Graph Analysis with Node Differential Privacy

Sofya Raskhodnikova

Penn State University

Joint work with Shiva Kasiviswanathan (GE Research),
Kobbi Nissim (Ben-Gurion U. and Harvard U.),
Adam Smith (Penn State)

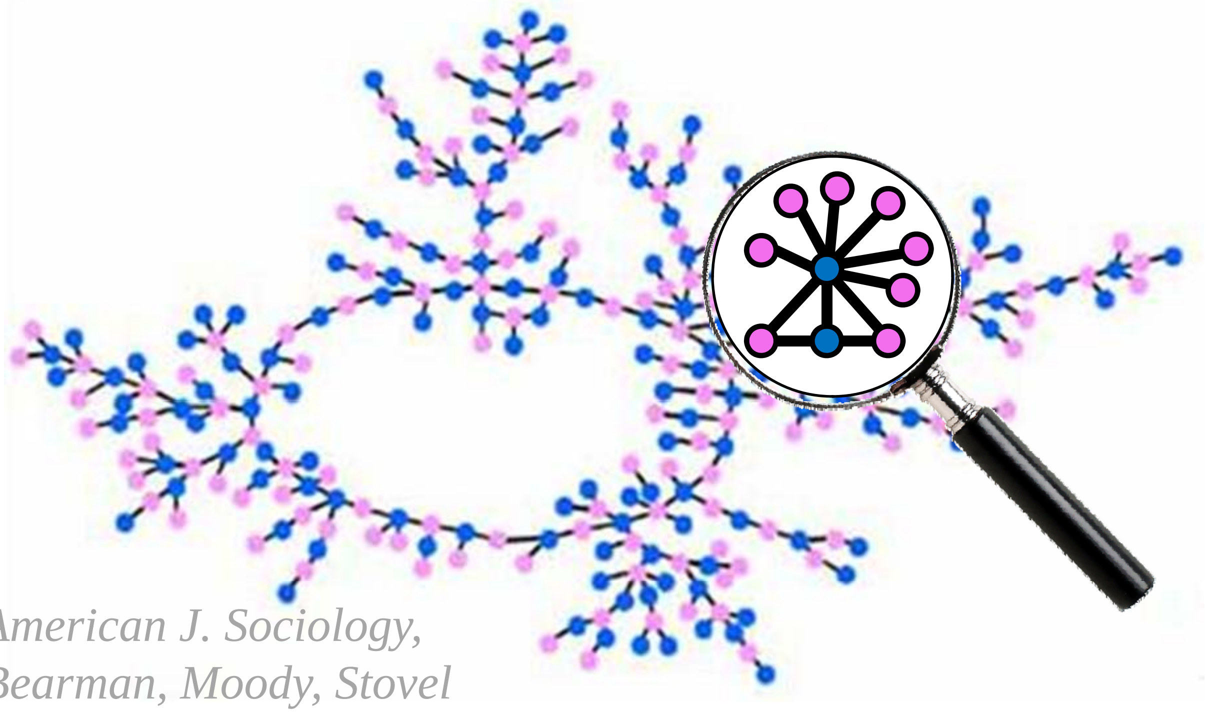
PENNSTATE
1855

Publishing information about graphs

Many datasets can be represented as graphs

- **“Friendships” in online social network**
- **Financial transactions**
- **Email communication**
- **Romantic relationships**

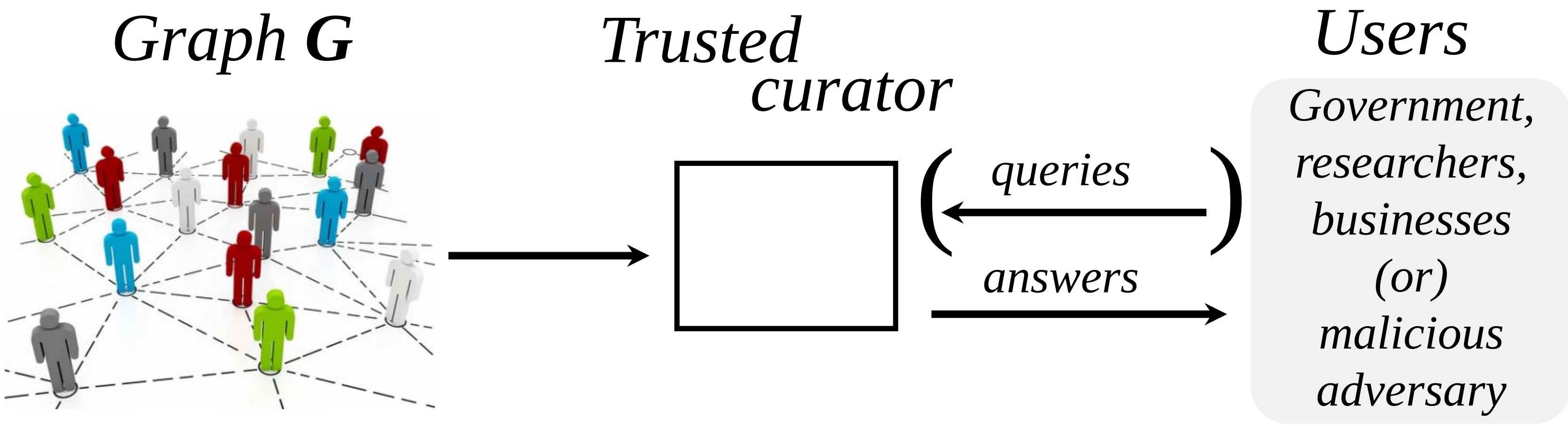
image source http://community.expressor-software.com/blogs/mtarallo/36-extracting-data-facebook-social-graph-expressor-tutorial.html



