On Node-differentially Private Algorithms for Graph Statistics

Om Dipakbhai Thakkar

August 26, 2015

Abstract

In this report, we start by surveying three papers on node differential privacy. First, we look at how the three papers revolve around the same idea. At the same time, we also discuss the differences in their approach and implementation. Then, we discuss some recent work by the author[1], which is related to privately releasing the count of graph structures in a graph. We discuss the best possible Lipschitz extension for counting the number of specific subgraphs in a graph. We see an upper bound on its ratio to the extension formulated in one of the papers.

1 Introduction

Social networking websites have proliferated rapidly over the last few years. These websites provide a rich source of information for data analysis. However, the data contains sensitive information. For example, according to Blocki et al. [1], the data might be used to infer sensitive details about an individual, like sexual orientation. They add that even in an 'anonymized' unlabeled graph, it is possible to identify people based on graph structures. Moreover, a person's Facebook friends or the set of people he emails can reveal a tremendous amount of information about him [4]. Thus, releasing the data directly might violate the privacy of some individuals. A social network can be modeled as a graph, in which its users form the vertices and the relationships between them form the edges. Hence, it is important to study how to release different graph statistics while providing rigorous guarantees of privacy.

We study *node-differential privacy*. A differentially private mechanism guarantees that changes to one person's data will not significantly affect the mechanism's output distribution [3]. As a result, any two neighboring networks will induce similar distributions over the statistics released. For social networks, there are two notions of neighboring or adjacent networks: (1) *edge adjacency* stipulating that adjacent graphs differ in just one edge; and (2) *vertex adjacency* stipulating that adjacent networks differ on just one vertex. Thus, edge-differential privacy ensures that an algorithm's output does not reveal the inclusion or removal of a particular edge in the graph, while node-differential privacy hides the inclusion or removal or a node together with all its adjacent edges.

Most of the existing work on differential privacy for graphs focuses on edge privacy [5, 6, 7, 9, 10]. However, each individual in a social network is represented by a node, and not an edge. As the influence of a node in such graphs is much higher than that of an edge, privacy guarantees based

[1] In collaboration with Ramesh Krishnan Pallavoor Suresh.

on nodes are more desirable than those based on edges. Node privacy guarantees that there is no statistical difference in the output of the algorithm by the inclusion or exclusion of an individual. All the three papers that we survey focus mainly on formulating mechanisms that satisfy node privacy.

Organization of this report: Section 2 contains some basic definitions that are required for understanding the concepts related to node privacy on graphs. Section 3 provides a view on the similarities and the differences in the approaches of [1, 2, 4]. Section 4 describes the work by the author on the efficiency of one of the techniques proposed by Kasiviswanathan et al. [4] on monotone functions. Section 5 describes the attempts by the author for analyzing the efficiency of using the same technique on the transformation of a non-monotone function. As the terminologies used in [1, 2, 4] vary, we explain the correspondence amongst them in Section 6.

2 Preliminaries

Definition 2.1 (Node distance [4])**.** *The node distance $d_{node}(G, G')$ between two vertex-labeled graphs G and G' is the minimum number of operations required on G' to obtain G. An operation consists of either adding a new node with an arbitrary set of edges to the existing nodes in the graph, or removing an existing node entirely. Graphs G and G' are called neighbors if $d_{node}(G, G') = 1$.*

Definition 2.2 ((ϵ, δ)-node differential privacy [4])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-node differentially private, if for all events S in the output space of $\mathcal{A}$, and for all neighboring graphs G and G', we have*

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

Suppose we want to output a function on some private data. One popular way to output a differentially private function is to add Laplace noise to the real-valued outputs of the actual function. The amount of noise added depends on how much the function varies on small changes in the input. These changes can be quantified by the various notions of "sensitivity" of the function.

Let $\mathcal{G}_n$ denote the set of all graphs with n nodes and f be a function such that $f : \mathcal{G}_n \to \mathbb{R}^p$, for some natural number p . Then, the local and global sensitivities, in terms of node privacy, are defined as follows:

Definition 2.3 (Local Sensitivity)**.** *For a graph $G \in \mathcal{G}$, the local sensitivity of f at G is defined as*

$$LS_f(G) = \max_{G'} \|f(G) - f(G')\|_1$$

where the maximum is taken over all node neighbors G' of G.

Definition 2.4 (Global Sensitivity)**.** *The global sensitivity of f is defined as*

$$GS_f = \max_{G,G'} \|f(G) - f(G')\|_1$$

where the maximum is taken over all pairs of node neighbors G and G'.

For example, the number of edges in a graph G has local sensitivity $|V(G)|$, where $V(G)$ represents the vertex-set of G, because we can get a neighboring graph by adding a node to G

which is connected to all its existing vertices. The global sensitivity of the function is $(n-1)$ when we have a bound n on the number of nodes. In contrast, the number of nodes in a graph G has local sensitivity 1, as all neighboring graphs have a difference of exactly one node from G. Similarly, the global sensitivity of the function is also 1, even when we consider graphs of all sizes (not just a fixed size n).

Definition 2.5 (Laplace Mechanism)**.** *The Laplace mechanism is defined as $\mathcal{A}(G) = f(G) + Lap(GS_f/\epsilon)^p$. It adds i.i.d. noise $Lap(GS_f/\epsilon)$ to each entry of f.*

The Laplace mechanism is ϵ-node private. Thus, we can release the number of nodes $|V|$ in a graph with noise of expected magnitude $1/\epsilon$ while satisfying node differential privacy. Given a public bound n on the number of nodes, we can release the number of edges $|E|$ with additive noise of expected magnitude $(n-1)/\epsilon$ (the global sensitivity for releasing edge count is $n-1$).

