Differentially Private Computation in Graphical and Networked Domains

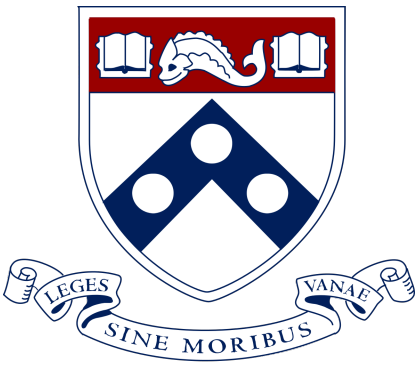
Brandon Lin
branlin@seas.upenn.edu

Advisor: Aaron Roth
aaroth@cis.upenn.edu

May 16, 2020

Abstract

With the ever-growing demand for insightful data analyses, the protection of user data becomes more paramount than ever before. *Differential privacy* is a recent mathematical formalization of maintaining privacy that has been incorporated in many large technology companies as a way of protecting their users' data. In this paper, we investigate how network analysis and graph theory inspired the creation of new techniques in differentially private algorithm design. We see that the particular structure and nature of networks makes them particularly sensitive to perturbation, and investigate techniques such as noise reduction and Lipschitz extensions in the privacy literature to mitigate this. Finally, we look at several applications of the aforementioned techniques to private combinatorial optimization and graph statistic release.



Contents

1 Introduction

In today's society, data is extremely valued – each data point an application has on a user is a bigger step in tailoring their user experience to them. In particular, individual interactions on online applications have significantly contributed to the insights one can provide to those users, such as giving personalized recommendations. However, with the rise of recent qualms regarding data privacy breaches at technology companies that amass such data, society has seen an increased scrutiny towards the rightful and secure usage of data. Even though data is extremely valuable for providing insights, simple anonymization and release is not sufficient to protect individual privacy, as demonstrated in the Netflix Prize de-anonymization in Narayanan and Shmatikov [NS06].

Differential privacy has emerged as a way to mathematically formulate the challenges of privacy. As the winner of the Gödel Prize in 2017, differential privacy has seen massive development and promise as a way to answer key problems in data analysis, such as combinatorial optimization and learning theory. Differential privacy, as introduced in [Dwo06], is a way of introducing relative privacy, since the assurance of absolute privacy for a dataset is impossible. Many techniques have been developed to design private algorithms, mainly outlined in Dwork and Roth [DR14].

In this paper, we focus on a particular area of private algorithm design: algorithms that operate over structured networks and graphs. Primarily, we will see how simply thinking about problems in graph theory helps to motivate the creation of new techniques for private algorithm design, namely projection-based methods. The graphical type of data structure is extremely practical as it allows us to model the interactions among many individuals in a community. Graphs offer a very nice abstraction over a variety of problems in, for example, combinatorial optimization, and offer a nice visualization over a large population of interactions. However, the structure of graphs will also prove difficult to create private algorithms for.

One of the major applications of privacy in networks is the release and computation of data in social network applications, such as Facebook, YouTube, Twitter, etc. Companies such as Facebook have had an increased focus on better privacy methods since the Cambridge Analytica scandal [Lap], so the development of such network analysis algorithms would be crucial for Facebook to work towards this goal. A prominent application is for companies to identify members of a particular population based on social network data (such as inclusion in a terrorist organization) without compromising the privacy of others. Proof of concept methods for this are detailed in Kearns et al. [KRWY15]. Other papers [GRU11, KNRS13, HLMJ09, XCT14] detail other private algorithms such as cut size computation and network data release, so there are a wide variety of uses for social networks to discover insights on their data while preserving privacy.

2 Background

2.1 Basic definitions

We begin by introducing the formalization of privacy; namely, its definition, analysis techniques, and mechanisms for ensuring privacy.

The term "private data release" seems very oxymoronic; how can we ensure accurate reporting of results without giving away valuable information about particular people, if the data we release is itself valuable? If we stick with an absolute measure of privacy, however, Dwork [Dwo06] illustrates an impossibility result about ensuring this. In this work, "auxiliary information" is cited as the main obstacle to absolute privacy, information one may gather about a particular database. Knowledge of such auxiliary information essentially gives a significant advantage to someone that has the dataset as opposed to someone without the dataset [Dwo06].

