Graph Analysis with Node-Level Differential Privacy

Shiva Kasiviswanathan† Kobbi Nissim ‡ Sofya Raskhodnikova* Adam Smith*

† GE Research ‡ Ben Gurion U. + MSR *The Pennsylvania State University

PENNSTATE 1855

Privacy for Network Data

Many datasets can be represented as graphs

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

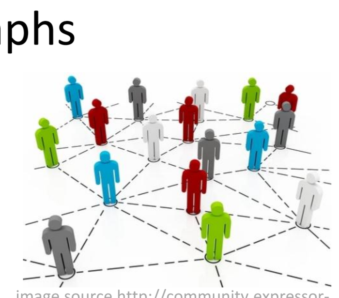
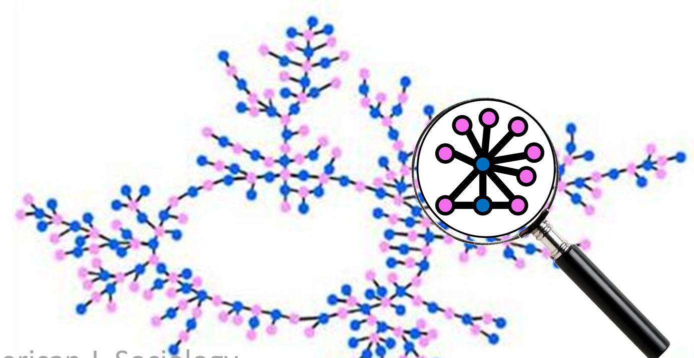


image source http://community.expressor-software.com/blogs/mtarallo/36-extracting-data-facebook-social-graph-expressor-tutorial.html



American J. Sociology, Bearman, Moody, Stovel

Privacy is a big issue!

Differential Privacy for Graph Data

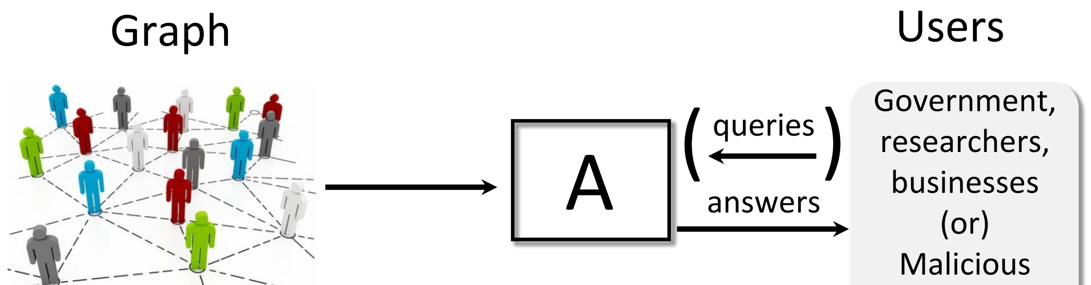


image source http://www.queticointernetmarketing.com/new-amazing-facebook-photo-mapper/

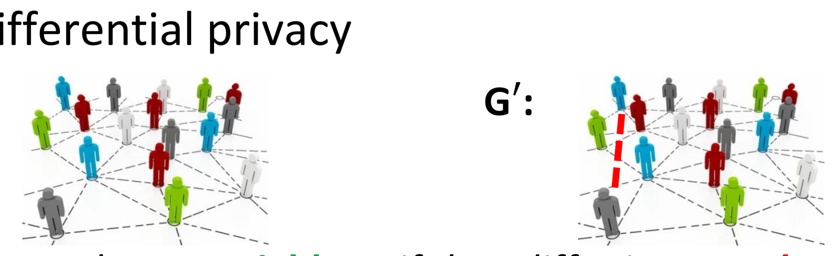
Differential privacy [Dwork McSherry Nissim Smith 06]

An algorithm A is ϵ-differentially private if for all pairs of **neighbors** G, G' and all sets of answers **S**:

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

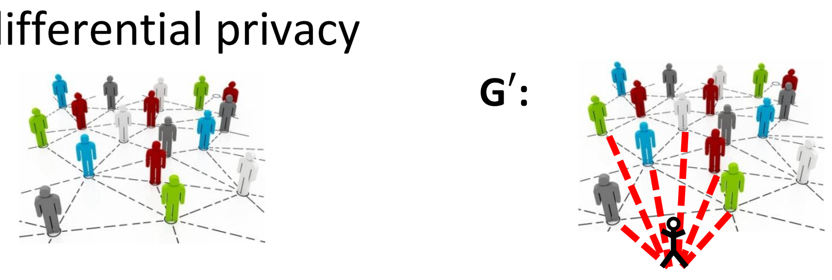
Two Notions of Neighbors

- **Edge** differential privacy



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

- **Node** differential privacy



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

Our Contributions

- First node differentially private algorithms that are accurate for sparse graphs
  - **private *for all*** graphs
  - **accurate *for a subclass*** of graphs, which includes
    - graphs with known (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