Let us take an example to understand local sensitivity better. To count the number of triangles in a graph G, $LS_f(G) = |E|$, where E is the set of edges in G. This is because we can add a vertex to G, which is connected to every vertex originally present in G, and it would add $|E|$ triangles to G. This is very high, and hence it motivates us to consider the down local sensitivity of G, defined as follows:

Definition 2.6 (Down local sensitivity [2])**.** *The down local sensitivity [2] of f at G is defined as*

$$DLS_f(G) = \max_{v \in V(G)} \|f(G) - f(G - \{v\})\|_1$$

where $V(G)$ is the set of vertices in G, and $G - \{v\}$ corresponds to the graph obtained by removing the vertex v, along with all of its edges, from G .

Recall that a subgraph H of G is induced if for any two vertices u, v in H, u and v are adjacent in H if and only if they are adjacent in G.

Definition 2.7 (Down sensitivity [2])**.** *The down sensitivity [3] of f at G is defined as*

$$DS_f(G) = \max_{G'} DLS_f(G')$$

where G' ranges over all induced subgraphs of G.

Although the down sensitivity and the down local sensitivity of a function at most inputs is generally less than the function's global sensitivity, adding noise proportional to these is not differentially private. For example, if G is an empty graph on n vertices, $DLS_{f_{edge}}(G) = DS_{f_{edge}}(G) = 0$, and hence adding noise proportional to either $DLS_{f_{edge}}(G)$ or $DS_{f_{edge}}(G)$ can reveal information about G. Adding noise proportional to global sensitivity satisfies differential privacy, but adding a large amount of noise reduces the accuracy of the mechanism. To overcome this situation, both Kasiviswanathan et al. [4], and Blocki et al. [1], discuss the notion of bounded degree privacy, which is used to design more accurate mechanisms.

A graph is D-bounded if the maximum degree of its vertices is at most D, and $\mathcal{G}_{n,D}$ is the class of graphs on n vertices with degree bound D. A way of achieving privacy is to make use of Lipschitz extensions of functions from degree-bounded graphs to $\mathcal{G}$.

[2]Down Local Sensitivity is referred to as Local Empirical Sensitivity by Chen and Zhou [2].
[3]Down Sensitivity is referred to as Global Empirical Sensitivity by Chen and Zhou [2].

Definition 2.8 (Lipschitz constant [8]). *Let $f : X \to Y$ be a function from a domain X to a range Y with associated distance measures d_X and d_Y respectively. f has a Lipschitz constant c (or equivalently, is c-Lipschitz) if $d_Y(f(x), f(x')) \leq c \cdot d_X(x, x')$, $\forall x, x' \in X$. We define $Lip(f)$ to be the smallest possible such value, i.e.,*

$$Lip(f) = \inf\{c \geq 0 : \forall x, x' \in X \ , \ d_Y(f(x), f(x')) \leq c \cdot d_X(x, x')\}.$$

For instance, if we have a function $h : \mathcal{G} \to \mathbb{R}$, then

$$GS_h = \max_{\substack{G,G' \in \mathcal{G} \text{ s.t.} \\ d_{node}(G,G')=1}} |h(G) - h(G')| = \max_{G,G' \in \mathcal{G}} \frac{|h(G) - h(G')|}{d_{node}(G, G')} = \max_{G,G' \in \mathcal{G}} Lip(h') \ ,$$

where $h' : \{G, G'\} \to \mathbb{R}$, and $h'(X) = h(X), \forall X \in \{G, G'\}$. Similarly, for any graph $G \in \mathcal{G}$,

$$DS_h(G) = \max_{G' \sqsubseteq G} DLS_h(G') = \max_{G_1,G_2 \sqsubseteq G} \frac{|h(G_1) - h(G_2)|}{d_{node}(G_1, G_2)} = \max_{G_1,G_2 \sqsubseteq G} Lip(\hat{h}) \ ,$$

where the notation $A \sqsubseteq B$ denotes that A is an induced subgraph of B, and $\hat{h}$ is the restriction of h to pairs of induced subgraphs (G_1, G_2) of G, i.e., $\hat{h} : \{G_1, G_2\} \to \mathbb{R}$, and $\hat{h}(X) = h(X), \forall X \in \{G, G'\}$.

Definition 2.9 (Lipschitz extension [8]). *Let $X' \subset X$. Fix a constant $c > 0$. Given a c-Lipschitz function $f' : X' \to Y$, we call $\hat{f} : X \to Y$ a Lipschitz extension of f' from X' to X if*

1. *$\hat{f}$ is an extension of f', that is, $f^*(x) = f'(x)$ on all $x \in X'$ and*
2. *$\hat{f}$ is c-Lipschitz*

When $Y = \mathbb{R}$ (with the usual metric), a Lipschitz extension always exists [8]. The classical construction, given a c-Lipschitz function $f' : X' \to \mathbb{R}$, defines $f^* : X \to \mathbb{R}$ as

$$f^*(y) = \inf_{x \in X'} (f'(x) + c \cdot d_{X'}(x, y)) \ . \tag{1}$$

The function f^* is also c-Lipschitz, but need not necessarily be easy to compute, even if f' admits efficient algorithms.

When $Y = \mathbb{R}^p, p > 1$, equipped with ℓ_1 or ℓ_2 distance, then 1-Lipschitz extensions need not always exist, even when $X = \mathcal{G}_{n,D}$ [8].

3 Key ideas in [1] , [2] and [4]

In this section, we summarize the results of [1] , [2] and [4]. We mainly focus on the approaches that are concerned with the differentially private versions of releasing the number of copies of specific graph structures in a graph. Both Blocki et al. [1] and Kasiviswanathan et al. [4] consider this problem directly in the context of graphs whereas Chen and Zhou [2] consider it as a special case of a class of database queries.

