TPDP 2024

The Power of Graph Sparsification in the Continual Release Model

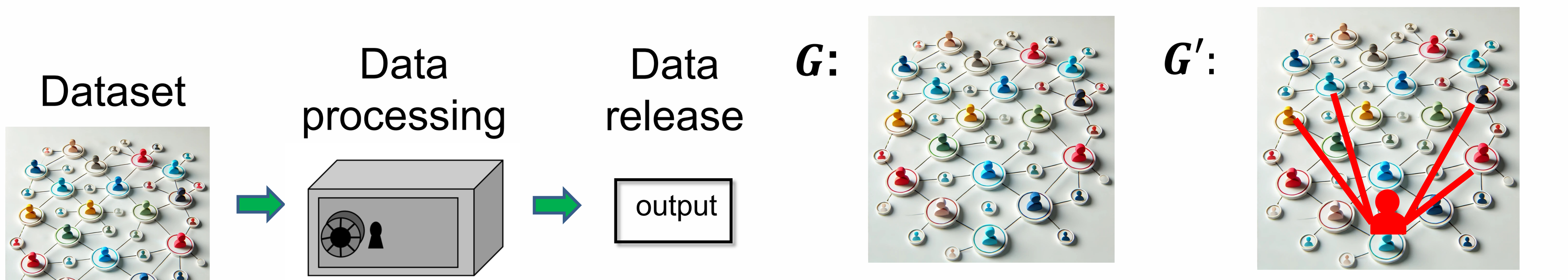
*Alessandro Epasto[1], Quanquan C. Liu[2], Tamalika Mukherjee[3], **Felix Zhou[2]***

[1] Google Research aepasto@google.com

[2] Yale University {quanquan.liu, felix.zhou}@yale.edu

[3] Columbia University tm3391@columbia.edu

Differential Privacy



Two graphs are **edge/node neighbors** if they differ in ***one edge or node, resp.***

Differential privacy [Dwork McSherry Nissim Smith '06, Nissim Raskhodnikova Smith '07]

An algorithm A is **(ϵ, δ)-differentially private** if for all pairs of **neighbors** G, G' and all possible sets of outputs S:

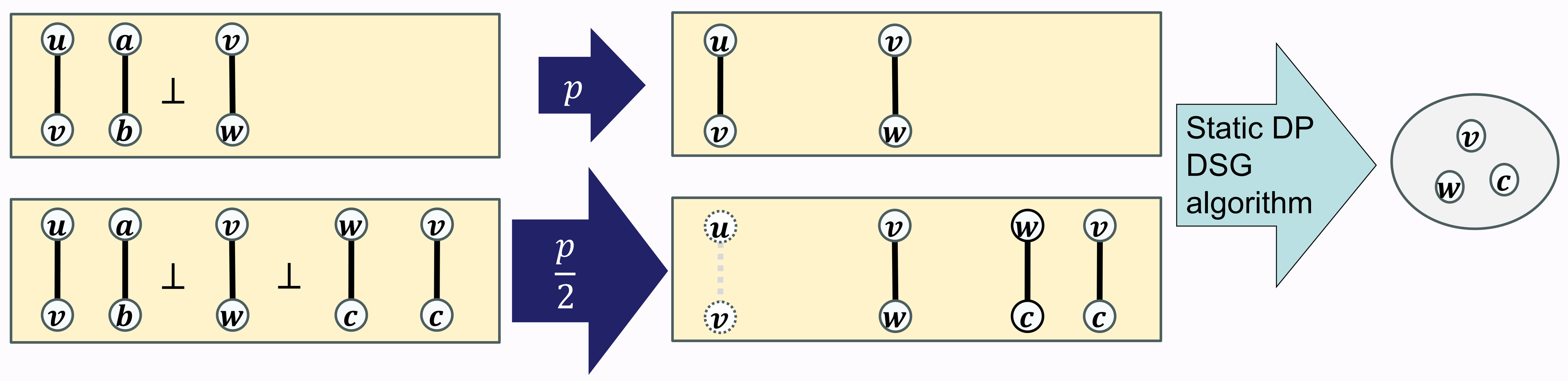
$$\Pr[A(G) \in S] \le e^{\epsilon} \Pr[A(G') \in S] + \delta$$

Results for Insertion-Only Graphs

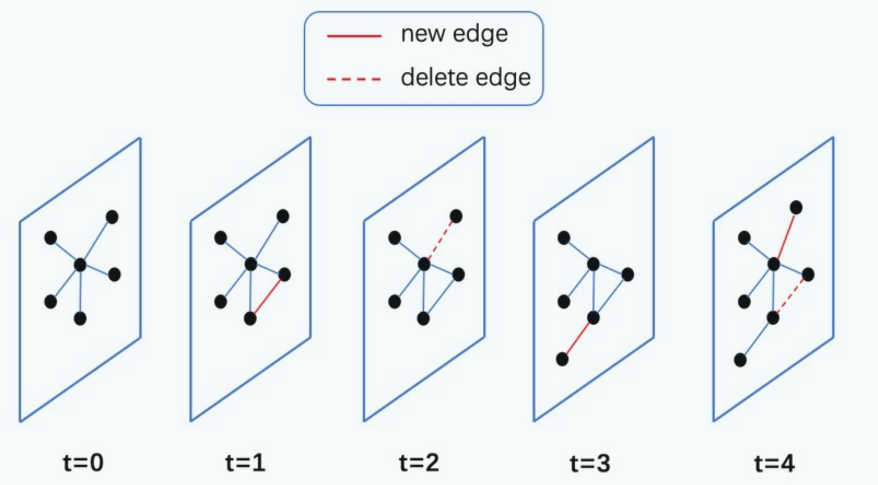
Problem		Previous Results	Our Results
Densest Subgraph	Error	$\left(1+\eta, \frac{\log^2 n}{\varepsilon}\right)$ (value only)	$\left(1+\eta, \frac{\log^5 n}{\varepsilon}\right)$
	Space	$\Theta(m)$	$\tilde{O}\left(\frac{n}{\varepsilon}\right)$
Maximum Matching Size	Error	$\left(1+\eta, \frac{\log^2 n}{\varepsilon}\right)$	$\left((1+\eta)(2+\tilde{\alpha}), \frac{\log^3 n}{\varepsilon}\right)$ (arboricity $\tilde{\alpha}$)
	Space	$\Theta(m)$	$O\left(\frac{\mathrm{poly}(\log n)}{\varepsilon}\right)$

