Survey of Techniques for Node Differential Privacy

Sofya Raskhodnikova

Penn State University,

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

also visiting Harvard

PENNSTATE
1855

Publishing information about graphs

Many types of data can be represented as graphs where

- nodes correspond to individuals
- edges capture relationships
  - **"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

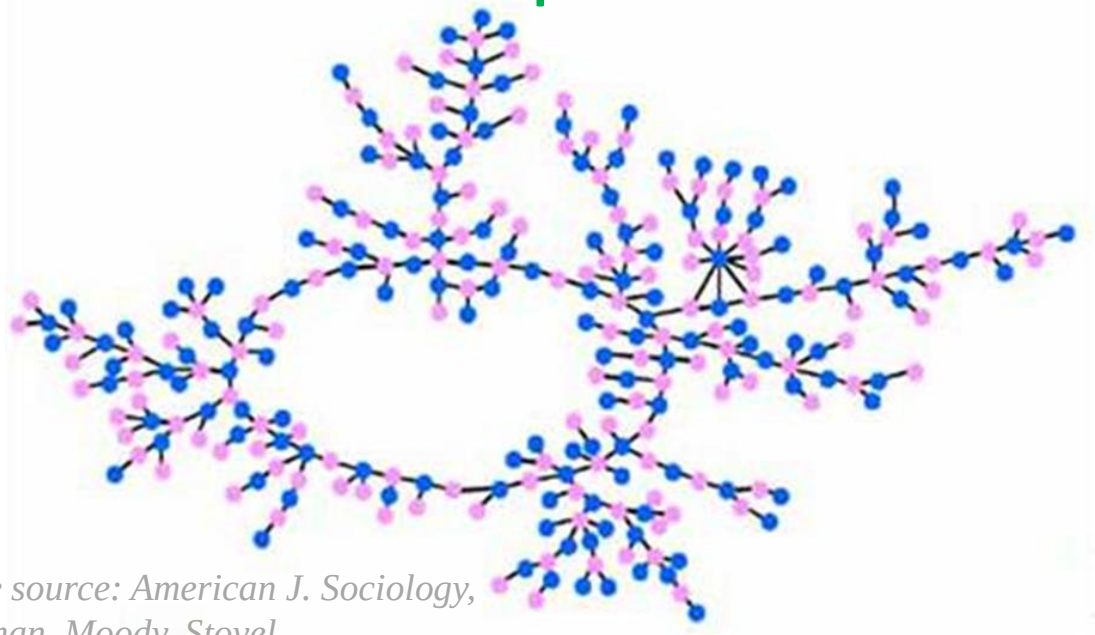
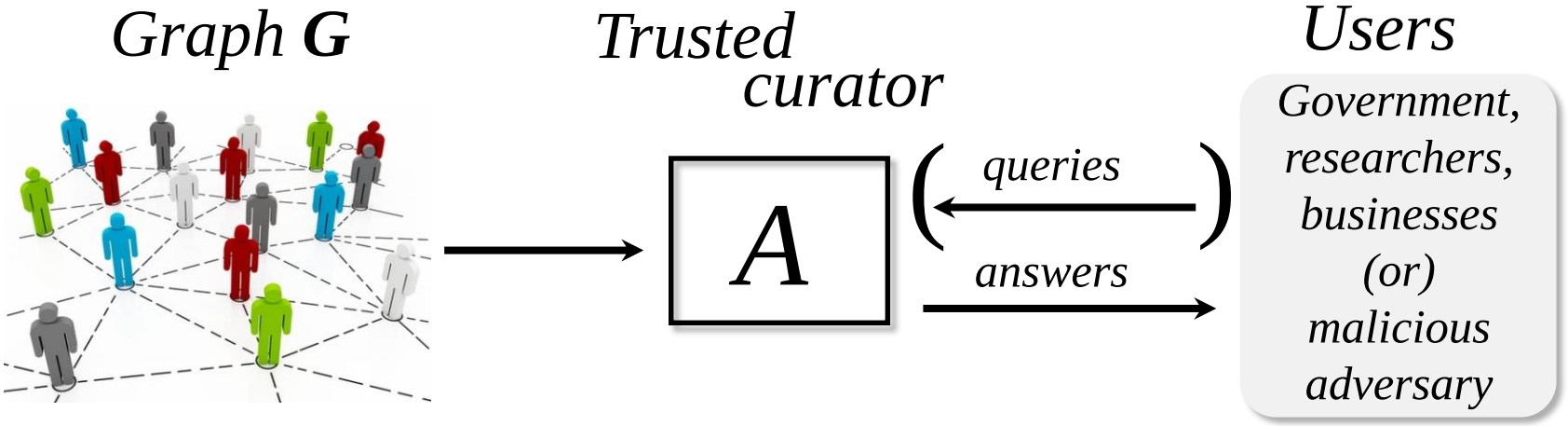


Image source: American J. Sociology, Bearman, Moody, Stovel

Privacy is a big issue!

Differential privacy (for graph data)



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/

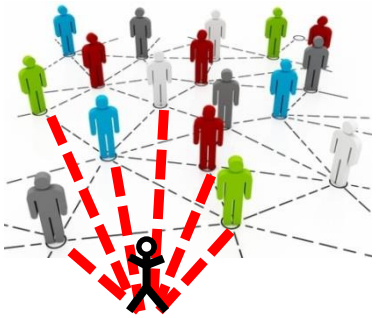
Two variants of differential privacy for graphs

- **Edge** differential privacy

G:  G': 