If we can't ensure absolute privacy, perhaps we can relax our definition of privacy and consider *relative privacy* instead [Dwo06]; that is, rather than defining a private data release as one where an individual can't be learned about, but instead one where including a certain individual within the data release does not allow an adversary to learn anything about that individual (namely, their presence within the dataset). Let $\mathcal{D}$ denote the space of all datasets. To formalize this definition,

Definition 2.1 ([Dwo06, DMNS06])**.** *An ε-differentially private algorithm $\mathcal{A} : \mathcal{D} \to \mathbb{R}^d$ satisfies the property that if $\forall D, D' \in \mathcal{D}$ differing in exactly one element, and for all $X \subseteq Ran(\mathcal{A})$, we have*

$$\Pr[\mathcal{A}(D) \in X] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in X]$$

Notice that by writing $e^{\varepsilon} \approx 1 + \varepsilon$, the definition becomes

$$\Pr[\mathcal{A}(D) \in X] \leq (1 + \varepsilon) \Pr[\mathcal{A}(D') \in X]$$

which we can more intuitively digest. This definition suggests that any additional individual added to a dataset should not increase the "learnability" of that dataset significantly (by more than an ε additive factor) [Dwo06]. The higher the value of ε, the more information we can afford to "leak" in $\mathcal{A}$'s output. When we say that D and D' differ in exactly one element, we mean this in the following way: if we view D and D' as elements in $\mathbb{R}^n$ (where each element of the vector represents an individual's data), the Hamming distance between them should be exactly 1 [Dwo06].

If we wish, we can extend this definition to allow an additive leeway to the probabilities:

Definition 2.2 ((ε, δ)-Differential Privacy, [DR14])**.** *An (ε, δ)-differentially private algorithm $\mathcal{A} : \mathcal{D} \to \mathbb{R}^d$ satisfies the property that if $\forall D, D' \in \mathcal{D}$ differing in exactly one element, and for all $X \subseteq Ran(\mathcal{A})$, we have*

$$\Pr[\mathcal{A}(D) \in X] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in X] + \delta$$

2.2 Notions of privacy in graphs

In graphs and networks, the input data to an algorithm is a highly structured dataset, and maintaining this structure is very important. Thus, there is a more specialized definition when considering protecting the privacy on a graph structure.

This definition of privacy comes from Kasiviswanathan et al. [KNRS13]. There are two types of privacy that we can consider: *edge differential privacy* and *vertex differential privacy* :

Definition 2.3 (Neighboring graphs, [KNRS13])**.** *The graphs G, G' are edge-neighboring if there exist edges e, e' in each graph, respectively, such that $G \setminus e \cong G' \setminus e'$. The graphs G, G' are vertex-neighboring (or node-neighboring) if there exist vertices v, v' in each graph, respectively, such that $G \setminus v \cong G' \setminus v'$.*

We write $G \sim_e G'$ to denote that G and G' are edge neighbors, and $G \sim_v G'$ to denote that they are vertex neighbors.

Definition 2.4 (Differential privacy in graphs, [KNRS13])**.** *An ε-edge differentially private algorithm $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ satisfies the property that if $\forall G, G' \in \mathcal{G}$ such that $G \sim_e G'$, and for all $X \subseteq Ran(\mathcal{A})$,*

$$\Pr[\mathcal{A}(G) \in X] \leq e^{\varepsilon} \Pr[\mathcal{A}(G') \in X]$$

Analogously, we say $\mathcal{A}$ achieves ε-vertex differential privacy if the above holds for all pairs of graphs $G \sim_v G'$.

We will see that designing vertex differentially private algorithms will be slightly more difficult than designing edge differentially private algorithms, simply because the removal of a vertex changes a graph's structure more drastically than the removal of an edge.

Finally, we will be often talking about the set of all graphs $\mathcal{G}$ as a metric space when we need to refer to the "distance" between two graphs. The metric is defined here:

Definition 2.5 ([KNRS13])**.** *Let $d_E : \mathcal{G} \times \mathcal{G} \to \mathbb{Z}_{\geq 0}$ be the edge graph-metric defined as follows: if*

$$G = G_0 \to G_1 \to G_2 \to \cdots \to G_k = G'$$

is a minimum-length sequence of graphs such that $G_i \sim_e G_{i+1}$ for all i, then $d_E(G, G') = k$.

