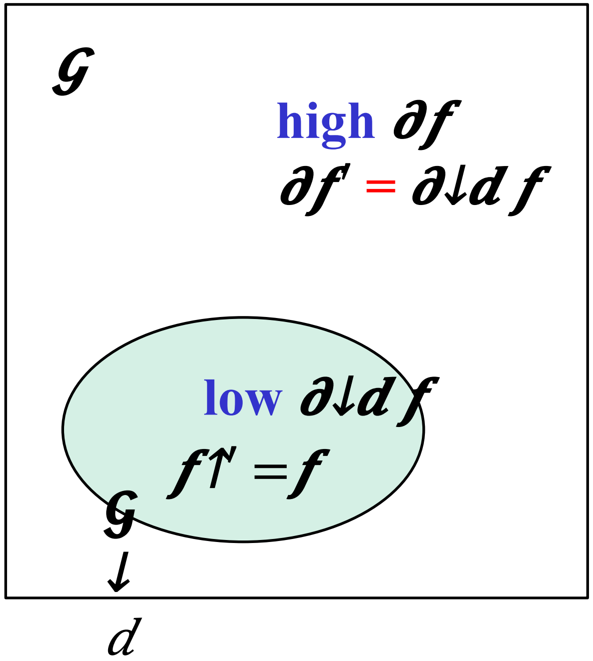
Method 2

Lipschitz extensions

Method 2: Lipschitz extensions [BBDS'13,KNRS'13]

A function f' is a **Lipschitz extension** of f from $\mathcal{G}{\downarrow}d$ to $\mathcal{G}$ if

- f' agrees with f on $\mathcal{G}{\downarrow}d$ *and*
- $\boldsymbol{\partial f' = \partial{\downarrow}d\, f}$



$\mathcal{G}$

high ∂f

$\partial f' = \partial{\downarrow}d\, f$

low $\partial{\downarrow}d\, f$

$f{\uparrow}' = f$

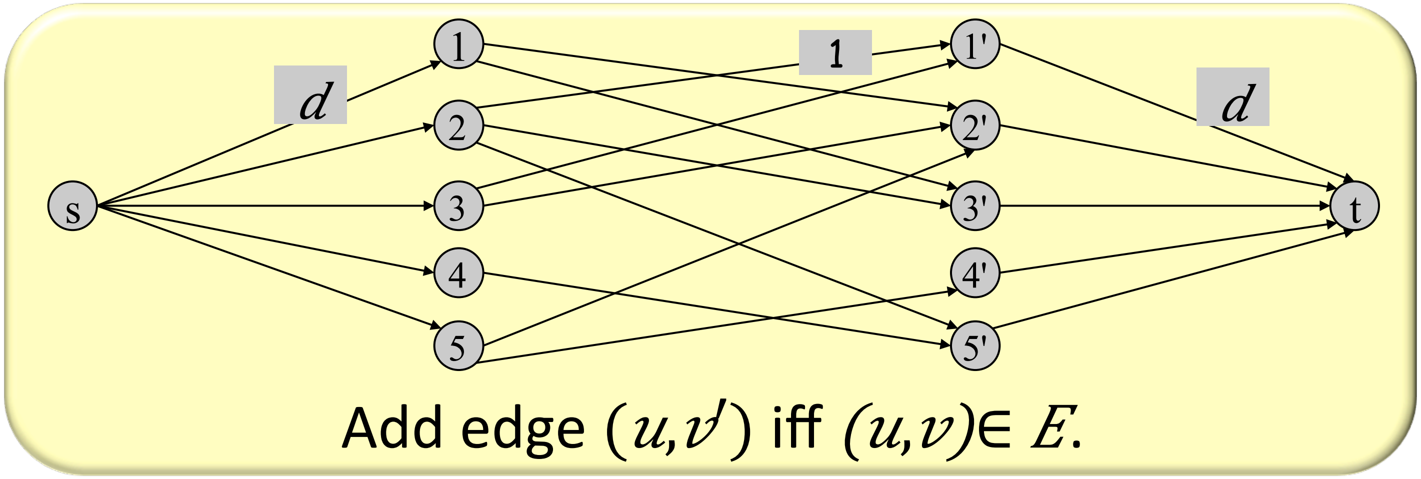
$\mathcal{G}$
$\downarrow$
d

- Release f' via GS framework [DMNS'06]
- There exist Lipschitz extensions for all real-valued fns [BBDS'13]
- Lipschitz extensions can be computed efficiently for
  - subgraph counts [KNRS'13]
  - degree distribution [RS]

Vector of real values

Lipschitz extension of f↓– : flow graph

For a graph G=(V, E), define **flow graph of G**:



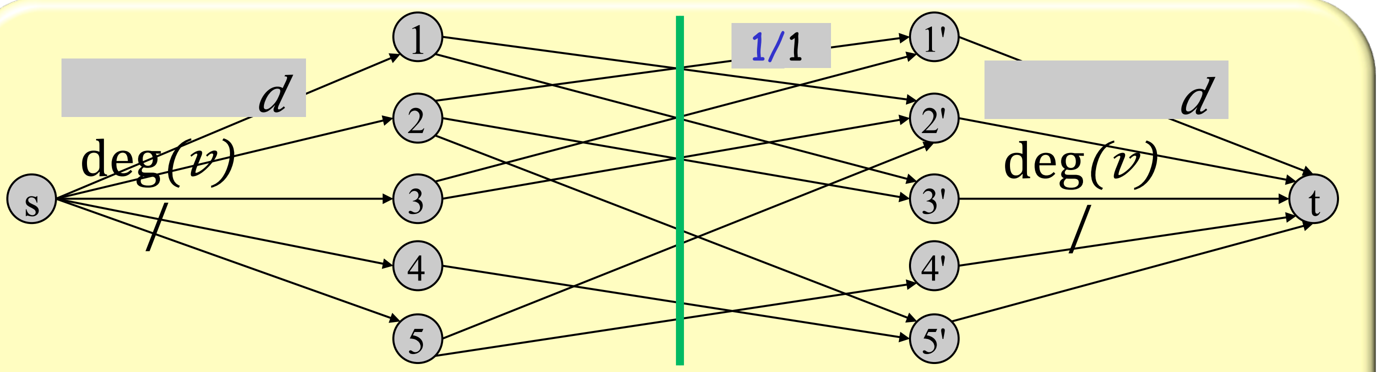
Add edge (*u*,*v'*) iff (*u*,*v*)∈ *E*.