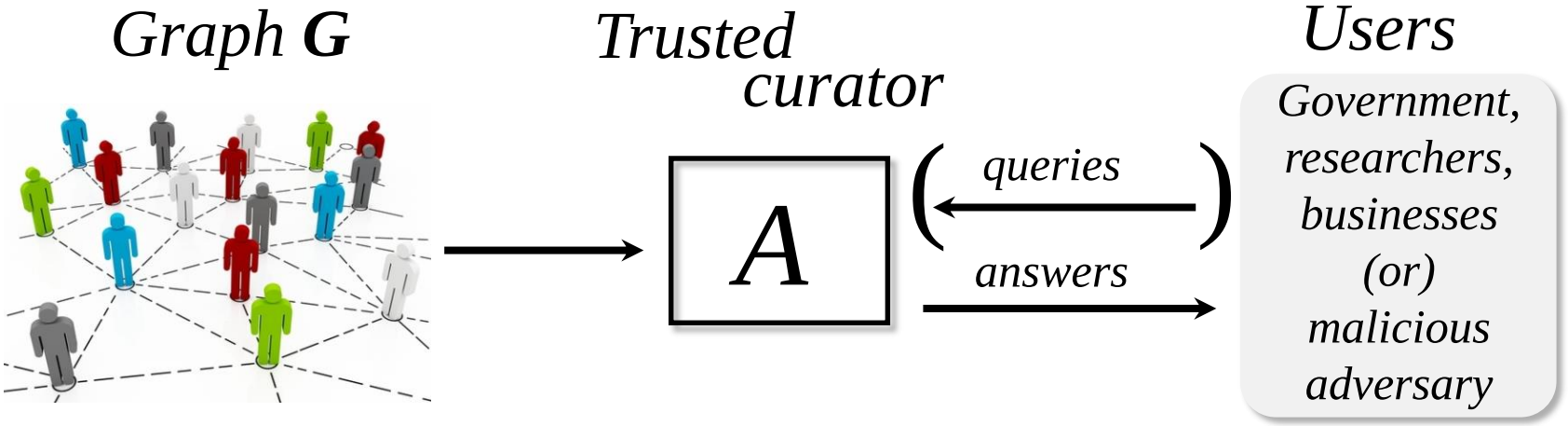
Two graphs are **neighbors** if they differ in ***one edge***.

- **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

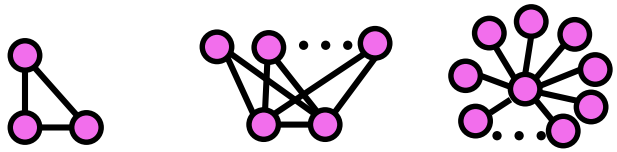


- **Two conflicting goals:** utility and privacy
  - Impossible to get both in the worst case
- **Want:** differentially private algorithms that are accurate on realistic graphs
  - ***differentially private*** *(**for all** graphs)*
  - ***accurate for a subclass*** *of graphs*

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

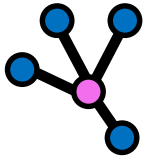
Graph statistics

- Number of edges
- Counts of small subgraphs

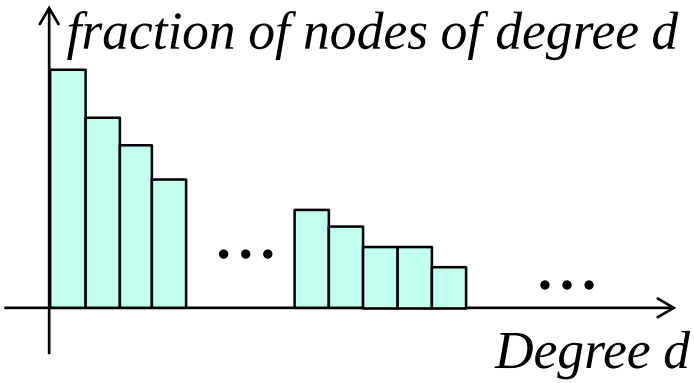


(e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

(If nodes have colors, node colors can be specified in template graphs.)



- 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

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

New Techniques

1. **Truncation + smooth sensitivity** [BBDS'13, KNRS'13]

– Generic reduction to privacy over bounded-degree graphs

– Fast

2. **Lipschitz extensions** [BBDS'13, KNRS'13]

– Releasing number of edges **via max flow** [KNRS'13]

– Releasing subgraph counts **via linear programming** [KNRS'13]

– Releasing degree distribution: **via convex programming** [RS]

– Slower, but more accurate

3. **Recursive mechanism** [CZ'13]

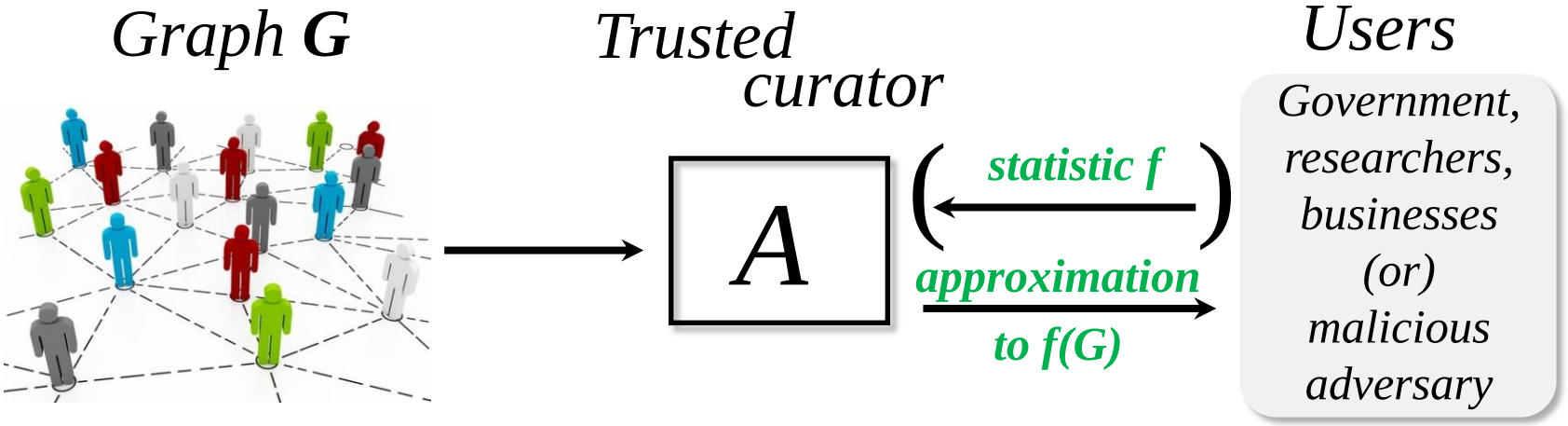
(can be viewed as an efficient construction of Lipschitz extensions)