American J. Sociology, Bearman, Moody, Stovel

Privacy is a big issue!

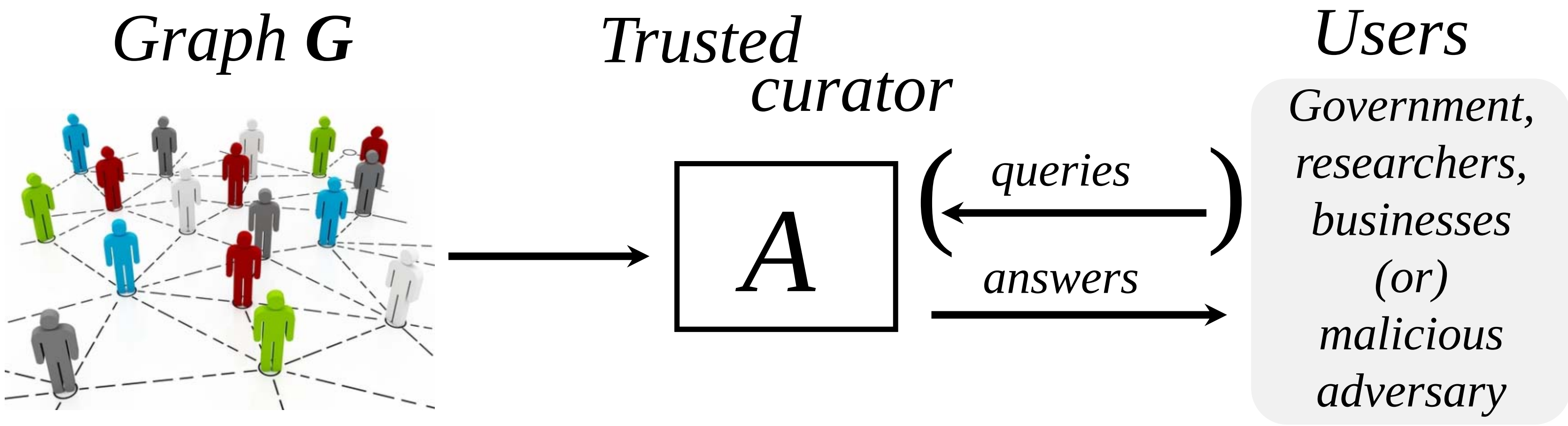
Private analysis of graph data



- **Two conflicting goals:** utility and privacy
  - utility: accurate answers
  - privacy: ?

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

Differential privacy for graph data



- **Intuition:** neighbors are datasets that differ only in some information we'd like to hide (e.g., one person's 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:

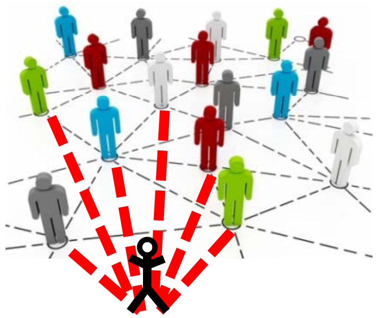
G′:

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

- **Node** differential privacy

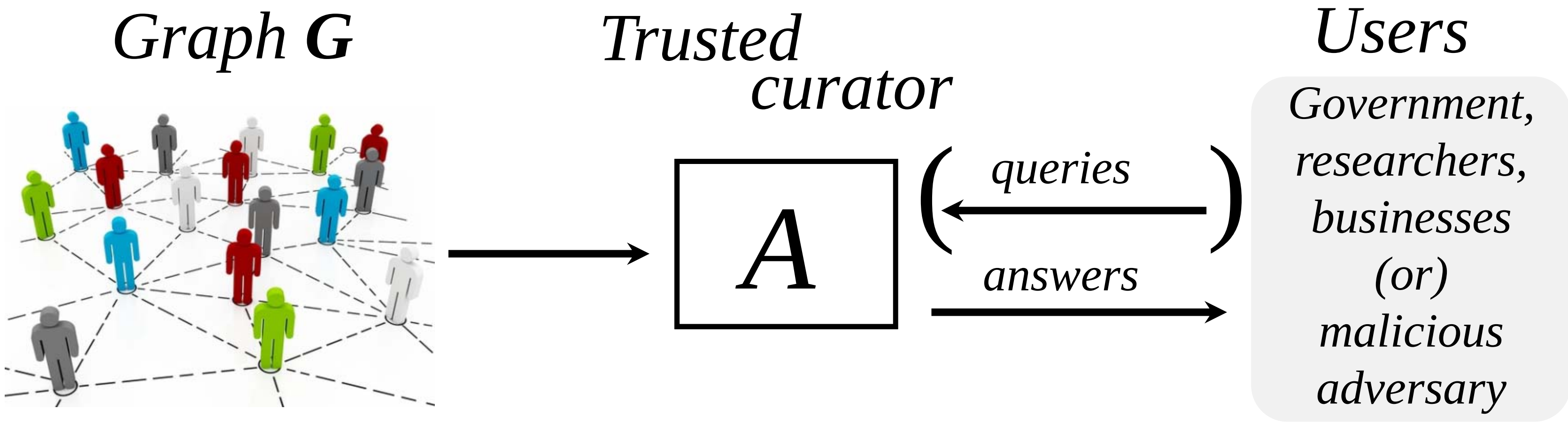
G:

G′:



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

Node differentially private analysis of graphs



- **Two conflicting goals:** utility and privacy
  - Impossible to get both in the worst case

- **Previously:** no node differentially private algorithms that are accurate on realistic graphs

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

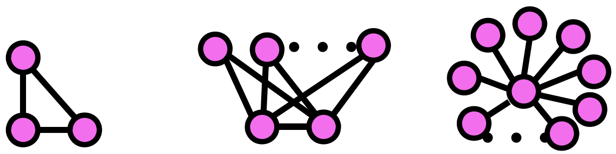
Our contributions

- First node differentially private algorithms that are accurate for sparse graphs
  - **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**
- Techniques for node differentially private algorithms
- Methodology for analyzing the accuracy of such algorithms on realistic networks

Concurrent work on node privacy [Blocki Blum Datta Sheffet 13]

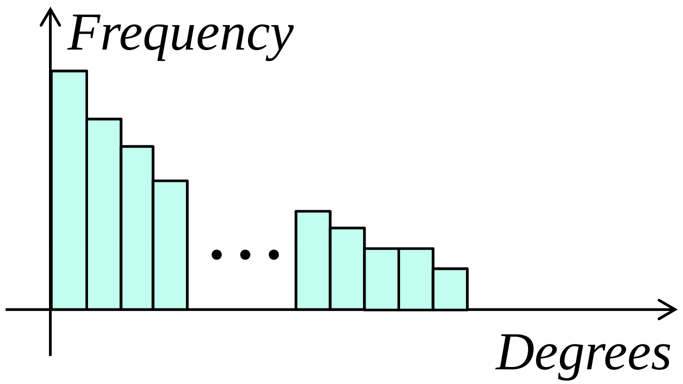
Our contributions: algorithms

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



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

  - degree distribution



Our contributions: accuracy analysis

- Accuracy analysis of our algorithms for graphs with not-too-heavy-tailed degree distribution
  - number of edges
  - counts of small subgraphs (e.g., **triangles**, ***k*-triangles**, ***k*-stars**)

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

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

Previous work on

differentially private computations on graphs

Edge differentially private algorithms

- **number of triangles**, **MST cost** [Nissim Raskhodnikova Smith 07]
- **degree distribution** [Hay Rastogi Miklau Suciu 09, Hay Li Miklau Jensen 09]
- **small subgraph counts** [Karwa Raskhodnikova Smith Yaroslavtsev 11]
- **cuts** [Blocki Blum Datta Sheffet 12]

Edge private against Bayesian adversary (*weaker* privacy)

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

Node zero-knowledge private (*stronger* privacy)

- **average degree, distances to nearest connected, Eulerian, cycle-free graphs for dense graphs** [Gehrke Lui Pass 12]

Differential privacy basics

Graph ***G*** *Trusted curator* *Users*

A

statistic f

approximation to f(G)

Government, researchers, businesses (or) malicious adversary

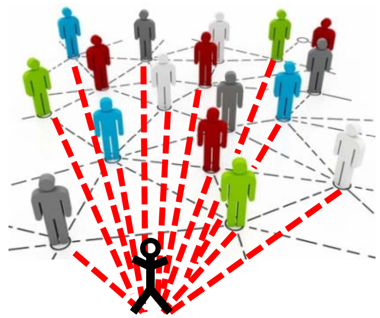
How accurately

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

Global sensitivity framework [DMNS'06]

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

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



- For every function f, there is an ϵ-differentially private algorithm that w.h.p. approximates f with additive error $\frac{\partial f}{\epsilon}$.