Independent work on node privacy: [Blocki,Blum,Datta,Sheffet]

Prior Work on DP 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 12]

Edge private against Bayesian adversary (*weaker* privacy)

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

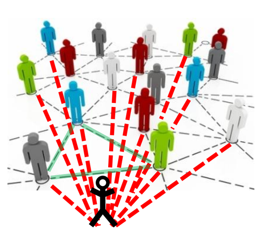
Edge zero-knowledge private (*stronger* privacy)

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

Our Techniques

Challenge with Node Privacy: High Local Sensitivity

- **Local sensitivity** [NRS'07]:
  $LS_f(G) = \max_{G':\, \text{neighbor of } G} |f(G) - f(G')|$
- **Global sensitivity** [DMNS'06]:
  $GS_f = \max_G LS_f(G)$



For many functions f of the data, node $LS_f(G)$ is high.

- Consider adding a node connected to all other nodes.
- **Examples:**

➢ $f_-(G) = |E(G)|$. Edge GS_{f_-} is 1; node $LS_{f_-}(G)$ is n for all G.

➢ $f_\Delta(G)$= # of Δs in G. Edge GS_{f_Δ} is n; node $LS_{f_\Delta}(G)$ is $|E(G)|$.

"Projections" on Graphs of Small Degree

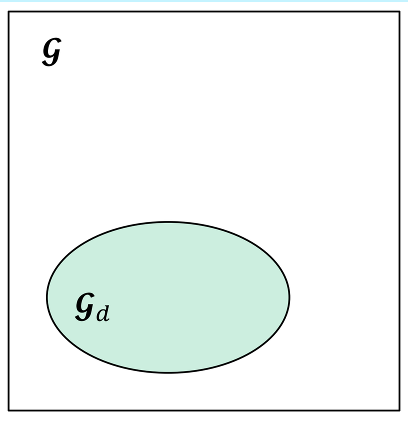
Let $\mathcal{G}$ = family of all graphs,

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

Notation. Δf = node GS_f over $\mathcal{G}$.

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

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



Examples:

➢ $\Delta_d f_- = d$ (compare to $\Delta f_- = n$)

➢ $\Delta_d f_\Delta = \binom{d}{2}$ (compare to $\Delta f_\Delta = |E|$)

Goal: privacy for all graphs

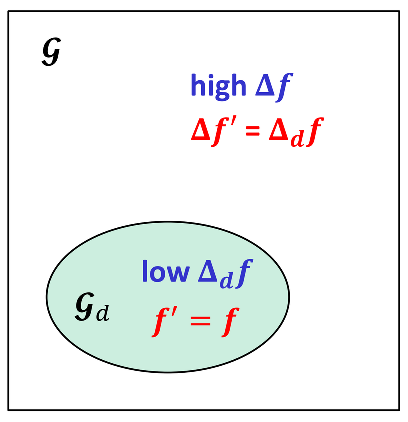
Idea: ``Project'' on graphs in $\mathcal{G}_d$ for a carefully chosen d << n.

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

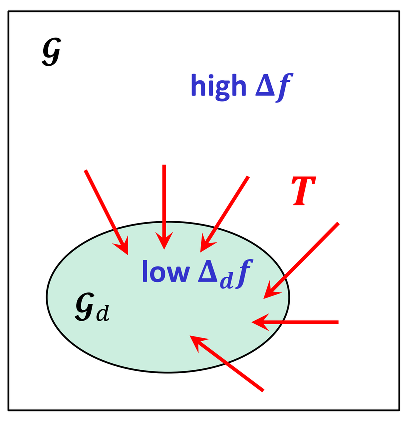
➢ $\Delta f' = \Delta_d f$



- 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

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 $\mathcal{G}_d$

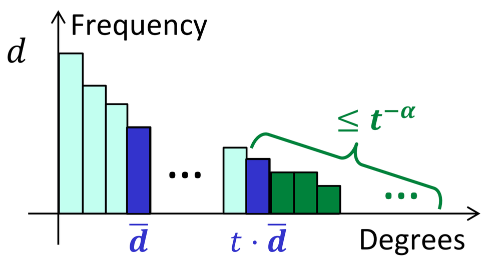
Our Results

- Node differentially private algorithms for releasing
  - number of edges
  - counts of small subgraphs (e.g., **triangles**, **k-triangles**, **k-stars**) } **via Lipschitz extensions**
  - Degree distribution } **via generic reduction**
- 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 $\ge d$