***v*↓flow** (G) is the value of the maximum flow in this graph.

Lemma. ***v*↓flow** **(G)/2** is a Lipschitz extension of ***f*↓–** .

Lipschitz extension of $f{\downarrow}-$: flow graph

For a graph G=(V, E), define **flow graph of G**:



Add edge (u,v') iff $(u,v)\in E$.

$\boldsymbol{v{\downarrow}}$**flow** (G) is the value of the maximum flow in this graph.

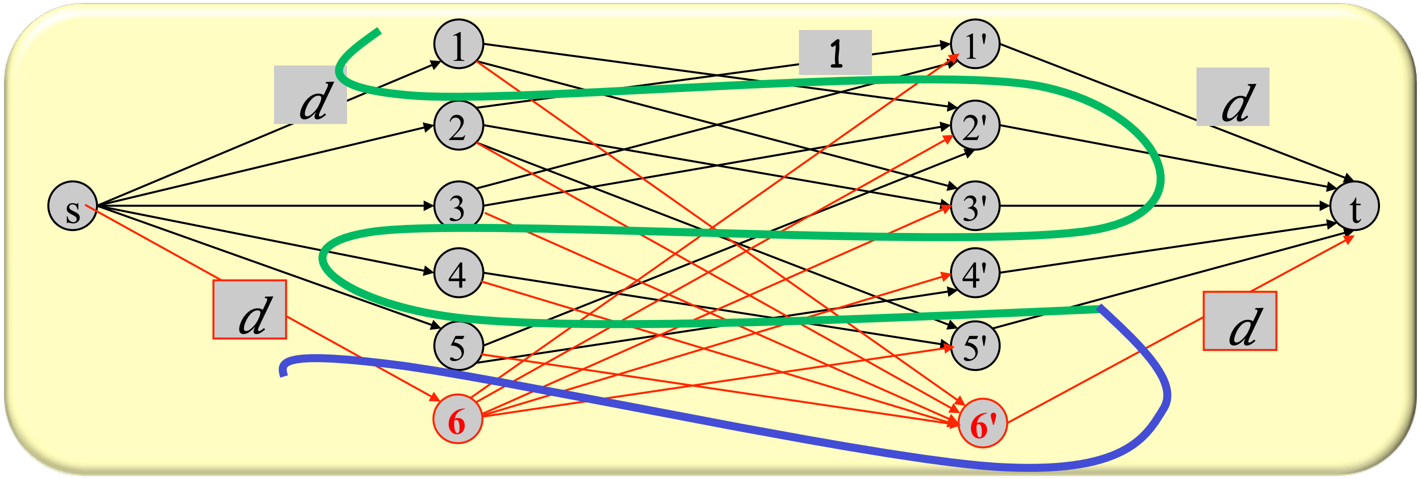
Lemma. $\boldsymbol{v{\downarrow}}$**flow** (G)/2 is a Lipschitz extension of $f{\downarrow}-$.

Proof: (1) $\boldsymbol{v{\downarrow}}$**flow** (G) = $\mathbf{2}\,f{\downarrow}-\ (G)$ for all $\mathbf{G}\in\mathcal{G}{\downarrow}d$

(2) $\partial\, \boldsymbol{v{\downarrow}}$**flow** $= 2\cdot \partial{\downarrow}d\, f{\downarrow}-$

Lipschitz extension of $f\downarrow-$: flow graph

For a graph G=(V, E), define **flow graph of G**:



$v\downarrow$**flow** (G) is the value of the maximum flow in this graph.

Lemma. $v\downarrow$**flow** (G)/2 is a Lipschitz extension of $f\downarrow-$.

Proof: (1) $v\downarrow$**flow** (G) = $2 f\downarrow-$ (G) for all G∈$\mathcal{G}\downarrow d$

(2) $\partial\, v\downarrow$**flow** = 2 · $\partial\downarrow d\, f\downarrow-$ = $2d$

Lipschitz extensions via linear programs

For a graph G=([n], E), define **LP** with variables x_T for all triangles T:

Maximize $\sum_{T=\triangle \text{ of } G} x_T$

$0 \le x_T \le 1$ for all triangles T

$\sum_{T: v\in V(T)} x_T \le \binom{d}{2}$ for all nodes v

$= \partial_d f_\triangle$

$v_{\mathbf{LP}}$ **(G)** is the value of **LP**.