- **Examples:**

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

➢ $f_{\triangle}$(G) is the number of triangles in G. $\partial f_{\triangle} = \binom{n}{2}$.

“Projections” on graphs of small degree

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

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

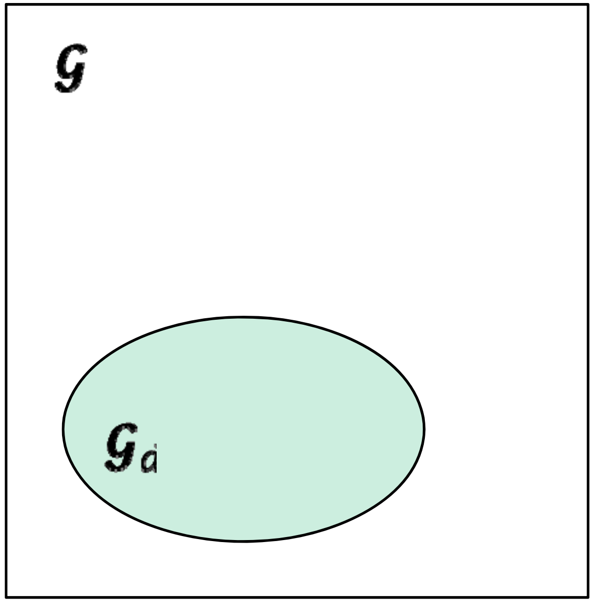
Notation. ∂f = global sensitivity of f over $\mathcal{G}$.

$\partial_d f$ = 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 $\partial f_{\triangle} = \binom{n}{2}$)



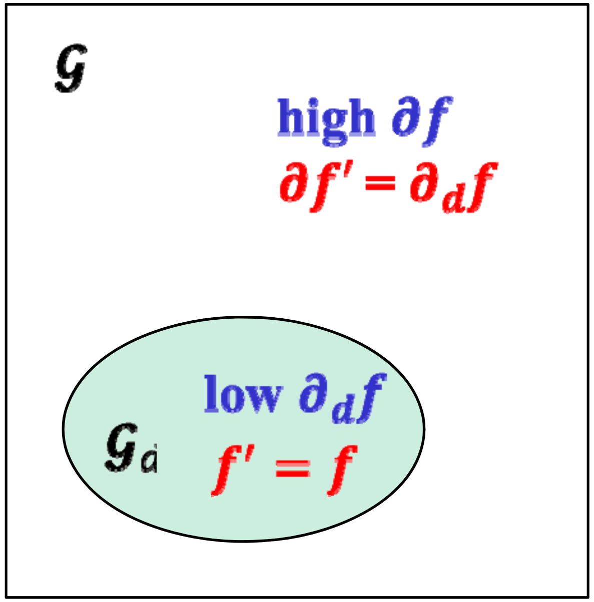
Goal: privacy for all graphs

Method 1: Lipschitz extensions

A function f' is a **Lipschitz extension**

of f from $\mathcal{G}_d$ to $\mathcal{G}$ if

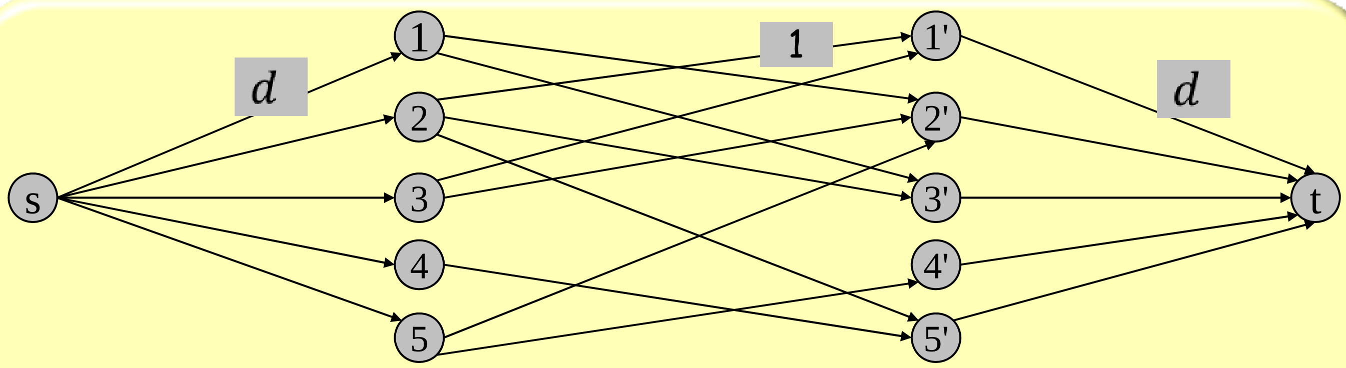
- f' agrees with f on $\mathcal{G}_d$ *and*
- $\partial f' = \partial_d f$