All the three papers follow a similar pattern. Recall that the Laplace mechanism is one of the most basic ways to release a differentially private approximation to the value of a function on a private input. As the noise added by the mechanism depends on the global sensitivity of the function to be computed, the authors first discuss the problem of privately releasing outputs of functions with very large global sensitivity through the Laplace mechanism. In such cases, the noise to be added to the true value of the function is too high, which adversely affects the accuracy of the output. For example, let $f_{edge}(G)$ be the function which counts the number of edges in the input graph G. If G has n vertices, the local sensitivity of $f_{edge}(G)$ is n, and the local sensitivity of $f_{edge}(G)$ increases as one increases the size of G. Thus, there is no bound to the global sensitivity of f_{edge}. Moreover, even if we were to add noise proportional to the local sensitivity of $f_{edge}(G)$, the noise would be comparable to the value of the function in sparse graphs and hence, would be too high.

To overcome this situation, the authors start by concentrating on some preferred subset S of the universe of all graphs, on which they can bound the global sensitivity of the function. For example, suppose S is the set of D-bounded graphs, that is, S is restricted to graphs whose vertices have a maximum degree of at most D. Thus, if we restrict the inputs of f_{edge} to S, then the global sensitivity of f_{edge} is equal to D.

Lipschitz extensions All the three papers make use of Lipschitz extensions from S to the complete set. They define Lipschitz extensions to which they add Laplace noise proportional to the global sensitivity of the function on S. However, computing the value of these extensions for graphs outside S is inefficient as it involves searching for a 'good' value of the function within S. To get around this problem, both Chen and Zhou [2], and Kasiviswanathan et al. [4] provide efficient Lipschitz extensions using linear programs for functions such as subgraph counting.

They define a relaxed mapping for which a linear program needs to be solved to obtain the output. It is interesting to see that the linear program defined by Kasiviswanathan et al. [4] corresponds to the concept of maximum flow in a network whereas the one defined by Chen and Zhou [2] is its dual. Although the solutions to these programs can be computed efficiently, the output of the function that gets released is an approximation to the value that we get by the generic method. Thus, the output is much better and more accurate than what one would get by the straightforward Laplace mechanism, along with being efficiently computable.

Chen and Zhou [2] also show how to privately select S for the extension so that the overall error is related to the down sensitivity of the function on S.

Projection functions Due to the inefficiency of general Lipschitz extensions, Blocki et al. [1], and Kasiviswanathan et al. [4] introduce projection functions. Let us choose a subset C which is a superset of the preferred subset S. A projection function μ maps each element in the universe to some element in C. Elements in S are mapped to themselves whereas each element not in S is mapped to some element in C. Both [1] and [4] provide projections for which S is the set of D-bounded graphs. In the projection introduced by Blocki et al. [1], a linear program needs to be solved to compute the projection of the input graph. Here, C is the set of $2D$-bounded graphs. Kasiviswanathan et al. [4] introduce a projection function called naive truncation. Here, C is equal

to S, and the projection has a threshold which depends on S. It deletes all nodes of degree greater than the threshold in the input graph. Thus, their truncation operator gives a projection of the input to S in linear time.

Consider the task of evaluating a function f on an input G. Now, to get an approximation to $f(G)$, both Blocki et al. [1], and Kasiviswanathan et al. [4] compute its projection $\mu(G)$, then calculate $f(\mu(G))$ and finally release $f(\mu(G))$ with noise added to it. Noise is added because μ may be sensitive at G, i.e., $\mu(G)$ may be far from $\mu(G')$, even when G and G' are close. Hence, along with μ, they define a "smooth" upper bound on the local sensitivity of f, which depends on both the global sensitivity of f when restricted to S, and the distance between G and $\mu(G)$. So, Laplace noise proportional to the "smooth" sensitivity of G is added to $f(\mu(G))$ before releasing it. Both papers bound the local sensitivity of the projection in terms of the structure of the input graph, though the bounds are different. Blocki et al. [1] bound it in terms of the graph's distance to S, whereas Kasiviswanathan et al. [4] bound it in terms of the number of vertices having prespecified degrees in the graph.

More general queries Additionally, Blocki et al. [1] also focus on various applications of different graph structures, with a specific focus on social networks. For example, finding the number of edges in a graph corresponds to finding the number of pairs of people who are friends in social network. Similarly, finding the number of triangles in a graph corresponds to finding the number of triplets who are friends with each other in a social network. They use different projection-based techniques to construct functions which maintain privacy and on specific datasets, are an accurate representation of the functions from which they are projected.

Chen and Zhou [2] define all sensitivity definitions with respect to databases. In general, graph structures correspond to relations in a database. Their main result corresponds to adding noise proportional to the down sensitivity of the database.

4 Approximate optimality of general extensions

Here, we first define optimal Lipschitz extensions. Kasiviswanathan et al. [4] construct a linear program based extension whose output is close to the value of the optimal Lipschitz extension, but lesser than it. We prove that the output is within a constant factor of the value of the optimal Lipschitz extension of the required function. We consider the general case of counting the copies of a smaller graph structure in a graph. Then, we consider the special case of counting the number of edges in a graph, which has been done by relating it to the maximum flow in a network.

4.1 Optimal Lipschitz extensions

Given a c-Lipschitz function $f' : X' \to \mathbb{R}$, we defined $f^* : X \to \mathbb{R}$ in definition 2.9 as

$$f^*(y) = \inf_{x \in X'} (f'(x) + c \cdot d_{X'}(x, y)).$$

Lemma 4.1. *f^* is the largest Lipschitz extension of f', i.e., if $\hat{f}$ is a Lipschitz extension of f' from $X \to \mathbb{R}$, then*

$$\hat{f}(y) \leq f^*(y), \forall y \in X \ .$$

Proof. Let us assume that there exists a Lipschitz extension of f', called $\hat{f}$, from $X \to \mathbb{R}$ and some element $z \in X$ such that $\hat{f}(z) > f^*(z)$. Thus, $\hat{f}(z) > \inf_{x \in X'}(f'(x) + c \cdot d_{X'}(x, z))$ (from (1)). However, this implies that $d_Y(\hat{f}(x), \hat{f}(z)) = |\hat{f}(x) - \hat{f}(z)| > c \cdot d_X(x, z)$, which contradicts the fact that $\hat{f}$ is c-Lipschitz. Hence, f^* is the largest possible Lipschitz extension of f'. □