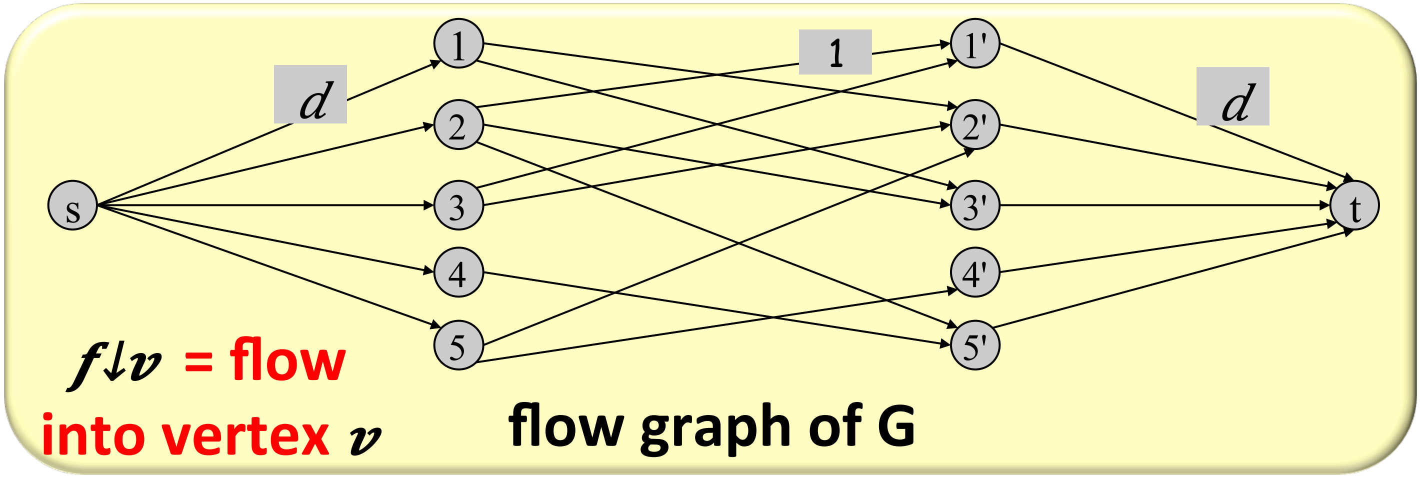
Lemma. $v_{\mathbf{LP}}$ **(G)** is a Lipschitz extension of $f_\triangle$.

- If we use δ instead of $\binom{d}{2}$ as a bound, get a function with GS δ.
  - It is a Lipschitz extension from a large set that includes G_d .

- Can be generalized to other counting queries

Lipschitz extension for a function that outputs a vector

Lipschitz extension of degree distribution via convex programming [RS]



$f\downarrow v$ = flow into vertex v

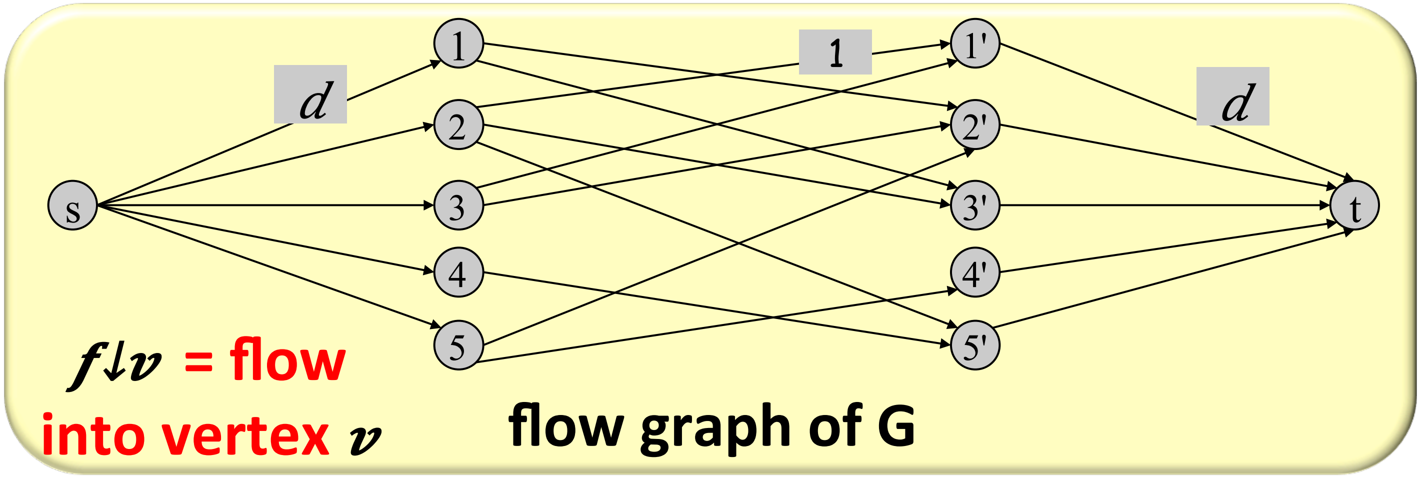
flow graph of G

Can we use $f\downarrow v$ as a proxy for degree of v?

Issue: max flow is not unique.

Want: unique flow that has low global sensitivity.