- Release f' via GS framework [DMNS'06]
- Requires designing Lipschitz extension for each function f
  - we base ours on maximum flow and linear and convex programs

Lipschitz extension of f_-: flow graph

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



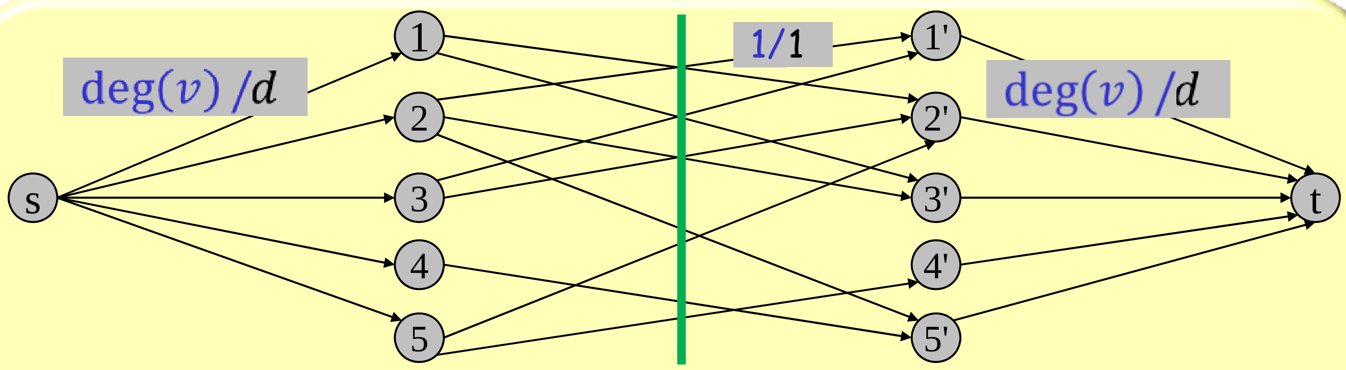
Add edge (u, v') iff $(u, v) \in E$.

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_-: flow graph

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



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

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

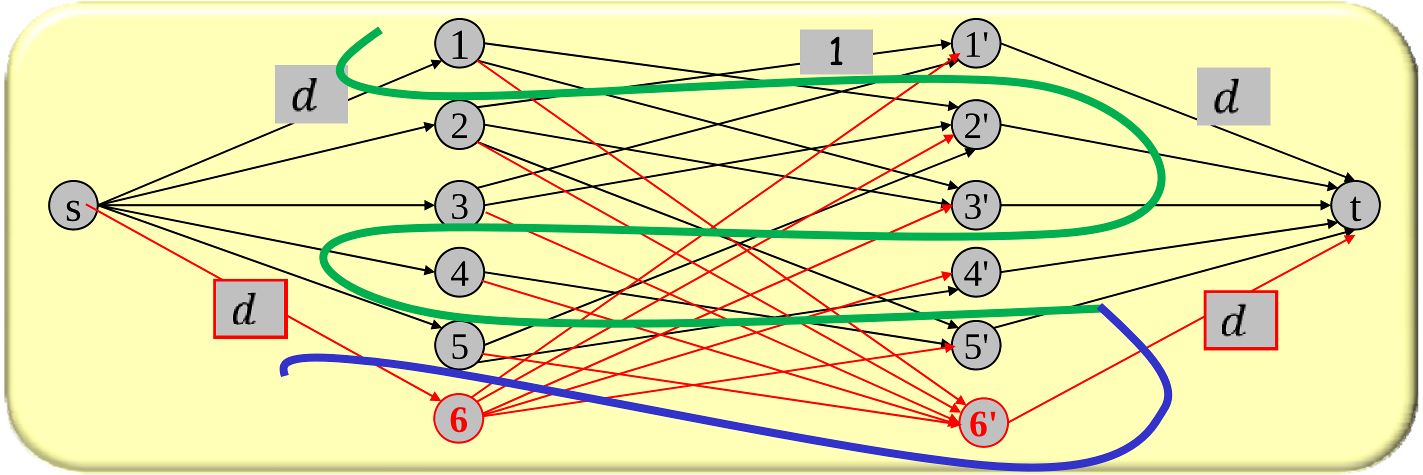
Lemma. $v_{\mathbf{flow}}(G)/2$ is a Lipschitz extension of f_-.

