Project Report -I

May 11, 2018

1 Project Description

1.1 Graph Differential Privacy

There are two types graph differential privacy.

1. **Edge-differential privacy**: In this setting two graphs are neighboring if they differ on a single edge.
2. **Node- differential privacy**: Here, two graphs are called neighbors if one can be obtained from the other by re- moving a node and its adjacent edges.

Given a privacy parameter $\epsilon > 0$, an algorithm A is ϵ node/edge differentially private if for all neighbor graphs G and G'[neighbor definition follows which of edge-DP or node-DP is being used] and for all sets S of possible outputs produced by A:

$$Pr[A(G) \subseteq S] \leq e^{\epsilon} \cdot Pr[A(G) \subseteq S] \tag{1}$$

Node privacy is a strictly stronger guarantee than edge privacy and as a result much more difficult. One challenge is that for many natural statistics, node privacy is impossible to achieve while getting accurate answers in the worst case. The problem, roughly, is that node-private algorithms must be robust to the insertion of a new node in the graph, but the properties of a sparse graph can be altered dramatically by the insertion of a well-connected node. For example, for common graph statistics – the number of edges, the frequency of a particular subgraph – the change can overwhelm the value of the statistic in sparse graphs. This is so because removing a node, in the worst case can remove n-1 edges [n is the total number of nodes] thereby giving rise to high sensitivity.

There are broadly two approaches towards ensuring node-DP solutions The main idea is to "project" the input graph onto the set of graphs with maximum degree below a certain threshold [15, 8, 3, 13]. The benefits of this approach are two-fold. First, node privacy is easier to achieve in bounded- degree graphs since the insertion of one node affects only a relatively small part of the graph. Tech-nically, the sensitivity of a given query function may be much lower when the function is restricted to graphs of a given degree. Second, for realistic networks this transformation loses relatively little information when the degree threshold is chosen carefully. The difficulty with this approach is that the projection itself may be very sensitive to a change of a single node in the original graph. This difficulty is handled via two different techniques.

1. **Lipschitz Extensions** A function f' is a Lipschitz extension of a function f from G_d to G if

   (a) f'=f for G_Θ

   (b) global sensitivity of f over G_Θ = global sensitivity of f' over G

   where G_Θ is the group of Θ-degree bounded graphs. This is useful for a certain class of statistics. These operators can be viewed as giving a fractional (low-degree) graph that is a solution to a convex optimization problem, typically given by a maximum flow instance or linear program.

2. **Truncation** Truncation is a much more general technique, where the nodes of a graph are truncated to bound the degree of the resulting subgraph. Next a smooth bound of the local sensitivity is computed using which efficient reduction can be constructed that allows us to apply any differentially private algorithm for bounded-degree graphs to an arbitrary graph. The smooth bounds on local sensitivity proposed by Nissim et al. is as follows: instead of using the local sensitivity, select noise magnitude according to a smooth upper bound on the local sensitivity,namely,a function S that is an upper bound on LS_f[local sensitivity of f] at all points and such that $ln(S(\cdot))$ has low global sensitivity. The level of smoothness is parameterized by a number β (where smaller numbers lead to a smoother bound) which depends on ϵ. Formally,

   For $\beta > 0$, a function $S : \mathcal{G}_n \mapsto R$ is a β-smooth upper bound on the local sensitivity of f if it satisfies the following requirements:

   $\forall G \in \mathcal{G}_n : S(G) \geq LS_f(G)$ *for all neighbors* $G, G' \in \mathcal{G}_n : S(G) \leq e^{\beta} S(G')$

   where $\mathcal{G}_n$ is the class of all graphs with n nodes. One can add noise proportional to smooth bounds on the local sensitivity using a variety of distributions.

Since the truncation technique is more general and in some sense, more intuitive, we will be first focusing on truncation for our project.

1.1.1 Existing Truncation Techniques

There are four existing truncation techniques as follows.

1. In [10] truncation is done by removing all nodes with degrees larger than Θ; and analyzing the sensitivity of publishing node-degree histograms after truncation is done in two steps.

2. Another projection method is proposed in [1], in which one chooses an arbitrary order among the edges, traverses the edges in this order, and removes each encountered edge that is connected to a node that has degree $> \Theta$.

3. In a recent work for publishing degree distributions under node- DP [14], another projection method was introduced. Given a graph G = (V,E) and a degree bound Θ, the method first constructs a weighted graph as follows. Starting with a source node s and a sink node t, for each node v $\in$ V, it creates two nodes, v_l and v_r, and adds edges (s, v_l) and (v_r, t) with capacity Θ. For every edge $e = (u, v) \in E$, it then adds a new edge (u_l, v_r) with capacity 1. With the weighted graph, the algorithm computes the maximum flow f from s to t, while minimizing $(\Theta - f(s, v_l))2 + (\Theta - f(v_r, t))2$, where f(u,v) is the $v \in V$ flow from u to v. Finally, the algorithm removes s from the maximum flow graph, and, for each node v $\in$ V , obtains the degrees of v_l, and constructs the degree distribution from these degrees.

4. In [4] the algorithm requires a stable ordering of all edges in an input graph G, denoted by $\Lambda(G)$. We say that a graph edge ordering Λ is stable if and only if given two neighboring graphs G = (V,E) and $G' = (V', E')$ that differ by only a node , $\Lambda(G)$ and $\Lambda(G')$ are consistent, in the sense that if two edges appear both in G and G', their relative ordering are the same in $\Lambda(G)$ and $\Lambda(G')$ where . The projection first forms a graph with all nodes in G but no edges, and then keeps inserting edges from Λ following the ordering whenever inserting an edge e = (u, v) will not cause the degree of either u or v larger than Θ.