$d_V : \mathcal{G} \times \mathcal{G} \to \mathbb{Z}_{\geq 0}$ is defined similarly, but for $\sim_v$.

2.3 Designing privacy mechanisms

When designing a strategy for ensuring privacy, there are three criteria that we should keep in mind:

- **Security**: how private is our mechanism? This is evaluated based on the specific definition of privacy we use; here we use (ε, δ)-differential privacy as defined above.

However, subsequent definitions of differential privacy may digress from this. For example, Rényi differential privacy [Mir17] relies on an alternative divergence measure (Rényi divergence) whereas (ε, δ)-differential privacy is formulated in term of the max divergence. f-differential privacy [DRS19] relaxes the original definition and uses a tradeoff function f to describe the privacy of a function.

- **Accuracy**: is our mechanism giving meaningful results? A trivial mechanism could simply output random results in order to completely preserve privacy, so we want to ensure some proximity towards the actual result in order for our algorithm to be useful. We will often evaluate these through the use of tail bounds, and see with what probability our algorithm's output deviates from the true value by a large amount (we want this to be small!).
- **Tractability**: can our private result be computed efficiently? We will see that even when we can find a differentially private algorithm for a problem, it may not be practical to use, e.g. having a runtime exponential in the input size.

We will see that often algorithms must make a compromise between these three criteria, but the most important measure of success of an algorithm, from our point of view, is privacy preservation. This is not to say that accuracy and tractability are not important to us; since privacy is the focus of designing differentially private algorithms in the first place, naturally, this is what we would want to guarantee.

2.4 Sensitivity of privacy

Naturally, one concern that is harbored within our definition is how much impact a certain individual will truly have on the output of a dataset analysis. For example, consider datasets that consist of people residing in the United States. The number of people in such a dataset will only be changed by 1 if a person is excluded, but if this dataset consisted of the number of online purchases people make, the exclusion of one individual would change this quantity much more. Fortunately, this notion is captured in the idea of *sensitivity*, how much an algorithm's output will change if a single individual is excluded from our dataset. The simplest and most common form of sensitivity is *global sensitivity*:

Definition 2.6 ([Dwo06])**.** *The global sensitivity of $f : \mathcal{D} \to \mathbb{R}^k$ is defined to be*

$$\Delta(f) = \max_{D, D' :\ D \sim D'} \|f(D) - f(D')\|_1$$

where $\|\cdot\|_1$ denotes the L_1-norm.

This definition captures the sensitivity over all possible neighboring datasets, so global sensitivity tends the most towards a "worst-case" possibility.

In most of our privacy-preserving mechanisms, we will want to add some perturbation proportional to the sensitivity of our statistic (since greater sensitivity means needing more

noise to "cover up" the actual value of our statistic) [DMNS06]. However, when considering problems in graph theory, statistics on a network can change drastically with the removal or addition of a single individual; for example, the number of edges can change by $O(|V|)$ [NRS07].

As a result, most results in privacy-preserving computation in networks employ new reduced sensitivity mechanisms that still preserve the same amount of privacy. In sections 3.2.1 and 4.1.2, we investigate various sensitivity measures that have been formulated to give better accurate and private outputs.

2.5 Composition theorems

A useful property that our definition of differential privacy gives us is that if we wish to run multiple mechanisms sequentially, the privacy parameters nicely compose with each other.

Theorem 2.7 (Composition [MM09, DL09])**.** *Let $\mathfrak{A} = \{\mathcal{A}_i\}_{i=1}^n$ be a sequence of algorithms which are each $(\varepsilon_i, \delta_i)$-differentially private, where $\mathcal{A}_i$ takes the outputs of all algorithms in $\{\mathcal{A}_j\}_{j=1}^{i-1}$ as input. Then any algorithm that takes the output of each algorithm in $\mathfrak{A}$ as input is $(\sum_{i=1}^n \varepsilon_i, \sum_{i=1}^n \delta_i)$-differentially private.*

There exist more advanced composition theorems that give tighter bounds (detailed in Dwork and Roth [DR14]), but we won't be making use of those in this paper.