Proof: **(1)** $v_{\mathbf{flow}}(G) = 2f_-(G)$ for all $G \in \mathcal{G}_d$

(2) $\partial v_{\mathbf{flow}} = 2 \cdot \partial_d f_-$

Lipschitz extension of f_-: flow graph

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



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

Lemma. $v_{\mathbf{flow}}(G)/2$ is a Lipschitz extension of f_-.

Proof: (1) $v_{\mathbf{flow}}(G) = 2f_-(G)$ for all $G \in \mathcal{G}_d$

(2) $\partial\, v_{\mathbf{flow}} = 2 \cdot \partial_d f_- = \mathbf{2d}$

Method 2: Generic reduction to privacy over $\mathcal{G}_d$

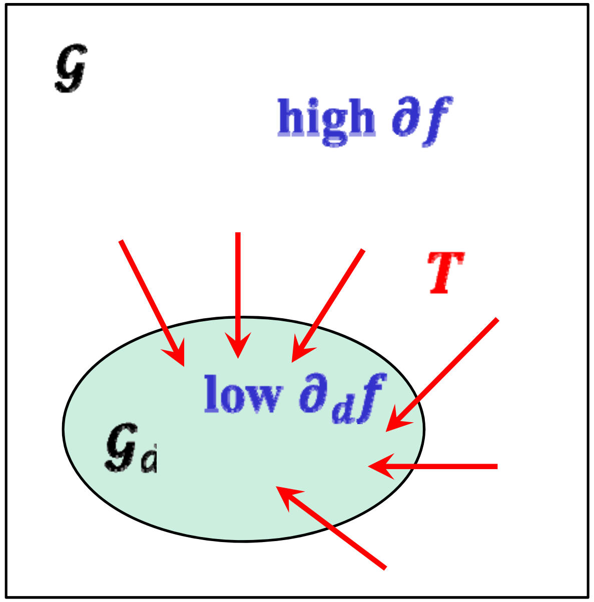
Input: Algorithm B that is node-DP over $\mathcal{G}_d$

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 Smooth Sensitivity framework [NRS'07]



Our results

- Node differentially private algorithms for releasing
  - number of edges
  - counts of small subgraphs (e.g., **triangles**, ***k*-triangles**, ***k*-stars**)
  - degree distribution

} **via Lipschitz extensions** (number of edges; counts of small subgraphs)

} **via generic reduction** (degree distribution)

Conclusions

- It is possible to design node differentially private algorithms with good utility on sparse graphs
  - One can first test whether the graph is sparse privately
- Directions for future work
  - Node-private synthetic graphs
  - What are the right notions of privacy for network data?

Lipschitz extensions via linear/convex programs

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

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

$$0 \leq x_T \leq 1 \qquad \text{for all triangles } T$$

$$\sum_{T:v\in V(T)} x_T \leq \binom{d}{2} = \Delta_d f_\triangle \qquad \text{for all nodes } v$$

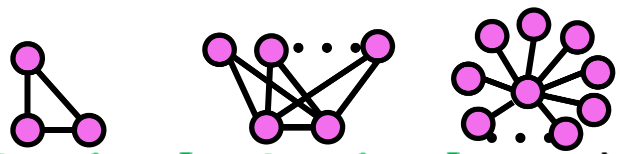
v_{LP}**(G)** is the value of **LP**.

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

- Can be generalized to other counting queries
- Other queries use convex programs

Our results

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

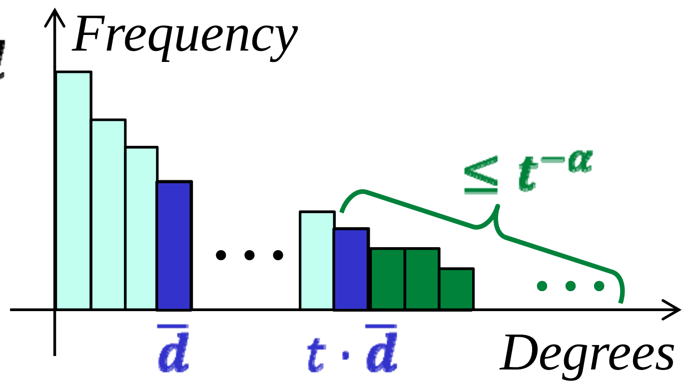


(e.g., triangles, k-triangles, k-stars)