1.2 Streaming Model differential privacy

There are two broad notions of differential privacy in the streaming model

1. **Pan Privacy** Collectors of confidential data, such as governmental agencies, hospitals, or search engine providers, can be pressured to permit data to be used for purposes other than that for which they were collected. To support the data curators, Dwork et al [5, 10, 2] introduced the notion of pan privacy; roughly speaking, these algorithms retain their privacy properties even if their internal state becomes visible to an adversary.

2. **User Level Privacy** Consider the following scenario- although the probability of selecting a specific individual may be small, once a record belonging to this individual has been stored, it is implausible to deny that the individual's data appear in the data stream. This is where user level privacy comes into play to provide for plausible deniability[5, 2]. In other words, any two streams, one with and the other without all information of a particular individual, will produce very similar distributions on states and outputs, even though the data of an individual are interleaved arbitrarily with other data in the stream.

Consider a source streaming model as follows. We assume the source stream D as an infinite sequence of tuples. Each tuple is of the form (u,s,t) and is an element from the domain $dom = \mathcal{U} \times \mathcal{S} \times \mathcal{T}$ where $\mathcal{U}$ is the set of user identifiers,$\mathcal{S}$ is a set of possible states, and $\mathcal{T}$ is an (infinite) set of timestamps. Each (u, s, t) records an atomic event, namely that user u was observed in state s at time t . There are mainly three types of query in the streaming setting [8]

1. **Queries on a single target state** A counting query takes a specific target state s and reports, for each time step t , the number of users who were observed in state s at time t.
2. **Sliding window** Sliding Windows. A sliding window query with window size w and target state s reports, for each time step t, the total number of times a user has been observed in state s in the most recent w time steps.
3. **Event Monitoring** While each tuple in the stream D captures an atomic event, the analyst may be interested in monitoring certain patterns in the event stream.

2 Tying them together

Graph processing in the data stream model is a widely used approach for handling massive graphs generated from applications with data about both basic entities and the relationships between these entities. As discussed above we see that there exist separate works in differentially private streaming model [5, 10, 2] and differentially private graph algorithms in the classical setting [6, 9, 7]. However, to the best of our knowledge there is no work addressing the privacy concerns of large-scale graph processing in the streaming model. Thus we aim to tie them together and propose differentially private graph algorithms in the streaming data model.

3 Milestones

3.1 Milestone 1

From the aforementioned discussion, we see that the first step is to define whether user level privacy or pan privacy should be the chosen DP guarantee for graph processing algorithms in the streaming model and come up with a suitable definition that translates it to an equivalent node-DP definition for the graph. We have to have a clear understanding of the semantics of the definition in this setting. For eg, is every node equivalent to an user, and the edges correspond to user data and thus can node-DP be a rough extension of user-level privacy? Next, for any node- DP graph algorithm, we will have to bound its degree. Thus, we have to propose a new projection scheme (possibly truncation scheme) to reduce an arbitrary graph to a Θ bounded graph that works in the streaming model. We aim to achieve these two goals by the first milestone.

3.2 Milestone 2

Graph algorithms can be broadly classified into two types-

1. **Numeric results** These consist of numeric responses to q query like degree distribution of a graph, the number of subgraphs present, connectivity etc

2. **Graph results** These consist of synthetic or subgraph responses like MST, cut sparsifiers etc.

In the second milestone, we would like to look at the numeric response problems, study their existing non-private streaming algorithms [6, 9, 7] to identify which subclass of query it fits (counting/window/event driven) and then propose a suitable DP algorithm satisfying our DP definition from Milestone 1.

3.3 Milestone 3

Finally we would like to study the case of returning non-numeric results from graph streaming algorithms. The challenges include firstly understanding the semantics of a DP definition in the context of a non-numeric result (what is a suitable DP MST as we are releasing some of the edges anyway- is the privacy concern here the weights of the edges rather than there presence? [11][12]) and then an analysis of the privacy guarantees of the possibility (or impossibility) results.

References

[1] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[2] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1375–1388, 2017.

[3] Zhou S Chen S. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, page 653–664, 2013.

[4] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138, 2016.

[5] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *In Proceedings of ICS*, 2010.

[6] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: The value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.

[7] Mohammad Ghodsi Hossein Jowhari. New streaming algorithms for counting triangles in graphs. In *Proceedings of the International Computing and Combinatorics Conference*, pages 710–716, 2005.

[8] Raskhodnikova S Smith A Kasiviswanathan SP, Nissim K. Analyzing graphs with node- differential privacy. In *Proceedings of Theory of Cryptography Conference*, page 457–476, 2013.

[9] Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014.

[10] Darakhshan Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 37–48, 2011.

[11] Rafael Pinot. Minimum spanning tree release under differential privacy constraints. *CoRR*, 2018.

[12] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 425–438, 2017.

[13] Sofya Raskhodnikova and Adam Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 495–504, 2016.

[14] Sofya Raskhodnikova and Adam D. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *CoRR*, 2015.

[15] Procopiuc CM Srivastava D Xiao X Zhang J, Cormode G. Private release of graph statistics using ladder functions. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 731–745, 2015.