– Releasing (generalization of) subgraph counts

– The same accuracy as (2), but slower

- Unifying idea: ``projections'' on ``graphs'' with low sensitivity

Basic question



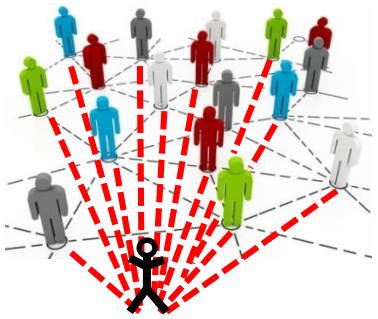
How accurately

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

Challenge for node privacy: high sensitivity

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

$$\partial f = \max_{(\text{node})\text{neighbors } G,G'} |f(G) - f(G')|$$



- **Examples:**

➢ $f_{\_}$(G) is the number of edges in G.

➢ $f_{\triangle}$(G) is the number of triangles in G.

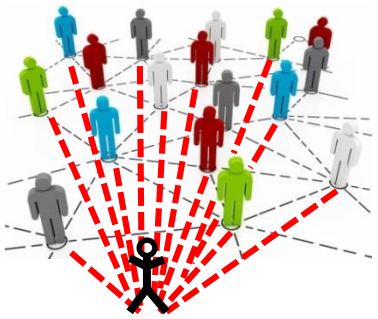
$\partial f_{\_} = n.$

$\partial f_{\triangle} = \binom{n}{2}.$

Challenge for node privacy: high sensitivity

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

$$\partial f = \max_{(\text{node})\text{neighbors } G,G'} |f(G) - f(G')|$$



- **Local sensitivity $LS_f(G)$**, $\max_{G' : \text{neighbor of } G} |f(G) - f(G')|$, is also high.
- New measure of sensitivity [Chen Zhou 13]

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

Idea: project onto graphs with low down sensitivity.

"Projections" on graphs of small degree [BBDS'13,KNRS'13]

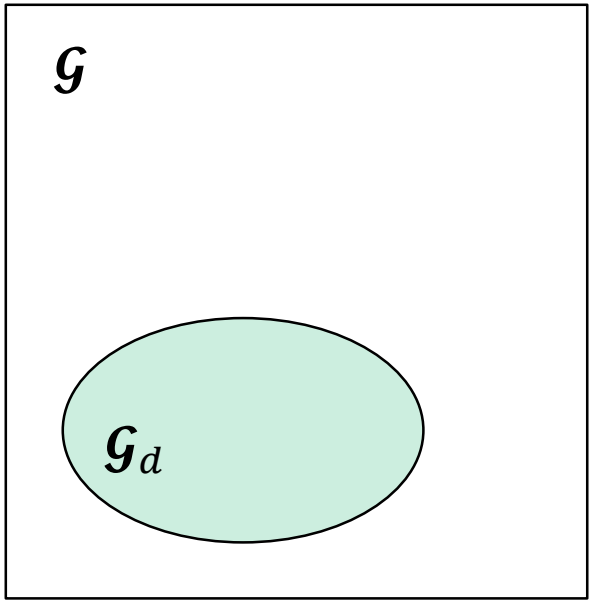
Let $\mathcal{G}$ = family of all graphs,

$\mathcal{G}_d$ = 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.