Lipschitz extension of degree distribution via convex programming [RS]

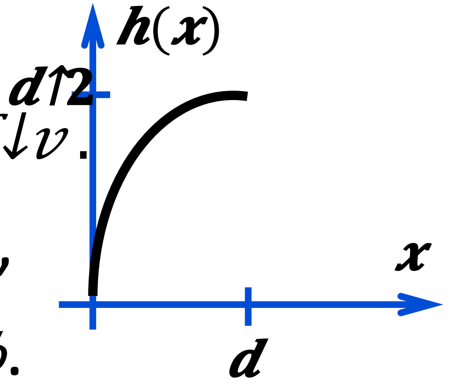


$f{\downarrow}v$ = flow into vertex v

flow graph of G

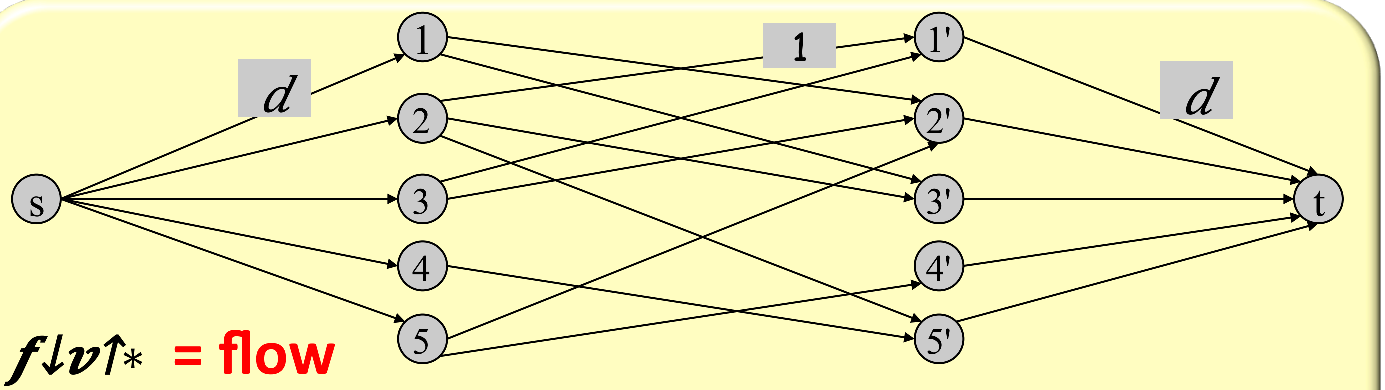
- Let $h(x)=x(2d-x)$.

Idea: maximize $\sum{\downarrow}v\, h(f{\downarrow}v)$ instead of $\sum{\downarrow}v\, f{\downarrow}v$.



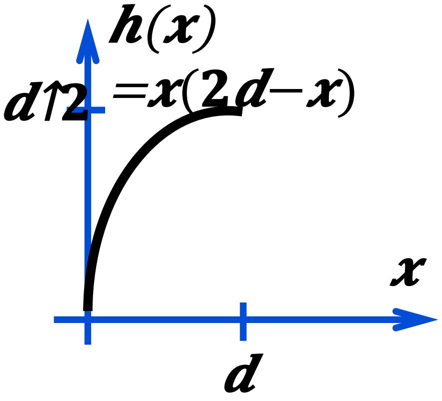
- Let ϕ be the flow maximizing $\sum{\downarrow}v\, h(f{\downarrow}v)$, and $f{\uparrow}*$ be the vector of s-out-flows in ϕ.
- $f{\uparrow}*$ is unique, since h is strictly concave.
- It can be computed in poly time [Lee Rao Srivastava 13].

Lipschitz extension of degree distribution via convex programming [RS]

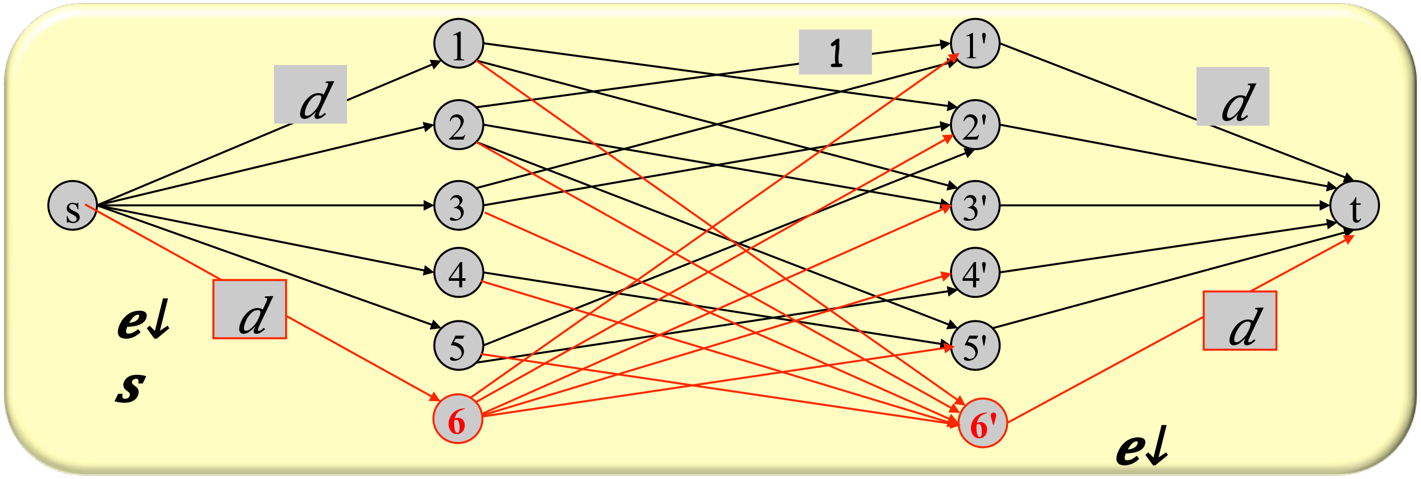


f_v^* = flow into vertex v $\quad \phi=\text{argmax}\sum_{v} h(f_v)$.