2.6 Standard methods for preserving privacy

2.6.1 The Laplace mechanism

We now introduce our first mechanism for transforming an algorithm into one with differentially private guarantees. Unfortunately, any deterministic algorithm will not have any good privacy guarantee [DR14]. Therefore, we employ the use of randomization in order to output appropriately.

Recall that the Laplace distribution $\text{Lap}(x|b)$ has the density function

$$\text{Lap}(x|b) = \frac{1}{2b} e^{-|x|/b}$$

Suppose we wish to design an ε-differentially private algorithm $\mathcal{A}' : \mathcal{D} \to \mathbb{R}^k$ for some collection of datasets $\mathcal{D}$, given a pre-existing algorithm $\mathcal{A} : \mathcal{D} \to \mathbb{R}^k$ that may not be differentially private. The *Laplace mechanism* (due to Dwork et al. [DMNS06]) adds noise drawn from a Laplace distribution to the output of our original algorithm:

$$\mathcal{A}'(D) = \mathcal{A}(D) + \underbrace{(Y_1, Y_2, \ldots, Y_k)}_{\mathbf{Y}} \qquad \text{where} \qquad Y_i \sim \text{Lap}\left(\frac{\Delta(\mathcal{A})}{\varepsilon}\right)$$

This simple mechanism is in fact all we need to make our algorithm private:

Theorem 2.8 (Laplace mechanism privacy [DMNS06]). *Suppose $\mathcal{A}'$ is an algorithm created through the Laplace mechanism from algorithm $\mathcal{A}$. Then $\mathcal{A}'$ is ε-differentially private.*

We can use this Laplace mechanism on any algorithm that returns a real-valued vector as its output.

2.6.2 The exponential mechanism

What if the output range of our algorithm isn't a real-valued vector, but rather a set of discrete outputs that don't have a total ordering (for example, outputting a color from the set $\{\text{Red}, \text{Green}, \text{Blue}\}$)? We can no longer perturb the output directly with noise, since our outputs are no longer real-valued. However, we can attempt to still add some "discrete noise" by randomizing the value we output slightly.

This new mechanism, the *exponential mechanism*, is due to McSherry and Talwar [MT07] and was originally developed in the context of auction pricing. Suppose R is the set of outputs of our algorithm. In order to get a sense of how well a certain output is paired with a dataset D, suppose that we have a function $q : \mathcal{D} \times R \to \mathbb{R}$ that represents a measure of how likely a user would prefer a certain output of the algorithm [MT07]. In optimization problems, q is closely associated with the objective function we wish to optimize.

Then, for a given dataset D, our new differentially private algorithm $\mathcal{A}'$ will output a single element of R over the probabilities [MT07]

$$\Pr[\mathcal{A}'(D, q) = k] \propto \exp\{\varepsilon q(D, k)\}$$

If a particular output of our algorithm is "good" for our dataset D, the algorithm will select this value as its output more likely than other unfavorable outputs.

The privacy of the algorithm is related to the sensitivity of the function q with respect to its dataset parameter, ranging over all possible values.

Theorem 2.9 (Exponential mechanism privacy [MT07]). *The exponential mechanism gives $2\varepsilon\Delta(q)$-differential privacy.*

How accurate is the result of this mechanism? As in, how close do we get to the optimal value of $q(D, k)$? This depends on the set of values we wish to output:

Theorem 2.10 (Accuracy of the exponential mechanism [MT07]). *Let $r^* = \mathcal{A}'(D, q)$ be the output of the exponential mechanism. Then,*

$$\Pr\left[q(D, r^*) \leq \max_{r \in R} q(D, r) - \frac{\log|R| + t}{\varepsilon}\right] \leq e^{-t}$$

In other words, the probability that the exponential mechanism outputs a value that yields more than (approximately) $O\left(\frac{\Delta(q)t}{\varepsilon}\right)$ error from optimal is exponentially small in t [MT07].

The fact that our error depends on the logarithm of $|R|$ gives more flexibility in the output set size.