Functions for which the down sensitivity at each input is equal to its down local sensitivity at that input are called submodular functions. For functions $f' : \mathcal{G} \to \mathbb{R}$ which are submodular and monotone, the value of f' at any input is never less than the value of $f^* : \mathcal{G}' \to \mathbb{R}$ at that input. We can understand it as follows: Let us assume that for some $G \in \mathcal{G}$,

$$\begin{aligned} f'(G) &< f^*(G) \\ &= \inf_{g \in \mathcal{G}'}(f'(g) + c \cdot d_{node}(G, g)) \\ &\leq (f'(g) + c \cdot d_{node}(G, g)), \forall g \in \mathcal{G}' \\ \Rightarrow f'(G) - f'(g) &< c \cdot d_{node}(G, g)), \forall g \in \mathcal{G}' \\ \Rightarrow DS_{f'}(G) &< c \end{aligned}$$

However, as we know that f' is c-Lipschitz for inputs in $\mathcal{G}'$, and as f' is submodular and monotone,

$$\Rightarrow DS_{f'}(G) \geq DLS_{f'}(G) \geq c.$$

Therefore, we arrive at a contradiction.

Thus, as the value of f' at any input is never less than the value of f^* at that input, for such functions f^* is also called an optimal Lipschitz extension of f' from X' to X.

4.2 Analyzing the LP for general substructures

First, we provide a bound for the ratio of the optimal Lipschitz extension of the number of copies of graph H in a graph, to the optimal value of the linear program defined by Kasiviswanathan et al. [4].

Let the function $f_H(G)$ give the number of copies of graph H in G. Let S_H be the set of copies of H in G. So, for any graph $G \in \mathcal{G}$, $f_H(G) = |S_H|$. Now, let Δ be the maximum number of copies each vertex of the graph can participate in, and let $\mathcal{G}_{H,\Delta}$ be the set of all such graphs. For any graph $G \in \mathcal{G}_{H,\Delta}$, the linear program for counting the number of copies of H consists of h variables and $v+h$ constraints, where h is the total number of copies of H in G. It can be defined as:

$$\begin{aligned} \text{Maximize} \quad & \sum_{h \in S_H} x_h \\ \text{subject to} \quad & \sum_{h \text{ contains } v} x_h \leq \Delta, \quad \forall v \in V \\ & x_h \leq 1, \quad \forall h \in S_H \end{aligned}$$

Let $\hat{f}_{H,\Delta}(G)$ represent the value of the optimal solution of the linear program above.

Lemma 4.2. *$\hat{f}_{H,\Delta}$ is a Lipschitz extension of counting the number of copies of H in a graph from $\mathcal{G}_\Delta$ to $\mathcal{G}$.*

Proof. To prove it, we need to show that

1. if $G \in \mathcal{G}_\Delta$, then $\hat{f}_{H,\Delta}(G) = f_H(G)$, and
2. the global node sensitivity of $\hat{f}_{H,\Delta}$ is at most Δ.

First, we will prove that if $G \in \mathcal{G}_{H,\Delta}$, then $\hat{f}_{H,\Delta}(G) = f_H(G)$.

Consider the following solution to the linear program for computing $\hat{f}_{H,\Delta}(G)$: assign $x_h = 1, \forall h \in S_H$. In this solution, all the constraints corresponding to the vertices are satisfied as $G \in \mathcal{G}_{H,\Delta}$ and thus, every vertex of G participates in at most Δ copies of H. All the constraints corresponding to the copies of H in G are tight, as we have set the values of all the variables equal to 1. Clearly, this solution is valid and has value $\hat{f}_{H,\Delta}(G) = \sum_{h \in S_H} x_h = f_H(G)$.

To see that this value is maximum, observe that due to the constraints corresponding to the copies of H in G, every variable can take a maximum value of 1 in any valid solution of the LP. In our solution, each variable gets assigned its maximum value possible. As the objective function is a maximization function involving the sum of all the variables in the LP, the value of the objective function in our solution is the maximum possible value it can achieve.

We now prove that the global node sensitivity of $\hat{f}_{H,\Delta}$ is at most Δ. Consider an arbitrary pair of neighbors G,$G' \in \mathcal{G}$, s.t. $G \sqsubseteq G'$. Let v be the vertex added to G to obtain G'. Let k be the number of copies of H created when v, along with its edges, is added to G. Adding v to G adds k variables and $(k+1)$ constraints to the linear program for $\hat{f}_{H,\Delta}(G)$. Since the number of copies of H only increases from G to G', the optimal solution for the LP for G is a valid solution for the LP for G'. So the value of $\hat{f}_{H,\Delta}(G')$ cannot decrease. To see that it can increase by at most Δ, observe that the new variables corresponding to vertex v can contribute at most Δ to $\hat{f}_{H,\Delta}(G')$, as in the constraint corresponding to vertex v, $\sum_{h \text{ contains } v} x_h \leq \Delta$. All the other variables together can contribute at most $\hat{f}_{H,\Delta}(G)$ to $\hat{f}_{H,\Delta}(G')$. It is because if they contribute more than this, then setting all the variables corresponding to node v to 0 will give a value of the solution more than $\hat{f}_{H,\Delta}(G)$ for G. This contradicts the fact that $\hat{f}_{H,\Delta}(G)$ is the maximum value possible for the LP for G. Thus, the value of the optimization function in the LP can increase by at most Δ. Therefore, the global node sensitivity of $\hat{f}_{H,\Delta}$ is at most Δ. □

Let $f^*_{H,\Delta}$ be the optimal Lipschitz extension of f_H from $\mathcal{G}_{H,\Delta}$ to $\mathcal{G}$. How far is the efficient extension $\hat{f}_{H,\Delta}$ from $f^*_{H,\Delta}$? There exist inputs G on which the ratio $\frac{f^*_{H,\Delta}(G)}{\hat{f}_{H,\Delta}(G)}$ is arbitrarily close to k, where k represents the number of vertices in H.