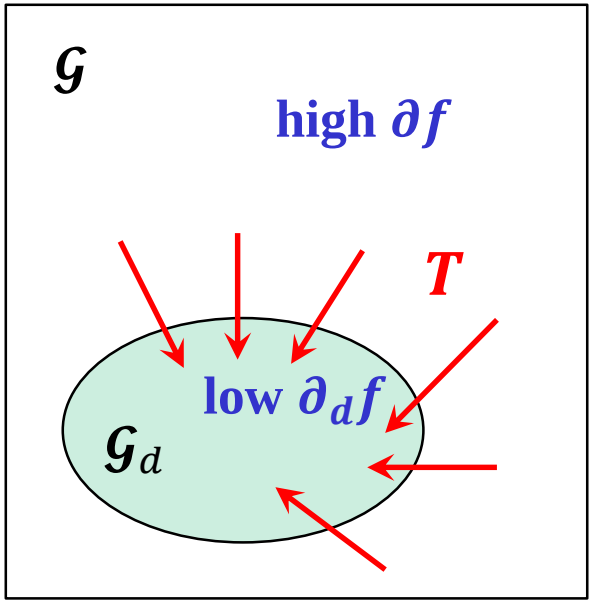
Method 1

Truncation + local-sensitivity-based frameworks

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

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

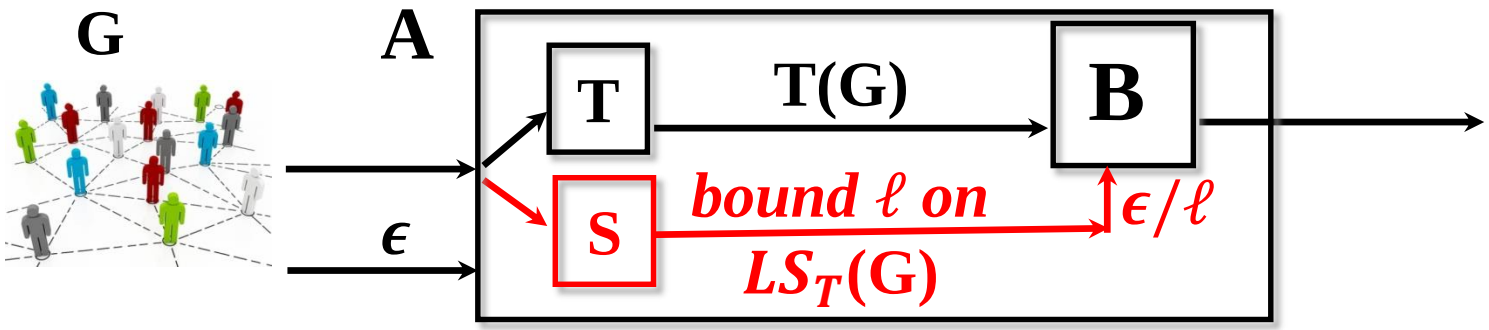
Output: Algorithm A that is node-DP over $\mathcal{G}$, has accuracy similar to B on "nice" graphs



- 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 local-sensitivity-based frameworks [NRS'07,Dwork Lei 09, KRSY'11]



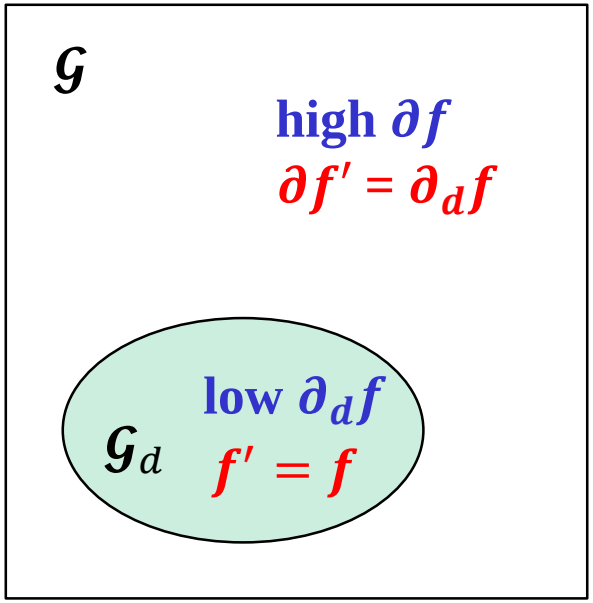
Method 2

Lipschitz extensions

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

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

- f' agrees with f on $\mathcal{G}_d$ *and*
- $\partial f' = \partial_d f$



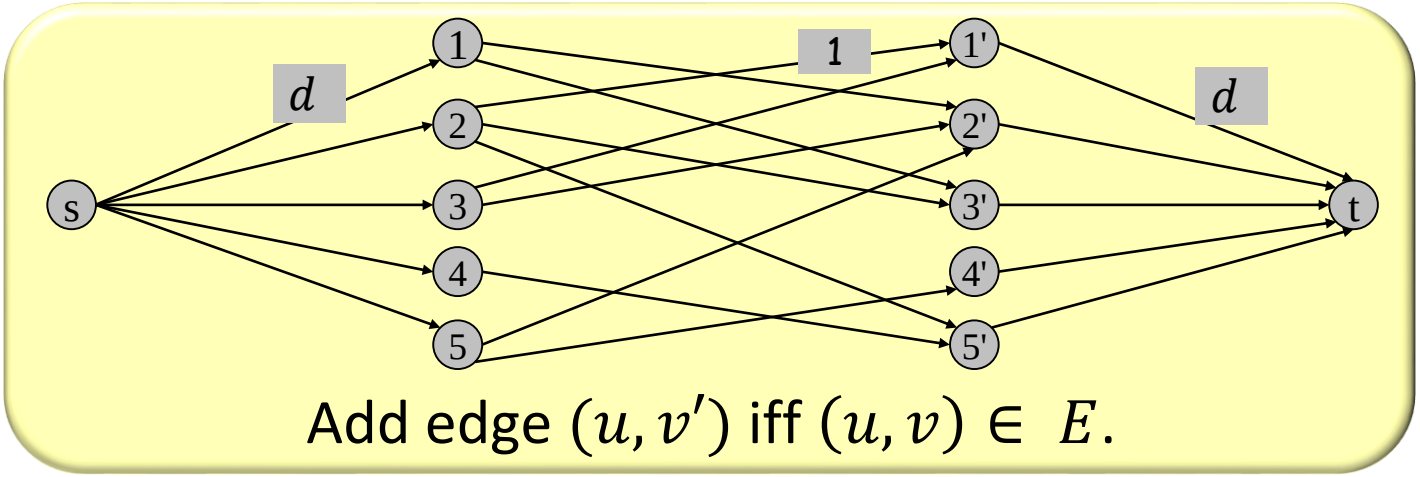
- Release f' via GS framework [DMNS'06]

- There exist Lipschitz extensions for all real-valued functions
- 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**:

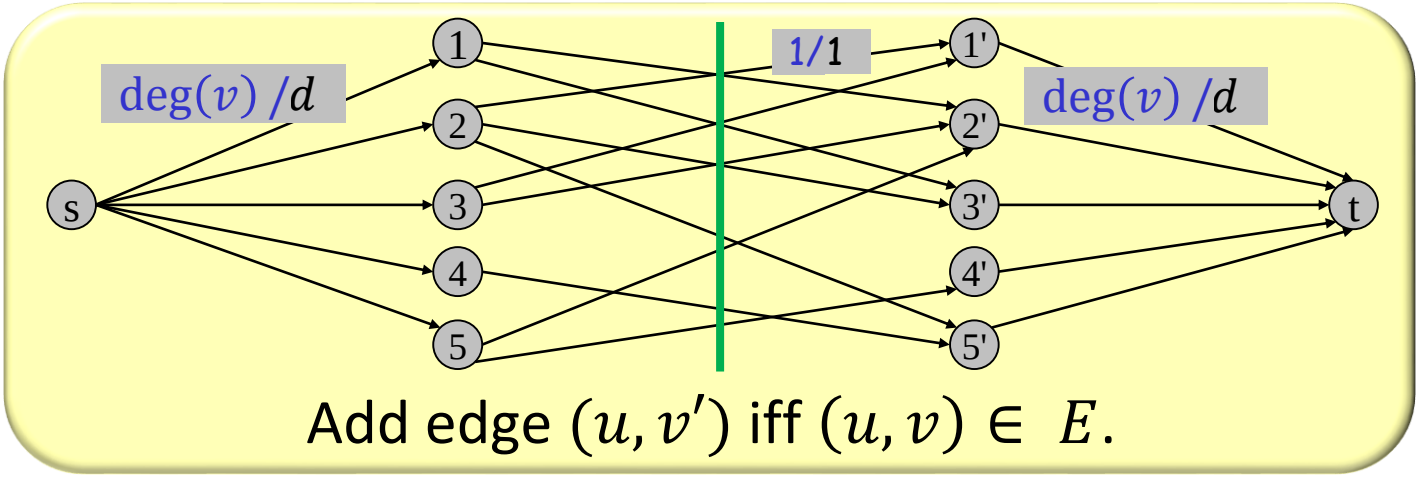


v_{flow}**(G)** is the value of the maximum flow in this graph.

Lemma. v_{flow}**(G)/2** is a Lipschitz extension of $f\_$.

Lipschitz extension of $f\_$: flow graph

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



v_{flow}(G) is the value of the maximum flow in this graph.

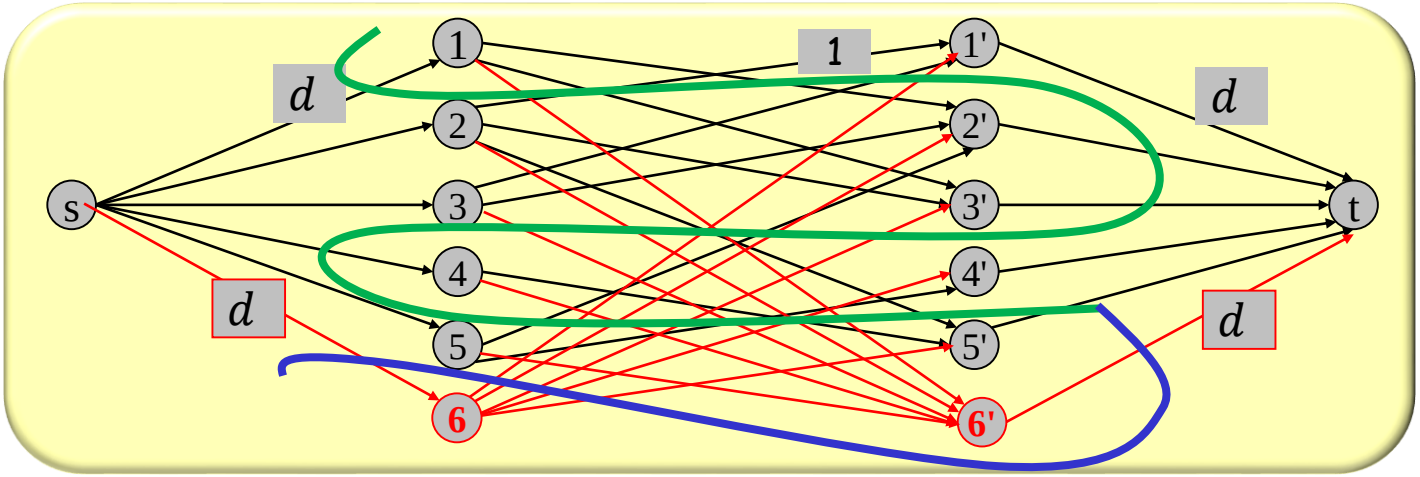
Lemma. v_{flow}(G)/2 is a Lipschitz extension of $f\_$.

Proof: (1) $v_{\text{flow}}(G) = 2f\_(G)$ for all $G \in \mathcal{G}_d$

(2) $\partial\, v_{\text{flow}} = 2 \cdot \partial_d f\_$

Lipschitz extension of $f\_$: flow graph

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



v_{flow}(G) is the value of the maximum flow in this graph.

Lemma. v_{flow}(G)/2 is a Lipschitz extension of $f\_$.

Proof: (1) $v_{\text{flow}}(G) = 2f\_(G)$ for all $G \in \mathcal{G}_d$

(2) $\partial\, v_{\text{flow}} = 2 \cdot \partial_d f\_ = 2d$

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

$= \partial_d f_\triangle$

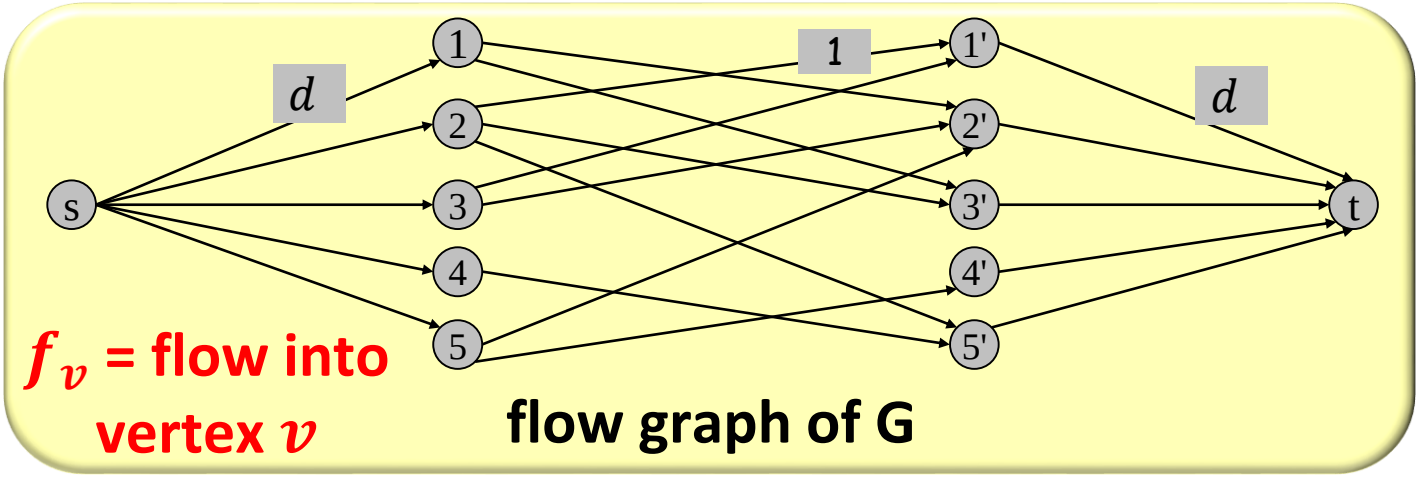
$\nu_{LP}(G)$ is the value of **LP**.