While the exponential mechanism gives good privacy guarantees on discrete outputs, it is difficult to efficiently sample from the above distribution; if $|R|$ is large, we need to compute $\Pr[\mathcal{A}'(D, q) = k]$ for each value $k \in R$. Efficient sampling from this mechanism is often problem-specific; we will see one example in the minimum cut problem in section 3.1.

3 Private combinatorial optimization in graph theory

One major application of graph theory in differential privacy is the setting of classical combinatorial optimization problems. Classical algorithms taught in a introductory algorithms course [CLRS09] often times have efficient implementations but don't do any anything at the individual datum level to preserve privacy (whether it be for relationships/edges or individuals/vertices). We turn to some well-known problems in graph theory and see how the tools of differential privacy can help us design private and accurate algorithms for analysis of graphs and networks.

3.1 Global Min-Cut

Recall that a cut $(S, V \setminus S)$ of an undirected graph $G = (V, E)$ is a partition of the vertices of G into two sets [CLRS09]. The size of the cut, $C_G(S)$ is equal to the number of edges whose endpoints are in different parts of the cut, namely:

$$C_G(S) = |\{\{u, v\} \in E \mid u \in S \text{ and } v \in V \setminus S\}| = |\text{Cut}_G(S)|$$

A property of common interest is the *global minimum cut* [CLRS09]; that is,

$$\text{MC}(G) = \min_{S \subseteq V} C_G(S)$$

Efficient algorithms exist for finding the minimum cut of a graph; a well-known algorithm due to Ford and Fulkerson [FF09] can (in certain implementations like that of Edmonds and Karp [EK72]) find the minimum cut in $O(|V|^3|E|)$ time. However, outputting an exact minimum cut in a graph will lead to a violation of differential privacy [GRU11]. This section will explore the work from Gupta et al. [GRU11] that devise a private algorithm to solve this problem.

Furthermore, we will look at the minimum cut problem as a means of discovering a way of designing private algorithms for optimization, by proving a lower bound on how much additive error we must incur in order to yield a private algorithm [GRU11].

3.1.1 An inefficient application of the exponential mechanism

First, noting that the output of the minimum cut algorithm is a set of discrete choices $S \subseteq V$, one can use the exponential mechanism to give a noisy response of the true minimum cut. What's more is that the quality score q in the exponential mechanism is directly related to the optimization objective in the problem – the number of edges in our cut. Thus, we can try and analyze the following algorithm as a warm-up:

1. Select $(S, V \setminus S)$ to be our minimum cut with probability

$$\Pr[S] \propto \exp\{-\varepsilon C_G(S)\}$$

2. Output $(S, V \setminus S)$.

Note that we want to maximize the quality score q while minimizing the size of the cut, hence the negative sign in the front. The aforementioned algorithm is very simple to describe, and can be shown achieves 2ε-differential privacy, just by using the result from the exponential mechanism, since q has a sensitivity of 1.

In order to compute $C_G(S)$ for a cut S, Gupta et al. [GRU11] use Karger's contraction algorithm [Kar93]. This algorithm runs in time and outputs a minimum cut with probability. However, in terms of efficiency, this algorithm evaluates all $O(2^{|V|})$ possible min-cuts in the graph, so this alone makes this algorithm very inefficient.

Ideally, we only want to sample among cuts that are close enough to the true minimum cut in order to yield a good accuracy. Fortunately, Karger's algorithm gives us a probabilistic bound on how good the outputted cut will be.

Theorem 3.1 ([Kar93])**.** *Fix any cut S such that $C_G(S) \leq k \cdot MC(G)$. Then,*

$$\Pr[\text{Karger's outputs } S] \geq \frac{1}{n^{2k}}$$

Corollary 3.1.1 ([Kar93])**.** *The number of cuts in G with size at most $k \cdot MC(G)$ is at most n^{2k}.*

Proof. Let K be the number of cuts of size at most $k \cdot \mathrm{MC}(G)$, and let $S_1, S_2, \ldots, S_K$ be those cuts in some order. Let E_i be the event that Karger's algorithm outputs cut S_i. Then,

$$\begin{aligned}
\Pr[\text{output of Karger's has size} \leq k \cdot \mathrm{MC}(G)] &= \Pr\left[\bigcup_{i=1}^{K} E_i\right] \\
&= \sum_{i=1}^{K} \Pr[E_i] \\
&\geq K \cdot \frac{1}{n^{2k}}
\end{aligned}$$