Proposition 4.3. *For every k, there exists a Δ and graphs G and H such that for some constant $c > 0$, $\frac{f^*_{H,\Delta}(G)}{\hat{f}_{H,\Delta}(G)} > k \cdot \left(1 - \frac{c}{n}\right)$, where n is the number of vertices in G.*

Proof. Let H be a clique on k vertices, and choose a Δ such that $\binom{\ell-1}{k-1} = \Delta$ for some $\ell > 0$. Now, let K_ℓ be a clique on ℓ vertices. As each vertex in K_ℓ participates in exactly Δ copies of H, $K_\ell \in \mathcal{G}_{H,\Delta}$. Also, the number of copies of H in K_ℓ, i.e., $f_H(K_\ell)$ is $\frac{\ell\Delta}{k}$. Let G be a clique on n vertices such that $n > \ell$. Notice that $G \notin \mathcal{G}_{H,\Delta}$, as each vertex in G participates in more than Δ copies of H. Thus,

$$f^*_{H,\Delta}(G) = \inf_{g \in \mathcal{G}_{H,\Delta}} (f_H(g) + \Delta \cdot d_{node}(G, g))$$

Every induced subgraph of G is a clique. Moreover, only cliques with size at most ℓ are in $\mathcal{G}_{H,\Delta}$. The infimum is achieved at K_ℓ as $K_\ell \in \mathcal{G}_{H,\Delta}$ and removing every additional vertex from K_ℓ will remove at most Δ from $f^*_{H,\Delta}(G)$ in the first term but add exactly Δ to it in the second term. Hence,

$$f^*_{H,\Delta}(G) = f_H(K_\ell) + \Delta \cdot d_{node}(G, K_\ell) = \frac{\ell\Delta}{k} + \Delta(n - \ell). \tag{2}$$

Now, for $\hat{f}(G)$, we can write the objective of the LP as a linear combination of its constraints. Specifically, if we multiply the constraint corresponding to every vertex by $\frac{1}{k}$, and multiply the constraint corresponding to every copy of H by 0, then adding+ them all results in the objective function on the LHS. It is because every copy h of H contains exactly k vertices, and $k \cdot \frac{x_h}{k} = x_h$. If x is an optimal solution, then we can say from the linear combination that

$$\hat{f}(G) = \sum_{h \in S_H} x_h = \frac{1}{k} \sum_{v \in V} \Big(\sum_{\substack{h : h \in S_H \ \& \\ h \text{ contains } v}} x_h \Big) \leq \frac{n}{k} \cdot \Delta \tag{3}$$

Thus,

$$\begin{aligned} \frac{f^*_{H,\Delta}(G)}{\hat{f}(G)} &\geq \frac{\frac{\ell \cdot \Delta}{k} + (n - \ell) \cdot \Delta}{\frac{n \cdot \Delta}{k}} && \text{(From (2) and (3))} \\ &= \frac{nk\Delta - (k-1)\ell\Delta}{n\Delta} = k \cdot \left(1 - \frac{(k-1)\ell}{kn}\right) \\ &> k \cdot \left(1 - \frac{c}{n}\right), \text{ for } c = \ell \end{aligned}$$

□

Theorem 4.4. $\frac{f^*_{H,\Delta}(G)}{k} \leq \hat{f}_{H,\Delta}(G) \leq f^*_{H,\Delta}(G)$.

The structure of the proof is as follows: we start by considering the dual of the LP given above. We first show that the value of the optimal integral solution of the dual is equal to $f^*_{H,\Delta}$. Next, we bound the value of the integral solution of dual from above, and show that it is at most the value of the integral solution of the primal multiplied by k. This implies that the LP has an integrality gap of at most k, which implies the statement of the theorem.

Proof. The dual of the LP given above has $v + h$ variables and h constraints. Let us denote the variables by y_h for all $h \in S_H$ and y_v for all $v \in V$. The dual is:

$$\begin{aligned} \text{Minimize} \quad & \sum_{h \in S_H} y_h + \Delta \cdot \sum_{v \in V} y_v \\ \text{subject to} \quad & y_h + \sum_{i=v_1}^{v_k} y_i \geq 1, \qquad \forall h = \{v_1, v_2, ..., v_k\} \in S_H \\ & y_h \geq 0, \qquad \forall h \in S_H \\ & y_v \geq 0, \qquad \forall v \in V \end{aligned}$$

Lemma 4.5. *$f^*_{H,\Delta}(G)$ = value of the optimal integral solution of the dual.*

Proof. Let us take an instance y of the optimal integral solution of the dual. Fix a vertex $v \in V$ and some $h' \in S_H$. By observing the structure of the dual LP, we can say that:

1. $y_v, y_{h'} \in \{0, 1\}$.

   The minimum value that y_v can take is 0, due to the constraint involving only y_v in the LP. To satisfy the constraints involving both y_v and y_h such that v participates in $h \in S_H$, y_v can take positive values. However, it is a minimization LP. Hence, in an optimal solution, y_v will take the least possible value required to satisfy the constraints, which is 1. Thus, $y_v \in \{0, 1\}$. Applying similar reasoning for $y_{h'}$, we can say that $y_{h'} \in \{0, 1\}$.

2. If $y_v = 1$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h = 0$.

   All the constraints involving these variables have been satisfied by the value of y_v. As it is a minimization LP, all the variables will take the least possible value required to satisfy the constraints. Hence, if $y_v = 1$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h = 0$. .

3. If $y_v = 0$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h \leq \Delta$.

   If $\sum y_h > \Delta$, then we can have a better solution y' by setting $y'_v = 1$, $y'_h = 0$ for all copies of $h \in S_H$ that contain v, and $y' = y$ for all other variables. Setting $y'_v = 1$ contributes exactly Δ to the objective value for y', whereas $\sum y_h$ contributes more than Δ to the objective value for y. All other variables contribute the same to the objective value in both y and y'. As a result, the objective value of y' must be less than that of y. However, this contradicts the fact that y is an optimal integral solution. Thus, if $y_v = 0$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h \leq \Delta$.