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

Notation: $\bar{d} =$ average degree

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

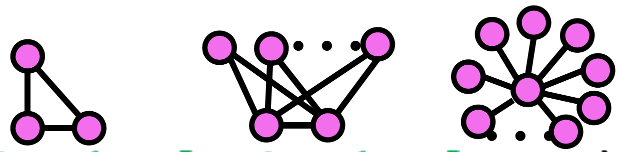


A graph G satisfies α-decay if for all $t > 1$: $P(t \cdot \bar{d}) \leq t^{-\alpha}$

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

Our results

- 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 $\alpha > 1$

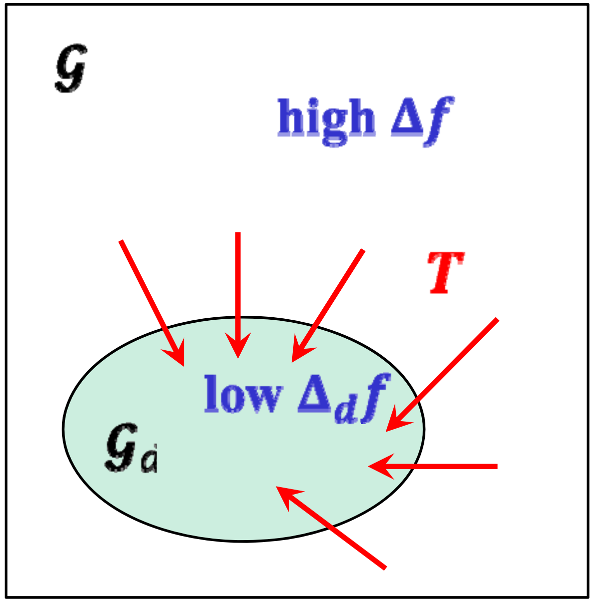
A graph G satisfies **α-decay** if for all $t > 1$: $P(t \cdot \bar{d}) \le t^{-\alpha}$

  - number of edges
  - counts of small subgraphs (e.g., **triangles**, ***k*-triangles**, ***k*-stars**) } **(1+o(1))-approximation**
  - degree distribution } $\left\|\mathbf{A}_{\epsilon,\alpha}(\mathbf{G}) - \mathbf{DegDistrib}(\mathbf{G})\right\|_1 = \mathbf{o}(1)$

Method 2: Generic reduction to privacy over $\mathcal{G}_d$

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

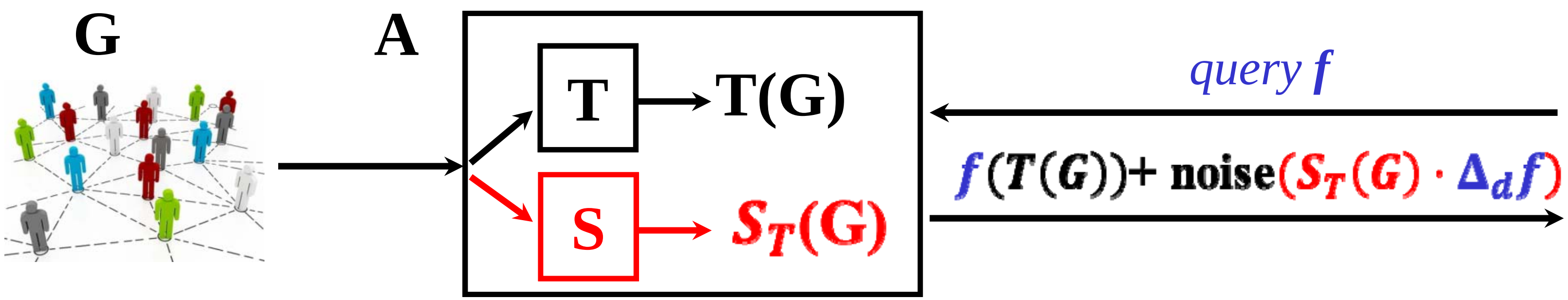
Output: Algorithm A that is node-DP over $\mathcal{G}$,
has accuracy similar to B on "nice" graphs



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

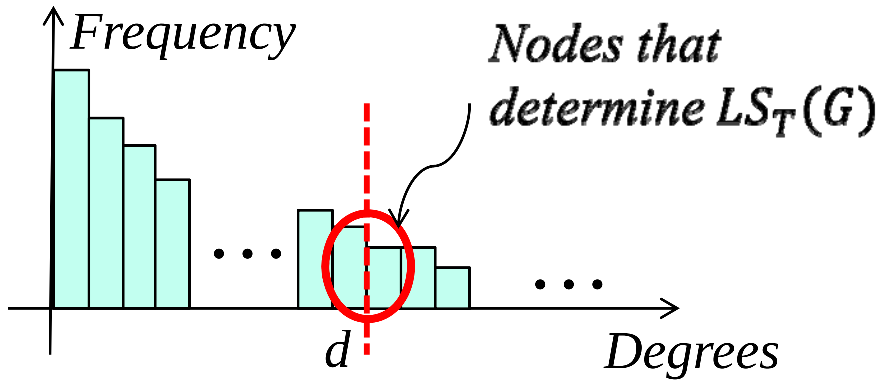
How it works: Truncation T(G) outputs G with nodes of degree $> d$ removed.