- If $\mathrm{G}\in\mathcal{G}_d$, then f_v^* = deg(v) for all v, since h is strictly increasing on $[0,d]$.
- **Lemma.** ℓ_1 **global sensitivity** $\partial f^* \le 3d$.



Lipschitz extension of degree distribution via convex programming [RS]



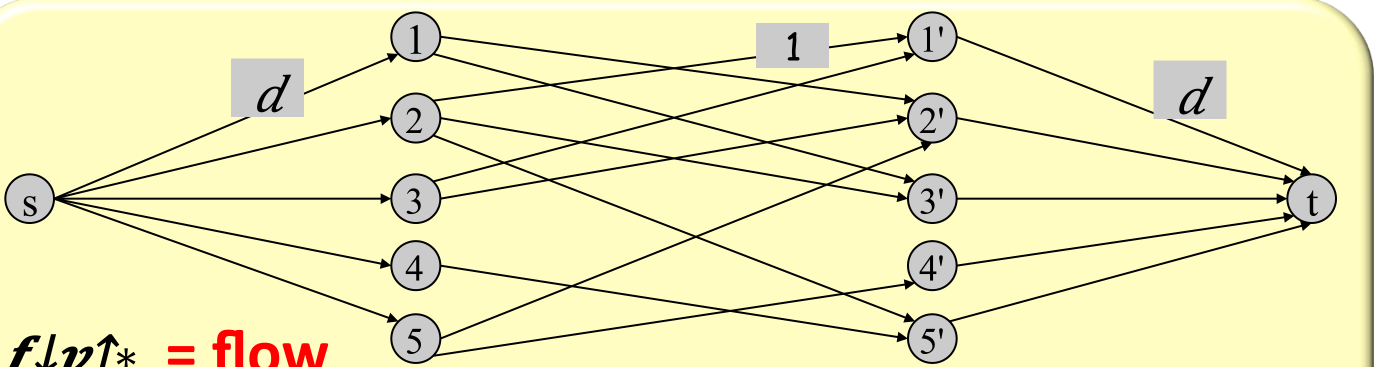
Lemma. ℓ_1 global sensitivity $\partial f^* \le 3d$.

Proof sketch: Consider $g = \phi_{new} - \phi_{old}$.

g is a union of simple s-t-paths and cycles of several types:

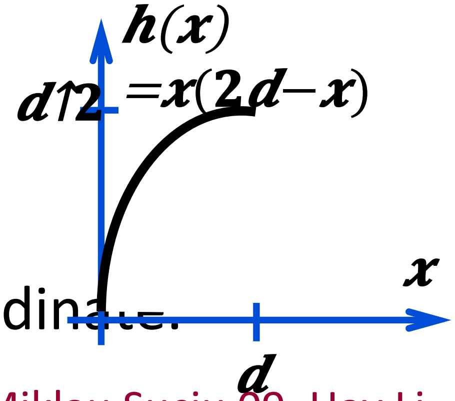
1. s-t-paths and cycles using e_s. Contribute $\le 2d$ to $|f_{new}^* - f_{old}^*|_1$
2. s-t-paths using e_t. $\le d$
3. Cycles using e_t. 0
4. Remaining paths and cycles. Do not exist.

Releasing degree distribution: summary



f_v^* = flow into vertex v

$\phi=\text{argmax}\sum_{v} h(f_v)$.



1. Construct flow graph of G.
2. Compute s-out-flows f^* .
3. Release vector f^* , with Lap$(3d/\epsilon)$ per coordinate.
4. Use post-processing techniques by [Hay Rastogi Miklau Suciu 09, Hay Li Miklau Jensen 09, Karwa Slavkovic 12, Kifer Lin 13] to remove some noise.

Method 3

Recursive mechanism

Method 3: recursive mechanism [Chen Zhou 13]

Strategy for releasing real-valued functions $f(G)$

- Define functions $X\downarrow\delta\,(\mathrm{G})$ with global sensitivity δ.

  As in projection methods,

  - $X\downarrow\delta\,(\mathrm{G})\leq f(G)$ and
  - $X\downarrow\delta\,(\mathrm{G})$ is closer to $f(G)$ for larger δ.

- Release $X\downarrow\delta\,(\mathrm{G})$ for a carefully chosen δ via Laplace mechanism.

Defining $X_{\delta}(G)$

- Given graph G, define sequence in R^{+} :

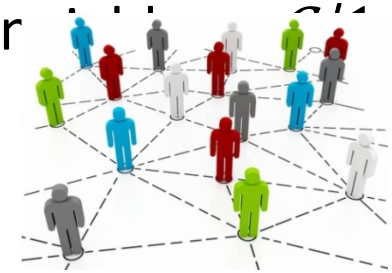
$0=H_{0}(G)\leq H_{1}(G)\leq\ldots\leq H_{n}(G)=\boldsymbol{f}(\boldsymbol{G})$.

E.g., $H_{i}(g)=\min_{subgraphs\ G'\ of\ G\ of\ size\ i}\boldsymbol{f}(\boldsymbol{G'})$.

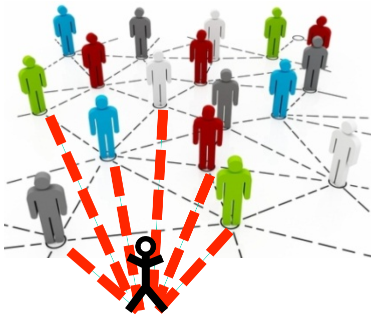
- H_{i}'s must be interleaving: $H_{i}(G_{2})\leq H_{i}(G_{1})\leq H_{i+1}(G_{2})$

for all r $\subset G_{2}$ and $i=0,1,$

$\boldsymbol{G_{1}}$:



$\boldsymbol{G_{2}}$:



- Define $X_{\delta}(\mathrm{G})=\min_{0\leq i\leq n}(H_{i}(G)+(n-i)\delta)$.