4. $y_{h'} = 1 \Leftrightarrow \forall v' \in V$ s.t. v' participates in h', $y_{v'} = 0$.

   The backward direction of the double implication is straightforward: If $y_{v'} = 0$ for all vertices $v' \in V$ contained in h', then to satisfy the constraint involving both $y_{h'}$ and the variables corresponding to the vertices of h', $y_{h'}$ has to take a positive value. As $y_{h'} \in \{0, 1\}$, we can say that $y_{h'} = 1$.

   The forward direction of the double implication can be understood as follows: if $y_{h'} = 1$, then the constraint involving both $y_{h'}$ and the variables corresponding to the vertices of h' has already been satisfied. If $\exists v' \in V$ s.t. v' participates in h' and $y_{v'} = 1$, then we can get a better solution to the LP by setting $y'_h = 0$, as doing this results in a lower value of the

objective function. However, this contradicts the fact that the solution we have is an optimal integral solution. Hence, if $y_{h'} = 1$, then $\forall v' \in V$ s.t. v' participates in h', $y_{v'} = 0$.

Now, let us take a solution of the dual LP which has the following restrictions:

1. $\forall v \in V, y_v \in \{0, 1\}$
2. $\exists h \in S_H$ s.t. $y_h = 1 \Leftrightarrow \forall v \in V$ s.t. v participates in $h, y_v = 0$
3. If $\exists v \in V$, s.t. $y_v = 1$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h = 0$
4. If $\exists v \in V$, s.t. $y_v = 0$, then $\forall h \in S_H$ s.t. v participates in h, $\sum y_h \leq \Delta$

Let us call it a reduced solution of the dual LP. A reduced solution is, in particular, feasible as it does not violate any constraints. Let G' be the induced subgraph formed by all the vertices $v \in V$ s.t. $y_v = 0$ in the solution. Notice that $G' \in \mathcal{G}_\Delta$, as for all $h \in S_H$ s.t. v participates in h, $\sum y_h \leq \Delta$. So, every reduced solution corresponds to a subgraph $G' \in \mathcal{G}_\Delta$. Similarly, given a subgraph $G' \sqsubseteq G$, such that $G' \in \mathcal{G}_\Delta$, we can get a reduced solution by setting

$$y_v = \begin{cases} 0 & \forall v \in G' \\ 1 & \text{otherwise} \end{cases}$$

$$y_h = \begin{cases} 1 & \forall h \in G' \\ 0 & \text{otherwise} \end{cases}$$

Thus, every subgraph $G' \sqsubseteq G$, such that $G' \in \mathcal{G}_\Delta$, corresponds to a reduced solution. Hence, there is a one-to-one correspondence between the reduced solutions of the dual LP and the subgraphs of G which are in $\mathcal{G}_\Delta$. So, if y is a reduced solution and G' is the corresponding subgraph, then

$$\text{value}(y) = f_H(G') + \Delta \cdot d_{node}(G, G').$$

Therefore, by the definition of $f^*_{H,\Delta}$, the value of the optimal integral solution of the dual linear program is exactly equal to $f^*_{H,\Delta}(G)$. □

From Lemma 4.5, we can say that $\hat{f}_{H,\Delta}(G) \leq f^*_{H,\Delta}(G)$. Therefore, to prove the theorem, it suffices to show that the integrality gap of the LP is almost k. So, let us consider an optimal integral solution of the primal. Since it is an integral solution, all x_h's have values in {0,1} because $0 \leq x_h \leq 1$. We will now construct a feasible solution of the dual LP from this solution of the primal and then try to bound the value of the optimal integral solution of the dual in terms of the value of the optimal integral solution of the primal. Our algorithm to construct a feasible solution of the dual LP from an optimal integral solution of the primal is:

In other words, assign:

$$\forall v \in V, y_v = \begin{cases} 1, & \text{if the constraint corresponding to } v \text{ is tight in } x \\ 0, & \text{otherwise} \end{cases}$$

$$\forall h \in S_H, y_h = \begin{cases} 1, \text{if } x_h = 1 \text{ and no constraint corresponding to the vertices of } h \text{ is tight in } x \\ 0, \text{ otherwise} \end{cases}$$

Input: An optimal integral solution x of the primal LP
Output: A feasible integral solution y of the dual LP

```
∀h ∈ S_H, assign y_h = 0 ;
for every v ∈ V, do
    if the constraint corresponding to v is tight in x, then
        assign y_v = 1;
    else
        assign y_v = 0;
    end
end
for every h = {v_1, v_2, ..., v_k} ∈ S_H, where x_h = 1, do
    if Σ_{i=1}^{k} y_{v_i} = 0, then
        assign y_h = 1;
    end
end
```

Next, let us see why y is a feasible solution of the dual. For every copy $h \in H$ where $x_h = 1$, if none of the constraints corresponding to its vertices are tight in x, $y_h = 1$. Otherwise, for at least one of its vertices v, $y_v = 1$. Thus, the constraint in the dual corresponding to h and the vertices of h is satisfied in y. For every copy $h \in H$ where $x_h = 0$, the only reason it is 0 because it has at least one vertex $v \in V$ for which the constraint in x is tight. Consequently, we have already set $y_v = 1$, thus satisfying the constraint in the dual corresponding to h and its vertices. Hence, y is feasible as it satisfies all the constraints in the dual LP.