This lower bound must be less than 1, so we get that $K \leq n^{2k}$, as desired. □

This is very close to our goal of having an efficient, private algorithm! Instead of computing values of all the cuts, we can use Karger's (a sufficient number of times) to generate all n^{2k} cuts of size $\leq k \cdot \mathrm{MC}(G)$, for a suitable k. The main issue here, however, is that even though Karger's algorithm gives us a small minimum cut, this new algorithm may not be differentially private. In particular, whenever the size of the minimum cut is small, releasing this min-cut consistently may violate privacy [GRU11].

3.1.2 Lower bounds on private min-cut

It turns out that in order to maintain edge-differential privacy for every edge in our graph, we need to add a number of edges logarithmic in the number of vertices. The following lower bound is from Gupta et al. [GRU11]:

Theorem 3.2 ([GRU11])**.** *Any ε-differentially private algorithm $\mathcal{A}$ for private min-cut must satisfy*

$$C_G(\mathcal{A}(G)) \geq MC(G) + \Omega\left(\frac{\ln n}{\varepsilon}\right)$$

in expectation.

To prove this, we need to find a particular graph that enforces this lower bound. In light of this, we prove the following lemma:

Lemma 3.3 ([GRU11])**.** *Suppose $\varepsilon = o(1)$. There exists a graph $G = (V, E)$ that satisfies the following properties:*

1. *For all $v \in V$, $\deg(v) = (1 \pm o(1))\left(\frac{\ln n}{3\varepsilon}\right)$.*
2. *For any cut $(S, V \backslash S)$ such that $2 \leq |S| \leq |V|-2$, we have that $C_G(S) \geq (1-o(1))\left(\frac{\ln n}{2\varepsilon}\right)$.*

Proof. [Rot20] Let $k = \frac{\ln n}{3\varepsilon}$. We show that each property holds in the Erdös-Rényi graph $G(n, \frac{k}{n-1})$ with probability $\geq \frac{3}{4}$.

We start with property 1. Clearly, the expected degree of each vertex is k (WLOG the calculation is done with v_1):

$$\mathbb{E}[\deg(v_1)] = \sum_{i=1}^{n-1} \Pr[\{v_1, v_i\} \in E] = (n-1)\left(\frac{k}{n-1}\right) = k$$

Let D be the degree of a particular vertex v. Note that S is the sum of $n-1$ independent Bernoulli random variables, so applying a Chernoff bound, we obtain

$$\begin{aligned}
\Pr[|D - \mathbb{E}[D]| \geq \delta\mathbb{E}[D]] &\leq \Pr[D \geq (1+\delta)\mathbb{E}[D]] + \Pr[D \leq (1-\delta)\mathbb{E}[D]] && \text{(union bound)} \\
&\leq e^{-\frac{\delta^2\mathbb{E}[D]}{3}} + e^{-\frac{\delta^2\mathbb{E}[D]}{3}} && \text{(by Chernoff)} \\
&= 2e^{-\frac{\delta^2\mathbb{E}[D]}{3}}
\end{aligned}$$

Letting $\delta = \sqrt{\frac{3\ln(n/\delta_0)}{\mathbb{E}[S]}}$, the inequality becomes

$$\Pr[|D - \mathbb{E}[D]| \geq \sqrt{3\mathbb{E}[D]\ln(n/\delta_0)}] \leq \frac{2}{n}\delta_0$$

for all $\delta_0 \in [0, 1]$. Thus, with probability at least $1 - \frac{2}{n}\delta_0$, the degree of a particular vertex v is in the range

$$k \pm \sqrt{3k\ln(n/\delta_0)}$$

Union bounding over all the vertices, we have that all vertices fall in this range with probability at least $1 - 2\delta_0$. Finally, taking $\delta_0 = \frac{1}{4}$ and noting that $k = \frac{\ln n}{3\varepsilon}$ and $\varepsilon = o(1)$, we have that this range is

$$k \pm \sqrt{3k\ln(n/\delta_0)} = k\left(1 \pm \sqrt{\frac{3\ln(n/\delta_0)}{k}}\right) = k(1 \pm o(1))$$