Lemma. $X_{\delta}(\mathrm{G})=f(G)$ for $\delta\geq\max_{i}(H_{i+1}(G)-H_{i}(G))$.

Global sensitivity of $X_{\delta}(G)$

- H_i's must be interleaving: $H_i(G_2) \leq H_i(G_1) \leq H_{i+1}(G_2)$

  for all neighbors $G_1 \subset G_2$ and $i=0,1,\ldots,n$.

- Define $X_{\delta}(\mathrm{G}) = \min_{0 \leq i \leq n} (H_i(G) + (n-i)\delta)$.

Lemma. Global node sensitivity of X_{δ} is δ.

Proof: Consider neighbors $G_1 \subset G_2$.

1. Want to show: $X_{\delta}(\mathrm{G}_2) \leq X_{\delta}(\mathrm{G}_1) + \delta$.

   Let i^* be the index that minimizes the expression for $X_{\delta}(\mathrm{G}_1)$.

   $$X_{\delta}(\mathrm{G}_2) \leq H_{i^*}(G_2) + (n+1-i^*)\delta$$

   $$\leq H_{i^*}(G_1) + (n-i^*)\delta + \delta = X_{\delta}(\mathrm{G}_2) + \delta.$$

2. Similarly, can show $X_{\delta}(\mathrm{G}_1) \leq X_{\delta}(\mathrm{G}_2)$.

Computationally efficient recursive mechanism

Recall: $X_{\delta}(G)=\min_{0\le i\le n}\left(H_i(G)+(n-i)\delta\right)$.

E.g., $H_i(G)=\min_{\blacksquare \text{subgraphs } G' \text{ of}}$ NP-hard $\boldsymbol{f}(G')$.

Idea: Use an LP-relaxation of H_i .

E.g., for $\boldsymbol{f}_{\Delta}$ (the number of triangles):

$$H_i(G)=\min \sum_{(u,v,w)=\Delta \text{ of } G} \max(0, (x_u+x_v+x_w-2))$$

$0\le x_v\le 1$ for all nodes v

$\sum_{v} x_v = i$

Output: $X_{\delta}(G)$ in global sensitivity framework.

Summary

- New techniques
  1. **Truncation + smooth sensitivity** [BBDS’13, KNRS’13]
  2. **Lipschitz extensions** [BBDS’13, KNRS’13]
  3. **Recursive mechanism** [Chen Zhou 13]

- Unifying idea: ``projections’’ on ``graphs’’ with low sensitivity
  - Generic reduction to privacy over bounded-degree graphs

    truncation + smooth sensitivity [BBDS’13,KNRS’13]
  - Releasing number of edges and subgraph counts

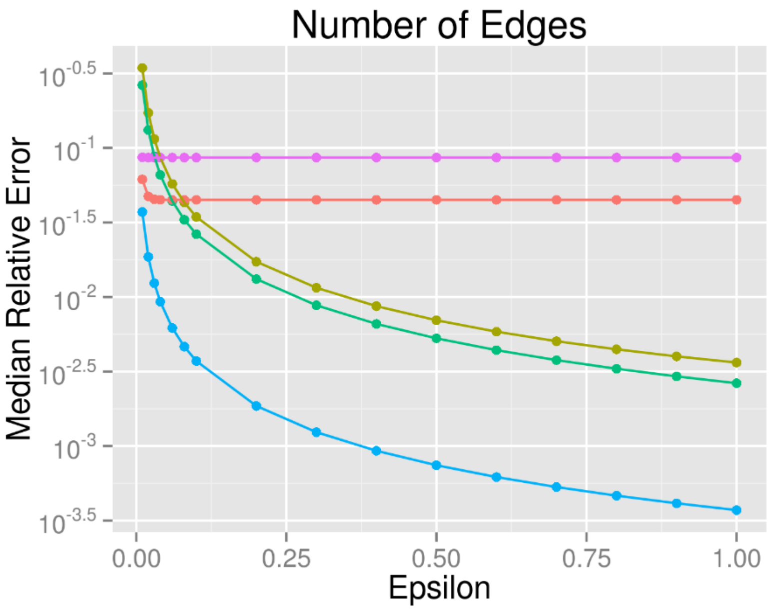
    Lipschitz extensions via max flow and LP [KNRS’13]
  - Releasing degree distribution

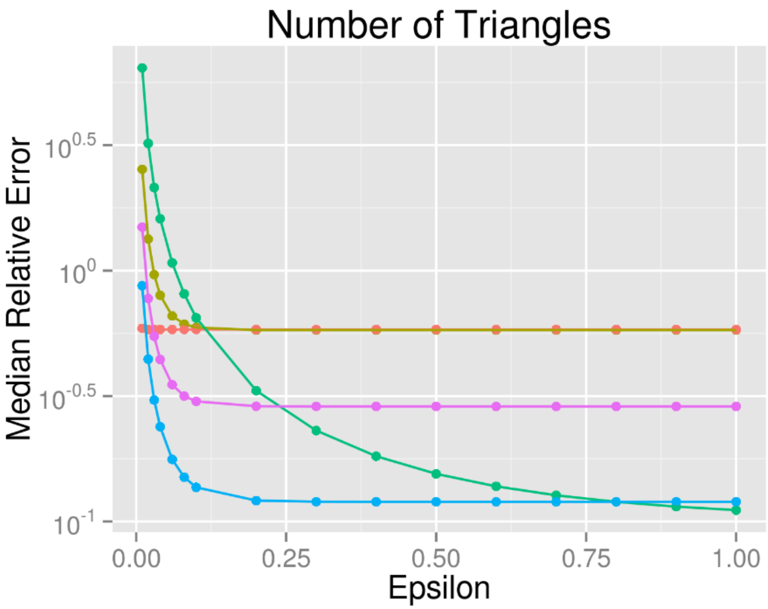
    Lipschitz extension via convex programming [Raskhodnikova Smith]
  - Releasing subgraph counts