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

- Computable in time $\tilde{O}(n + f_\triangle(G))$ using [Plotkin Shmoys Tardos]
- 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]

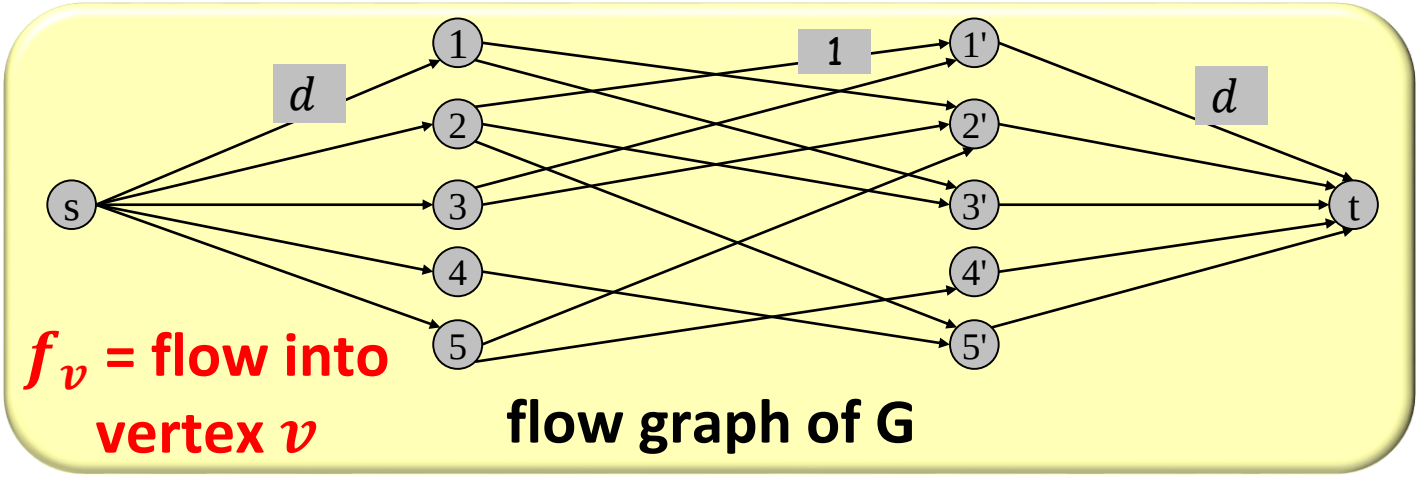


Can we use f_v as a proxy for degree of v?

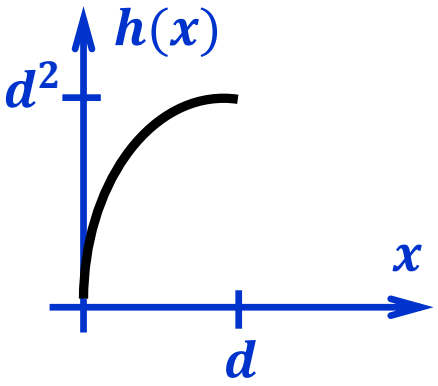
Issue: max flow is not unique.

Want: unique flow that has low global sensitivity.

Lipschitz extension of degree distribution via convex programming [RS]



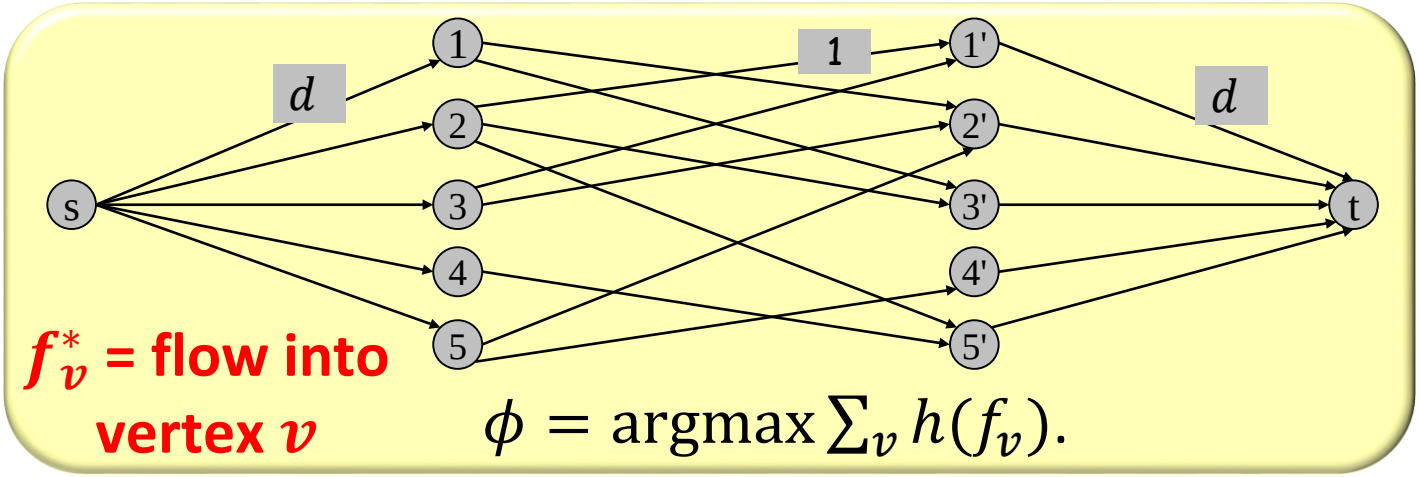
- Let $h(x) = x(2d - x)$.