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

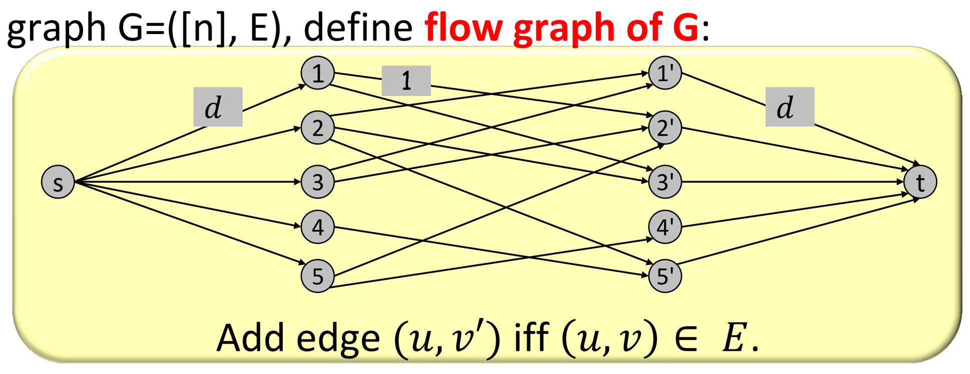


- Every graph satisfies 1-decay
- Natural graphs (e.g., **"scale-free" graphs**, **Erdos-Renyi**) satisfy $\alpha > 1$

Obtaining Lipschitz Extensions

Lipschitz Extension of f_- via Flow Graph

For a graph G=([n], 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 Extensions via Linear and Convex Programs

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

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

$$0 \le x_T \le 1 \quad \text{for all triangles } T$$

$$\sum_{T:\, v \in V(T)} x_T \le \Delta_d f_\Delta \quad \text{for all nodes } v$$

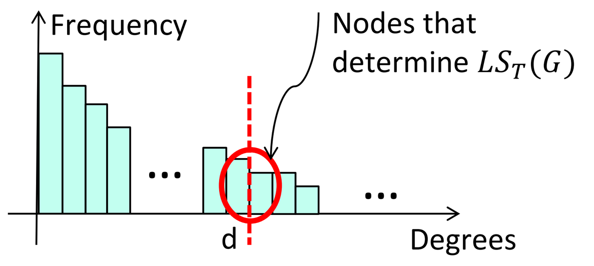
v_{LP}**(G)** is the value of **LP**.

Lemma. v_{LP}**(G)** is a Lipschitz extension of f_Δ.

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

Generic Reduction (via Smooth Sensitivity)

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



How much noise?

- Look at **local sensitivity** of T as a map $\{graphs\} \to \{graphs\}$
  - $dist(G, G') = \#(node\ changes\ to\ go\ from\ G\ to\ G')$

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

Lemma. $LS_T(G) = 1 + |\{nodes\ of\ degree\ d\ or\ d+1\}|$

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

Smooth Sensitivity Framework [NRS '07]

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

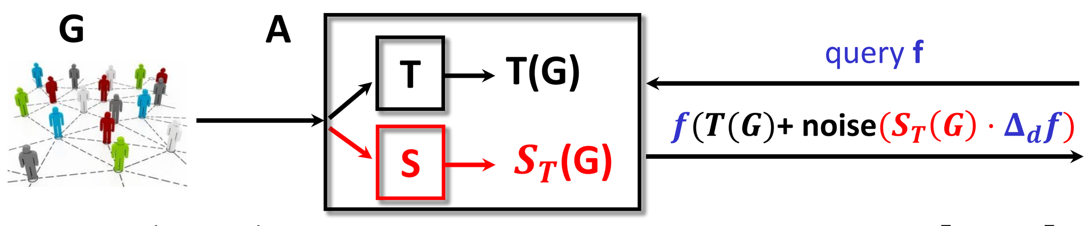
- $S_f(G) \le LS_f(G)$
- $S_f(G) \le e^{\epsilon} S_f(G')$ for all neighbors G and G'

Lemma.

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

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 $[d, 2d]$,

$$E[S_T(G)] \le \underbrace{(P(d)n)}_{\#(\text{nodes of degree above } d)} \frac{2 \log n}{\epsilon d} + \frac{1}{\epsilon} + 1$$

If G is d-bounded, add noise $O(\Delta_f / \epsilon^2)$

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 $\alpha > 1$.

Conclusions

- First nontrivial node-private algorithms for sparse graphs
- Technique: projections onto graphs of small degree

NSF