Now, let us see how large the value of y can be, compared to the value of x. The value of x is $\sum_{h \in S_H} x_h$. The value of y is $\sum_{h \in S_H} y_h + \Delta \cdot \sum_{v \in V} y_v$. We will bound the value of the dual by "charging" part of the dual objective to each copy $h \in S_H$. So, we want to find a "charge" $c(h)$ such that

$$\text{Value of } y = \sum_{h \in S_H} c(h) \leq k \sum_{h \in S_H} x_h$$

Specifically, for all $h \in S_H$, set

$$c(h) = \begin{cases} 0 & \text{if } x_h = 0 \\ 1 & \text{if } x_h = y_h = 1 \\ \#\{v \text{ contained in } h : y_v = 1\} & \text{if } x_h = 1 \text{ and } y_h = 0 \end{cases}$$

Lemma 4.6. *Value of* $y = \sum_{h \in S_H} c(h)$.

Proof.

$$\text{Value of } y = \sum_{h \in S_H} y_h + \Delta \cdot \sum_{v \in V} y_v \tag{4}$$

Now, for every $h \in S_H$ such that $y_h = 1$, we have $c(h) = 1$. Thus,

$$\sum_{h \in S_H} y_h = \sum_{h \in S_H : y_h = 1} c(h) \tag{5}$$

Also, by our construction,

$$\sum_{h \in S_H : y_h = 0} c(h) = \sum_{h \in S_H : y_h = 0} \#\{v \text{ contained in } h : y_v = 1\}$$

$$\Rightarrow \sum_{h \in S_H : y_h = 0} c(h) = \Delta \cdot \sum_{v \in V} y_v. \tag{6}$$

We can understand equation (6) as follows: For every $v \in V$ such that $y_v = 1$, the constraint corresponding to v is tight in x. Thus, y_v adds a "charge" of 1 to every copy h of H which contains v and $x_h = 1$. Therefore, every such $v \in V$ adds a combined "charge" of exactly Δ to the copies of H.

Substituting the value of equations (5) and (6) in equation (4), we get the statement of the lemma, as desired.

□

Lemma 4.7. $\sum_{h \in S_H} c(h) \leq k \sum_{h \in S_H} x_h$.

Proof. To prove the lemma, we will consider all the cases it covers. For every $h \in S_H$:

1. if $x_h = 0$, then $c(h) = 0 = x_h$
2. if $x_h = 1$ and $y_h = 1$, then $c(h) = 1 = x_h$
3. if $x_h = 1$ and $y_h = 0$, then $c(h) = (\#v \in V$ contained in h s.t. $y_v = 1)$. As there are k vertices in every copy of H, $\Rightarrow c(h) \leq k = k \cdot x_h$

From the 3 cases above, we can say that $\sum_{h \in S_H} c(h) \leq k \sum_{h \in S_H} x_h$, as desired. □

Let $\tilde{f}(G)$ denote the value of the optimal integral solution of the primal. The value of the optimal integral solution of the dual, $f^*_{H,\Delta}(G)$, is at most the value of y, which is the solution constructed by us. Thus,

$$\begin{aligned}
f^*_{H,\Delta}(G) \leq\ & \text{Value of } y \\
&= \sum_{h \in S_H} y_h + \Delta \cdot \sum_{v \in V} y_v \\
&= \sum_{h \in S_H} c(h) && \text{(From Lemma 4.6)} \\
&\leq k \cdot \sum_{h \in S_H} x_h && \text{(From Lemma 4.7)} \\
&= k \cdot \tilde{f}(G) \\
\Rightarrow \frac{f^*_{H,\Delta}(G)}{k} &\leq \tilde{f}(G)
\end{aligned}$$

Also, we know that the value of the optimal integral solution of the primal is at most $\hat{f}_{H,\Delta}(G)$.

$$\Rightarrow \frac{f^*_{H,\Delta}(G)}{k} \leq \hat{f}_{H,\Delta}(G)$$

□

4.3 Counting the number of edges in a graph

It is worth mentioning here that when H is an edge, that is, f_H counts the number of edges in its input, we can make use of flow graphs to efficiently get the solution to our linear program. We can define a flow graph as follows:

Definition 4.8 (Flow graph [4])**.** *Given an (undirected) graph $G \in \mathcal{G}_n$ such that $G = (V, E)$; let $V_l = \{v_l | v \in V\}$ and $V_r = \{v_r | v \in V\}$ be two copies of V, called the left and right copies, respectively. Let D be a natural number less than n. The flow graph of G with parameter D, a source s and a sink t is a directed graph on nodes $V_l \cup V_r \cup \{s, t\}$ with the following capacitated edges: edges of capacity D from the source s to all nodes in V_l and from all nodes in V_r to the sink t, and unit-capacity edges (u_l, v_r) for all edges $\{u, v\} \in E$. Let $v_{fl}(G)$ denote half the value of the maximum flow in the flow graph of G.*

Kasiviswanathan et al. [4] showed that v_{fl} is a Lipschitz extension for counting the number of edges in a graph from G_D to $\mathcal{G}$. Thus, the value of the solution of the primal LP defined earlier is the same as that of the flow graph, i.e., the value of $\hat{f}_{H,\Delta}$ is equal to v_{fl}. The formulation into a flow graph allows employing faster algorithms.

5 Connected components in a graph

Now, we look at the problem of counting the number of connected components in a graph. As the addition/removal of a vertex from a graph can either increase or decrease the number of connected components in it, we see that it is not a monotone function. For example, in a k-star, removing the central vertex (which is connected to the k-vertices) increases the number of connected components in it. However, the addition of an isolated vertex also has the same effect on it. Similarly, in a graph with k isolated vertices, removing a vertex decreases the number of connected components in it. However, adding a vertex which is connected to all the vertices in the graph also has the same effect. Hence, to make our function monotone, we change it to be defined as

$$f(G) = n - c$$

where n is the number of vertices in G and c is the number of connected components in G. Note that, f is a monotone function now, as the addition of a vertex to G can only result in either f increasing or staying the same. Similarly, the removal of a vertex from G can only result in either f decreasing or staying the same. Here, we require a monotone function because in such situations, it is easier to get inferences from monotone functions.