Idea: maximize $\sum_v h(f_v)$ instead of $\sum_v f_v$.

- Let ϕ be the flow maximizing $\sum_v h(f_v)$, and f^* be the vector of s-out-flows in ϕ.
- f^* is unique, since h is strictly concave.
- Poly-time computable (e.g., [Lee Rao Srivastava 13]).

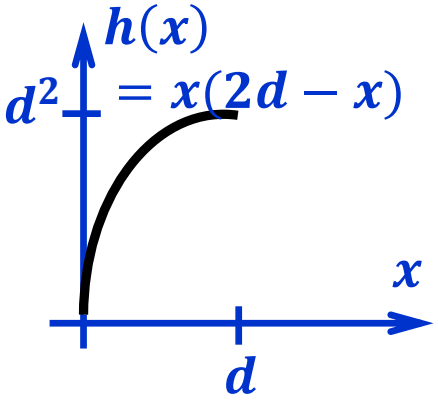
Lipschitz extension of degree distribution via convex programming [RS]



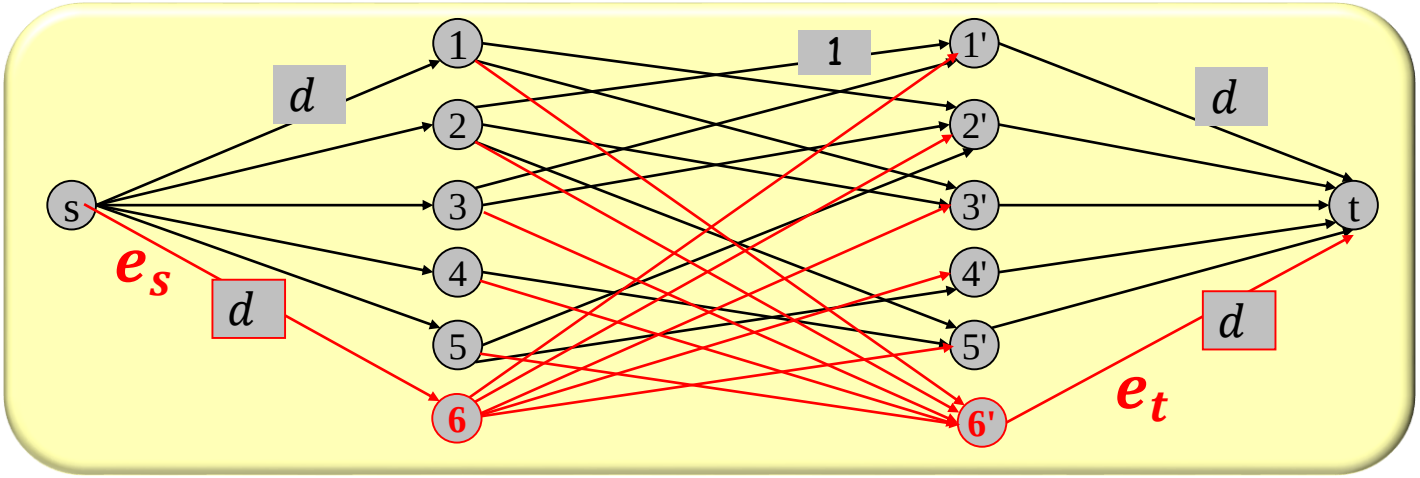
- If **G**$\in \mathcal{G}_d$, then f_v^***= deg(v) for all v**,

  since h is strictly increasing on [0,d].

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



Lipschitz extension of degree distribution via convex programming [RS]



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

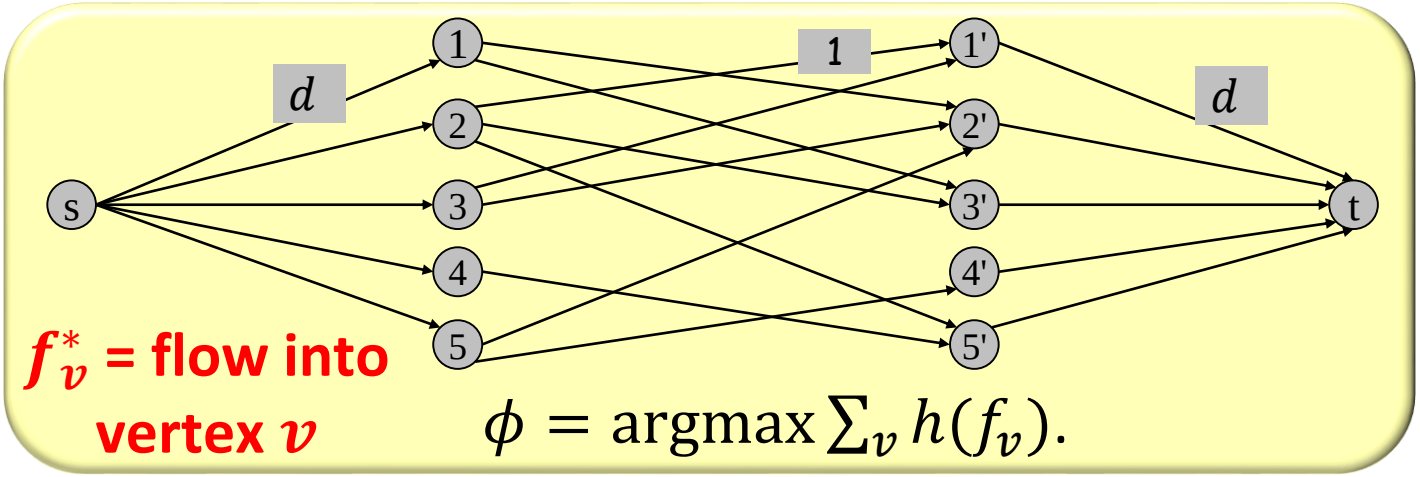
Proof sketch: Consider $g = \phi_{new} - \phi_{old}$.