Sublinear Space Densest Subgraph (DSG)

- Uses uniform sampling [McGregor Tench Vorotnikova Vu '15; Esfandiari Hajiaghayi Woodruff '16]
  - ➢ Sample each edge with $p \approx \frac{n \log n}{m_t}$, obtain $O(n \log n)$ sized sample, rescale sampled solution by $\frac{1}{p}$
- Need to **adaptively** choose sampling probability p to release **answer after every update**
  - ➢ Use Sparse Vector Technique (SVT) to **halve** sampling probability when edge count **doubles**
- Ensure $O\left(\frac{\log n}{\varepsilon}\right)$ **additive error** from static private algorithm does not compound when **rescaling**
  - ➢ Ensure returned subgraph has sufficiently high density, absorb additive error as multiplicative error
- Approximation guarantee follows from intricate **Chernoff bound** argument
  - ➢ accounts for **SVT / static algorithm errors**
- ε-DP guarantee from DP of SVT, edge edit distance being preserved, and composition



Insertion-Only Graph Continual Release Model



[Dwork Naor Pitassi Rothblum '10], [Chan Shi Song '11]

- Edges arrive in a stream of T edge addition / empty updates
- Two graph streams are neighboring if differ by one update
- Required to return private output after **each** update

Note: preserve privacy of entire stream of T outputs

Prior Work

- Continual release of numerical graph statistics [Song Little Mehta Vinterbo Chaudhuri '18; Fichtenberger Henzinger Ost. '21; Jain Smith Wagaman '24]

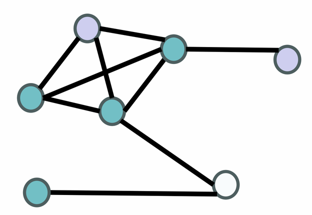
Prior continual release techniques can only return **value of solution**

Prior continual release works compute **exact** graph statistics and require storing **entire graph**

- Online streaming algorithms (non-private)
  - Independent set [Halldorrsson Halldorsson Losievkaja Szegedy '16; Cormode Dark Konrad '18]
  - Dominating set, matching [Chen Chitnus Eades Wirth '23]
  - Edge coloring [Ghosh Stoeckl '23]
- Static private algorithms
  - Densest subgraph, ***k***-Core decomposition [Dhulipala Liu Raskhodnikova Shi Shun Yu '22; Dhulipala Li Liu '23; Dinitz Kale Lattanzi Vassilvitskii '24; Henzinger Sricharan Zhu '24]

One-Shot Implicit Node DP Vertex Cover via Sparsification

- Two graph streams are neighboring if differ by **all** edge updates adjacent to a **single vertex**
- Previous work requires **(nearly) bounded degree** graph streams to output values of solutions [Song Little Mehta Vinterbo Chaudhuri '18; Fichtenberger Henzinger Ost '21; Jain Smith Wagaman '24]
- Based on bounded degree sparsifiers [Solomon '21]
  - ➢ Store incoming edge if both endpoints have degree $\le O(\tilde{\alpha})$
- Use static private implicit vertex cover algorithm [Gupta Ligett McSherry Roth Talwar '10]



Implicit Vertex Cover releases information so every edge knows which vertex covers it

	Our Result
Error	$\left(3+\eta+O\left(\frac{\tilde{\alpha}}{\varepsilon}\right), O\left(\frac{\tilde{\alpha}\log n}{\varepsilon}\right)\right)$
Space	$O(n\tilde{\alpha})$ (arboricity $\tilde{\alpha}$)

Sublinear Space via Sparsification

Graph sparsification computes **a smaller subgraph** of input graph

Determine property of sparsified graph as **approximate answer** of original graph

Process Edge Update Meta-Algorithm (Input: edge e)

1. Keep or discard e using problem specific (randomized) sparsification procedure
2. If <u>value</u> of optimal solution in <u>sparsified</u> graph did not change significantly, skip to 4.
3. Else use static private algorithm to compute new solution in <u>sparsified</u> graph
4. **Return** last computed solution

Sparsification needs to approximately preserve (coupled) edge edit distance between neighboring graphs

Lazy updates to avoid $O(T)$ error from releasing new solution per update

Need to ensure error from static private algorithm does not blowup in original graph

Fully Dynamic Lower Bounds

Problems	Previous Results	Our Results
Matching size, triangle count, connected components	$(1, \Omega(\log T))$	$\left(1, \Omega\left(\min\left(\sqrt{\frac{n}{\varepsilon}}, \frac{T^{1/4}}{\varepsilon^{3/4}}\right)\right)\right)$

- Based on reduction to inner product queries
- Encode secret dataset within initial graph
- Use edge addition/deletions to simulate multiple inner product queries of the secret database

SCAN ME