This shows property 1 holds with probability at least $\frac{1}{2}$. Now assume property 1 holds; for property 2, fix a cut $(S, V \setminus S)$. WLOG, assume that $|S| \leq \frac{n}{2}$. Then the expected size of the cut is

$$\mathbb{E}[C_G(S)] = |S| \cdot (n - |S|) \cdot \frac{k}{n-1} = O(k|S|)$$

Let C be a random variable denoting the size of the cut $C_G(S)$. Applying a Chernoff bound to this gives

$$\Pr[C \leq \mathbb{E}[C] - \sqrt{2\mathbb{E}[C]\ln(1/\delta_1)}] \leq \delta_1$$

Thus, with probability $1-\delta_1$, the size of a particular cut is at least $O(k|S|) - \sqrt{2k|S|\ln(1/\delta_1)}$. Now note that there are at most $\binom{n}{|S|} \leq n^{|S|}$ cuts with a partition of size $|S|$. Thus, by taking $\delta_1 = \frac{1}{2n^{|C|+1}}$ and union bounding over all cuts of size $|S|$, we have that with probability $1 - \frac{1}{2n}$, all cuts of size $|S|$ have sizes at least the above amount. But union bounding again over all possible vertex sizes of the cut (from 2 to $n/2$), we get that all cuts satisfy the desired property with probability at least $1 - \frac{1}{2} = \frac{1}{2}$.

Therefore, combining the results from both properties, we get that both of them hold with probability at least $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$, so such a graph must exist [Rot20]. □

Proof of Theorem 3.2. This proof is due to Gupta et al. [GRU11]. Once again, let $k = \frac{\ln n}{3\varepsilon}$. Let $G = (V, E)$ be the graph from Lemma 3.3. Let $\mathcal{A}$ be a ε-differentially private algorithm for Min-Cut. Consider the probability distribution of the output of $\mathcal{A}(G)$. Let $(\{v\}, V \setminus \{v\})$ be a particular single cut of G. There are n singleton cuts, so the probability that $\mathcal{A}$ outputs any particular singleton cut is $\leq \frac{1}{n}$:

$$\Pr[\mathcal{A}(G) = (\{v\}, V \setminus \{v\})] \leq \frac{1}{n}$$

Now, let

$$G' = G \setminus \{\{v, w\} \in E\}$$

which is simply G after removing all edges incident on v. Note that the min-cut in G' is zero since v is an isolated vertex. Since G and G' differ in at most $(1 \pm o(1))k$ edges, by our differential privacy guarantee we have

$$\begin{aligned}\Pr[\mathcal{A}(G') = (\{v\}, V \setminus \{v\})] &\leq e^{(1\pm o(1))k\varepsilon} \Pr[\mathcal{A}(G) = (\{v\}, V \setminus \{v\}) \\ &\leq n^{(1+o(1))/3} \cdot \frac{1}{n} \\ &= \frac{1}{n^{2/3}}\end{aligned}$$

for sufficiently large n. Note that in G', the cut $(\{v\}, V \setminus \{v\})$ is the min-cut since it has size zero. All other cuts have size at most $\frac{\ln n}{2\varepsilon} - \frac{\ln n}{3\varepsilon} = \Omega(\frac{\ln n}{\varepsilon})$. Thus, the expected size of the cut produced is at least

$$\left(1 - \frac{1}{n^{2/3}}\right) \cdot \Omega\left(\frac{\ln n}{\varepsilon}\right) = \Omega\left(\frac{\ln n}{\varepsilon}\right)$$

via the proof of Gupta et al. [GRU11]. □

3.1.3 Take 2: an improved min-cut algorithm

Thus, we investigate the approach detailed by Gupta et al. [GRU11] to fix these issues. Their algorithm randomly adds an appropriate amount of edges to lower bound the size of the minimum cut, and then runs the original algorithm. We present their algorithm here [GRU11]:

1. Arbitrarily fix a sequence of sets of edges on n vertices:

$$\emptyset = H_0 \subset H_1 \subset \cdots \subset H_{\binom{n}{2}-1} \subset H_{\binom{n}{2}} = E(K_n)$$