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

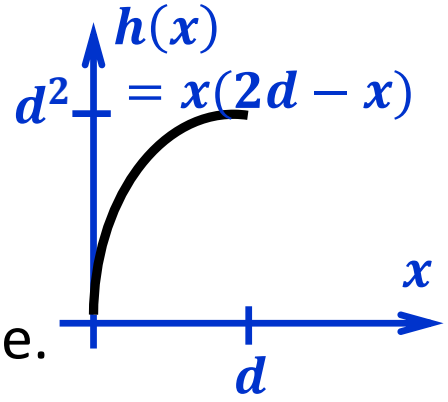
1. s-t-paths and cycles using e_s. Contribute $\leq 2d$ to $|f^*_{new} - f^*_{old}|_1$
2. s-t-paths using e_t. $\leq d$
3. Cycles using e_t. 0
4. Remaining paths and cycles. Do not exist.

Use strict concavity of h

Releasing degree distribution: summary



1. Construct flow graph of G.
2. Compute s-out-flows f^*.
3. Release vector f^*, with $\text{Lap}\left(\frac{3d}{\epsilon}\right)$ 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.



Summary

1. **Truncation + smooth sensitivity** [BBDS'13, KNRS'13]

– Generic reduction to privacy over bounded-degree graphs

– Fast

2. **Lipschitz extensions** [BBDS'13, KNRS'13]

– Releasing number of edges **via max flow** [KNRS'13]

– Releasing subgraph counts **via linear programming** [KNRS'13]

– Releasing degree distribution: **via convex programming** [RS]

– Slower, but more accurate

3. **Recursive mechanism** [CZ'13]

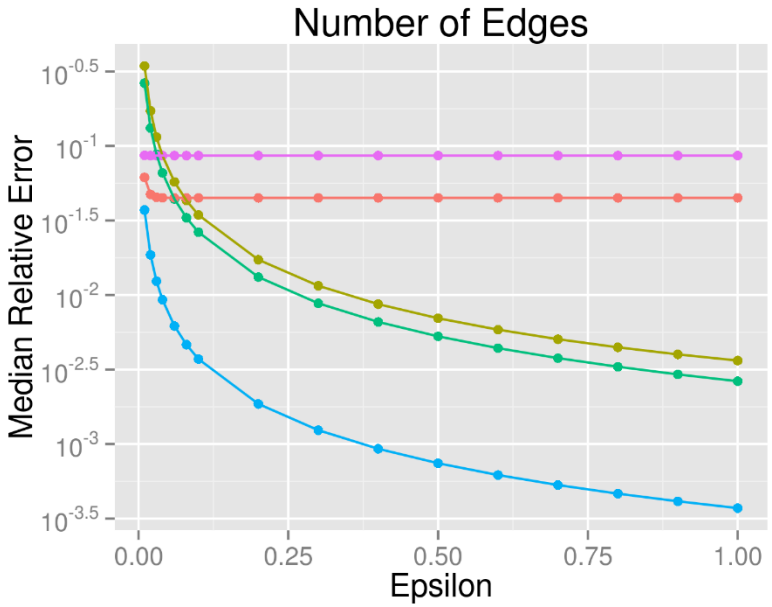
(can be viewed as an efficient construction of Lipschitz extensions)

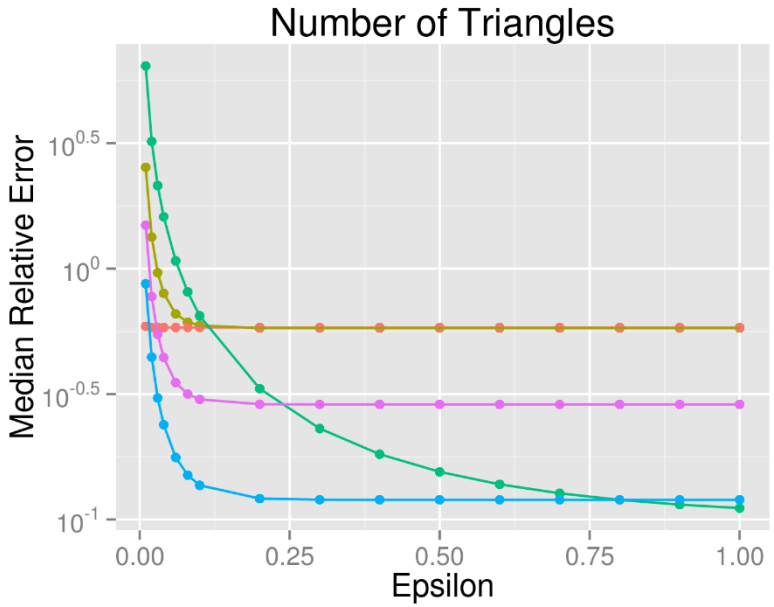
– Releasing (generalization of) subgraph counts

– The same accuracy as (2), but slower

Experimental evaluation

Experiments for the flow and LP method [Lu]





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.
- Longer running times than in flow- and LP-based methods implemented by [Lu] for node-DP algorithms.

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:
  - To which other queries do they apply?
- 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")