    Recursive mechanism [Chen Zhou 13]

Experimental evaluation

Experiments for the flow and LP method [Lu]


Number of Edges


Number of Triangles

	Graph	# nodes	# edges	Max degree	Time, secs # edges	Time, secs # Δs
🟡	CA-GrQc	5,242	28,992	81	0.02	7
🟢	CA-HepTh	9,877	51,996	65	0.68	0.5
🔴	CA-AstroPh	18,772	396,220	504	0.34	10,222
🔵	com-dblp-ungraph	317,080	2,099,732	343	2	2128
🟣	com-youtube-ungraph	1,134,890	5,975,248	28,754	9	94

Other experimental results

[Lu] showed that truncation is less accurate than flow and LP-based methods.

[Chen Zhou 13] provide experimental evaluation on random and real-world graphs.

- (Mostly) better accuracy than in [KRSY'11] for edge-DP algs.
- Comparable (slightly better?) accuracy on smaller graphs than in experiments of [Lu] for node-DP algorithms.
- Longer running times.
- Not enough experiments to compare the two node-DP methods.

Conclusions

- We are close to having edge-private and node-private algorithms that work well in practice for basic graph statistics.
- Interesting projection techniques that might be useful for design of DP algorithms in other contexts.

Open questions

- New techniques:
  - Can special-purpose LP-solvers make them more efficient?
  - To which other queries do they apply?
  - What's the best way to choose the degree/sensitivity cut off?
- Specific queries:
  - Releasing cuts with node-DP
  - Releasing pairwise distances between nodes with DP

Open questions (continued)

- DP synthetic graphs
- Simultaneous release of answers to many queries
- What are the right notions of privacy for graph data?
- What are the right ways to state utility guarantees?
  - Some proposals in [KRSY'13, KNRS'13, Chen Zhou 13]
- Social networks have node and edge attributes. What queries are useful?
- Hypergraphs (that capture relationships such as "people appearing on the same photo")

"Projections" on graphs of small degree

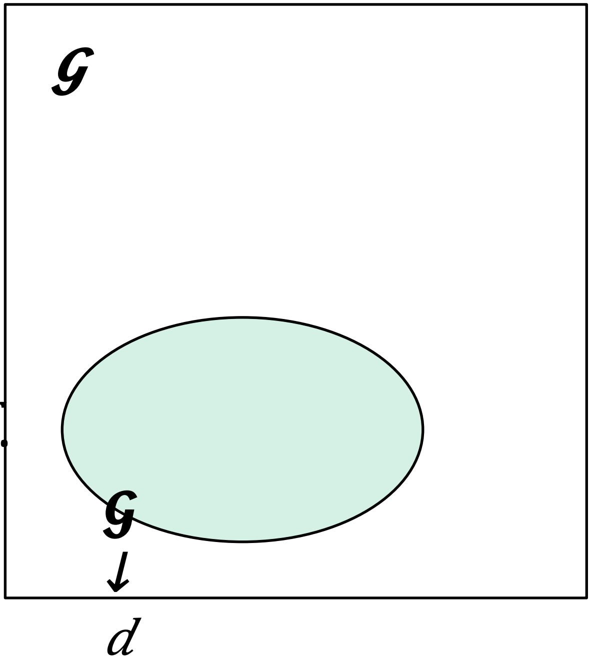
Let $\mathcal{G}$ = family of all graphs,

$\mathcal{G}_d$ = family of graphs of degree $\leq d$.

Notation. ∂f = **node** GS_f over $\mathcal{G}$.

$\partial_d f$ = **node** GS_f over $\mathcal{G}_d$.

Observation. $\partial_d f$ is low for many useful f.



Examples:

- $\partial_d f_{-} = d$ **(compare to** $\partial f_{-} = n$**)**
- $\partial_d f_{\triangle} = \binom{d}{2}$ **(compare to** $\partial f_{\triangle} = \binom{n}{2}$**)**

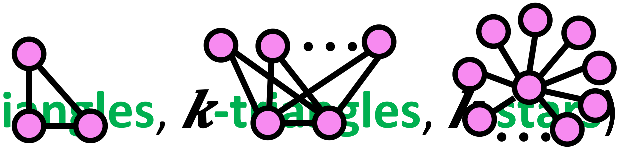
Goal: privacy for all graphs

Idea: ``Project'' on graphs in $\mathcal{G}_d$ for a carefully chosen d << n.