The key observation here is that $f(G)$ is the number of edges in the spanning forest of G. For example, if G is a k-clique, $f(G) = k - 1$, which is the number of edges of its spanning tree. More generally, if G is a graph on n vertices and has k connected components, then in the spanning forest of G, every connected component of G forms a tree of length one less than the number of vertices in that component. Thus, the number of edges in the spanning forest of G is equal to $n - k$, which is equal to $f(G)$.

The Lipchitz extension for f from G_D to $\mathcal{G}$ can be given by the LP for finding a spanning forest in D-bounded graphs, which can be defined as:

$$\begin{aligned} \text{Maximize} \quad & \hat{f} = \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e:\ e \text{ is incident on v}} x_e \le D, \qquad \forall v \in V \\ & \sum_{e:\ e=\{u,v\} \text{ and } \{u,v\} \subseteq S} x_e \le |S| - 1, \quad \forall S \subseteq V \text{ s.t. } |S| > 1 \end{aligned}$$

Lemma 5.1. *$\hat{f}$ is a Lipschitz extension of f from G_D to $\mathcal{G}$.*

Proof. To prove the lemma, first we will prove that if a graph G is D-bounded, then $\hat{f}(G) = f(G)$.

Consider a spanning forest, S, of G, with k connected components. For the LP defined above, assign all the variables $x_e = 1, \forall e \in E(S)$, where $E(S)$ represents the edge-set of S. Clearly, this solution is valid and has value equal to $f(G)$. It is because, as G is D-bounded, its spanning forest is also D-bounded. So, the first constraint is satisfied for all vertices in G. Also, every connected component, C, in G, has $|V(C)| - 1$ edges in its spanning tree, where $V(C)$ represents the vertex-set of C. Thus,

$$\hat{f}(G) = \text{No. of edges in spanning forest of G } = \sum_{i=1}^{k} (|V(C_i)| - 1) = n - k = f(G)$$

We will now prove that the global node sensitivity of $\hat{f}(G) = D$. When we add a node v with some adjacent edges to G, it corresponds to adding $|E_v(G)|$ variables to the LP defined on G above, where $E_v(G)$ is the set of edges connected to node v . Since we only add edges, the previous maximum solution to the LP is a valid solution in the newly formed graph. So the value of $\hat{f}$ cannot decrease. To see that it can increase by at most D, observe that in a solution of the LP, every node can only be connected to at most D edges. Thus, if the value of $\hat{f}$ increases by more than D, removing v would give a value greater than the previous optimal value, thus arriving at a contradiction. By symmetry, removing a node from G cannot increase the value of $\hat{f}$, and can decrease it by at most D. Therefore, rewiring can change the value of $\hat{f}$ by at most D.

Hence, we prove that $\hat{f}$ is a Lipschitz extension of f from G_D to $\mathcal{G}$. □

Lemma 5.2. *$\forall G \in \mathcal{G}$, $\hat{f}(G) \le f(G)$.*

Proof. If G is D-bounded, we have proved in Lemma 4.5 that $\hat{f}(G) = f(G)$.

If G is not D-bounded, then, at least one spanning forest of G is not D-bounded. As the solution defined by the LP has to be in the form of a forest which is D-bounded, the number of

edges in the solution cannot be greater than the number of edges in the spanning forest of G. Thus, $\hat{f}(G) \leq f(G)$.

Hence, we prove that $\forall G \in \mathcal{G}$, $\hat{f}(G) \leq f(G)$.

□

The optimal Lipschitz extension, f^*, can be defined as

$$f^*(G) = \inf_{g \in G_D} (f(g) + D \cdot d_{node}(G, g))$$

The dual for the LP defined above, can be given by:

$$\begin{aligned} \text{Minimize} \quad & D \cdot \sum_{v \in V} y_v + \sum_{s \subseteq V \text{ s.t. } |s|>1} (|s| - 1) \cdot y_s \\ \text{subject to} \quad & \sum_{s: u,v \in S} y_s + y_u + y_v \geq 1, \qquad \forall e = \{u, v\} \in E \end{aligned}$$

There have been attempts to bound the ratio of f^* to $\hat{f}$, but so far they have not yielded any conclusive results. Also, there are many interesting open questions related to this problem. Some examples are:

1. Does $\hat{f}(G) = f(G)$ when the down sensitivity of $G \leq \Delta$?
2. Is there a way to generalize this problem to directed graphs?

6 Equivalences between terms from [1],[2] and [4]

Here, we present a (partial) list of terminologies which correspond to each other from [2] and [4]:

1. Linear query in [2] $\iff$ Generalized counting query where tuples, that are summed, get assigned real numbers in [4]
2. Function X in [2] $\iff$ Lipschitz extension with parameter $\hat{\Delta}$ in [4]
3. Δ in [2] $\iff$ Some approximation to the "best" setting of the Lipschitz parameter in [4]
4. $\hat{\Delta}$ in [2] $\iff$ Differentially private version of Δ in [4]

The terminology used in [1] and [4] is almost the same.

References

[1] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. *CoRR*, abs/1208.4586, 2012. URL `http://arxiv.org/abs/1208.4586`.

[2] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 653–664, 2013. URL http://doi.acm.org/10.1145/2463676.2465304.

[3] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *In Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[4] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *TCC*, pages 457–476, 2013.

[5] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 921–930, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623683. URL http://doi.acm.org/10.1145/2623330.2623683.

[6] D. Mir and R. N. Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 167–176, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1143-4. doi: 10.1145/2320765.2320818. URL http://doi.acm.org/10.1145/2320765.2320818.

[7] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *In STOC*, pages 75–84. ACM, 2007.

[8] S. Raskhodnikova and A. D. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *CoRR*, abs/1504.07912, 2015. URL http://arxiv.org/abs/1504.07912.

[9] V. Rastogi and G. Miklau. Relationship privacy: Output perturbation for queries with joins. In *In ACM Symposium on Principles of Database Systems, 2009. [13] Yossi.*

[10] Q. Xiao, R. Chen, and K.-L. Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 911–920, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623642. URL http://doi.acm.org/10.1145/2623330.2623642.