- Answer queries on T(G) instead of G
  - via Smooth Sensitivity framework [NRS'07]
  - via finding a DP upper bound ℓ on $LS_T(G)$ [Dwork Lei 09, KRSY'11] and running any algorithm that is $\left(\frac{\epsilon}{\ell}\right)$-node-DP over $\mathcal{G}_d$



Generic Reduction via Truncation

- **Truncation T(G)** removes nodes of degree $> d$.
- On query f, answer $A(G) = f\big(T(G)\big) + noise$



How much noise?

Local sensitivity of T as a map $\{graphs\} \to \{graphs\}$

$dist(G, G') = \#(node\ changes\ to\ go\ from\ G\ to\ G')$

$$LS_T(G) = \max_{G':\ \text{neighbor of } G} dist\big(T(G), T(G')\big)$$

Lemma. $LS_T(G) \le 1 + \max\,(n_d\,, n_{d+1})$,

where n_i= #{nodes of degree i}.

Global sensitivity $\max_G LS_T(G)$ is too large.

Smooth Sensitivity of Truncation

Smooth Sensitivity Framework [NRS '07]

$S_f(G)$ is a **smooth bound on local sensitivity** of f if

- $S_f(G) \geq LS_f(G)$
- $S_f(G) \leq e^{\epsilon} S_f(G')$ for all neighbors G and G'

Lemma.

$$S_T(G) \leq \max_{k \geq 0} e^{-\epsilon k}\left(1 + \#\left\{nodes\ of\ degree\ \left(d \pm (k+1)\right)\right\}\right)$$

is a smooth bound **for T**, computable in time $O(m+n)$

"**Chain rule**": $S_f(G) = S_T(G) \cdot \Delta_d f$ is smooth **for $f \circ T$**

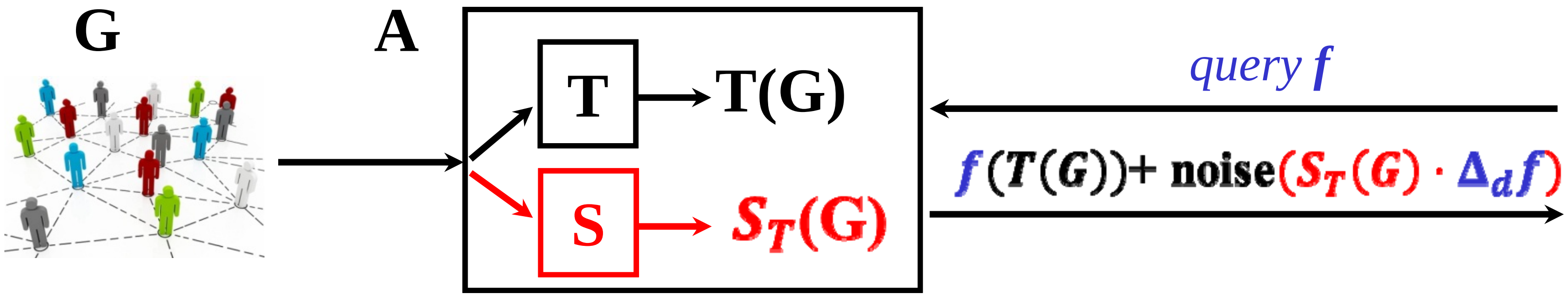
Lemma. $(\forall G, d)$ If we truncate to a *random* degree in $[2d, 3d]$,

$$E[S_T(G)] \leq (P(d)n)\frac{3 \log n}{\epsilon d} + \frac{1}{\epsilon} + 1$$

#(nodes of degree above d)

Utility: *If G is d-bounded, add noise* $O(\Delta_{3d} f / \epsilon^2)$

Releasing Degree Distribution via Generic Reduction



- Application: Releasing the degree distribution

Theorem: There exists a node-DP algorithm A such that

$$\left\| A_{\epsilon,\alpha}(G) - DegDistrib(G) \right\|_1 = o(1)$$

with prob. at least $^2/_3$ if G satisfies α-decay for $\alpha > 2$.