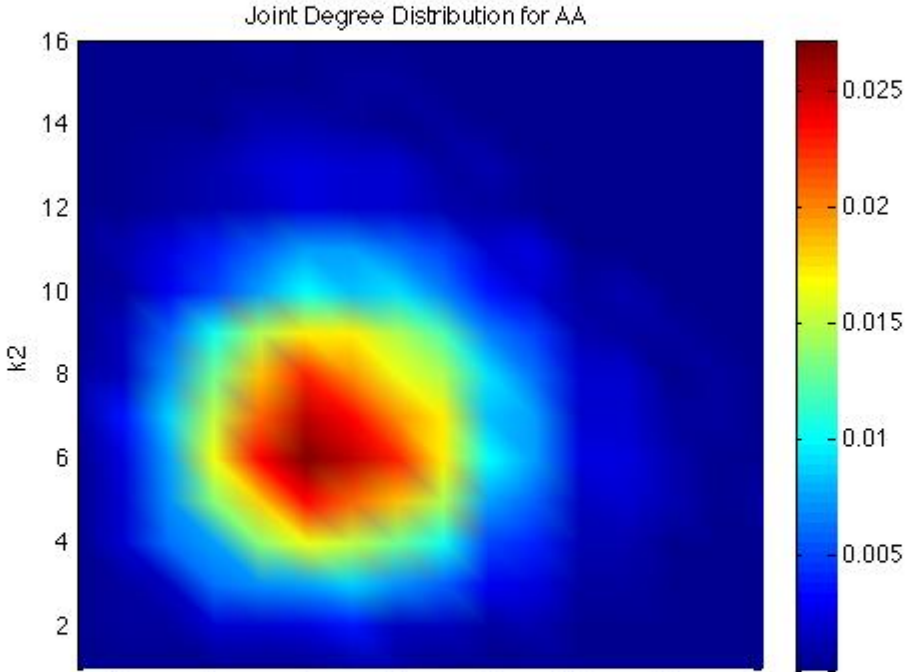
Graph statistics

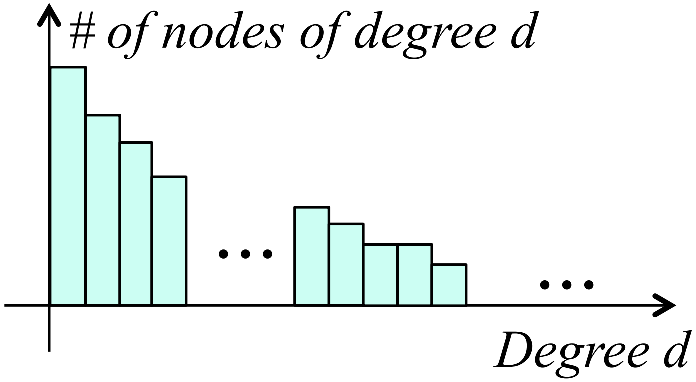
- Number of edges
- Counts of small subgraphs

(e.g., triangles, *k*-triangles, *k*-stars)



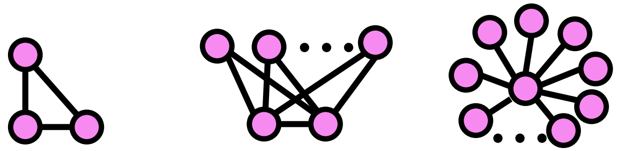
- Degree distribution
- Joint degree distribution
- Cuts





Our contributions: algorithms

- Node differentially private algorithms for releasing
  - number of edges
  - counts of small subgraphs



(e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

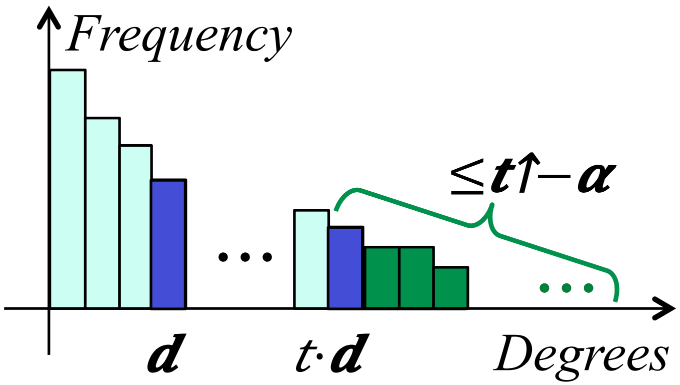
  - degree distribution

- Accuracy analysis of our algorithms for graphs with not-too-heavy-tailed degree distribution: with ***α*-decay** for constant $\alpha>1$

Notation: $\boldsymbol{d} =$ *average degree*

$\boldsymbol{P(d)}=$ fraction of nodes in G of degree $\geq d$

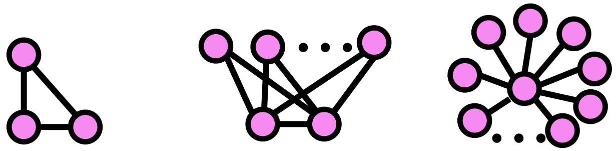
A graph G satisfies ***α*-decay** if for all $t>1$: $P(t\cdot d) \leq t^{-\alpha}$



  - Every graph satisfies 1-decay
  - Natural graphs (e.g., **"scale-free" graphs**, **Erdos-Renyi**) satisfy $\alpha>1$

Our contributions: accuracy analysis

- Node differentially private algorithms for releasing
  - number of edges
  - counts of small subgraphs



(e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

  - degree distribution
- Accuracy analysis of our algorithms for graphs with not-too-heavy-tailed degree distribution: with ***α*-decay** for constant $\alpha > 1$

A graph G satisfies ***α*-decay** if for all $t > 1$: $P(t \cdot \bar{d}) \le t^{-\alpha}$

  - number of edges
  - counts of small subgraphs (e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

$\}$ **(1+o(1))-approximation**

  - degree distribution $\}$ $\|\mathbf{A}_{\epsilon,\alpha}(\mathbf{G}) - \mathbf{DegDistrib}(\mathbf{G})\|_1 = \mathbf{o}(\mathbf{1})$