2. Select i with probability

$$\Pr[i] \propto \exp\left\{-\varepsilon \left|\mathrm{MC}(G \cup H_i) - \frac{8 \ln n}{\varepsilon}\right|\right\}$$

3. Run Karger's algorithm n^7 times on $G \cup H_i$ to generate n^7 cuts.

4. Select S from this set of cuts to be our minimum cut with probability

$$\Pr[S] \propto \exp\left(-\varepsilon C_{G \cup H_i}(S)\right)$$

This new, improved algorithm makes use of the exponential mechanism twice: once (with $q(G, i) = -|\mathrm{MC}(G \cup H_i) - \frac{8 \ln n}{\varepsilon}|$) to select the edges to add, and again (with $q(G \cup H_i, S) = C_{G \cup H_i}(S)$) to select the cut. It also runs Karger's algorithm a polynomial number of times to generate a set of candidate cuts that are (with high probability) close to the true min-cut size.

Lemma 3.4 ([GRU11])**.** *With high probability, the optimal min-cut of the graph $G \cup H_i$ is in the range*

$$\left[\frac{4\ln n}{\varepsilon}, \mathrm{MC}(G) + \frac{12\ln n}{\varepsilon}\right]$$

Proof. This proof is due to Gupta et al. [GRU11]. Suppose $\mathrm{MC}(G) > \frac{8\ln n}{\varepsilon}$. Then we know that $\mathrm{MC}(G \cup H_i) > \frac{4\ln n}{\varepsilon}$. Furthermore, the optimal value of $q(G, i)$ (as stated above) attained at $i = 0$, since the min-cut only increases as i increases. Thus, by Theorem 2.10,

$$\Pr\left[\left|\mathrm{MC}(G \cup H_i) - \frac{8\ln n}{\varepsilon}\right| \geq \left(\mathrm{MC}(G) - \frac{8\ln n}{\varepsilon}\right) + \frac{1}{\varepsilon}(2\ln n + t)\right] \leq e^{-t}$$

$$\Pr\left[\left|\mathrm{MC}(G \cup H_i) - \frac{8\ln n}{\varepsilon}\right| \geq \mathrm{MC}(G) - \frac{4\ln n}{\varepsilon}\right] \leq \frac{1}{n^2}$$

$$\Pr\left[\mathrm{MC}(G \cup H_i) \geq \mathrm{MC}(G) + \frac{4\ln n}{\varepsilon}\right] \leq \frac{1}{n^2}$$

so the probability $\mathrm{MC}(G)$ falls outside of the claimed interval is upper-bounded by $\frac{1}{n^2}$.

Next, suppose $\mathrm{MC}(G) < \frac{8\ln n}{\varepsilon}$. Since the min-cut changes by at most 1 when considering consecutive terms in the sequence

$$\mathrm{MC}(G \cup H_0), \mathrm{MC}(G \cup H_1), \ldots, \mathrm{MC}(G \cup H_{\binom{n}{2}})$$

we must have some i that achieves $\mathrm{MC}(G \cup H_i) = \frac{8\ln n}{\varepsilon}$. Thus, by Theorem 2.10 again,

$$\Pr\left[\left|\mathrm{MC}(G \cup H_i) - \frac{8\ln n}{\varepsilon}\right| \geq \frac{1}{\varepsilon}(2\ln n + t)\right] \leq e^{-t}$$

$$\Pr\left[\left|\mathrm{MC}(G \cup H_i) - \frac{8\ln n}{\varepsilon}\right| \geq \frac{4\ln n}{\varepsilon}\right] \leq \frac{1}{n^2} \qquad \text{(letting } t = 2\ln n\text{)}$$

$$\Pr\left[\left(\mathrm{MC}(G \cup H_i) \geq \frac{12\ln n}{\varepsilon}\right) \cup \left(\mathrm{MC}(G \cup H_i) \leq \frac{4\ln n}{\varepsilon}\right)\right] \leq \frac{1}{n^2}$$

so the result follows, via the proof of Gupta et al. [GRU11]. □

Theorem 3.5 ([GRU11])**.** *Assuming Lemma 3.4, with high probability, the algorithm chooses i and outputs a cut S such that*

$$C_{G \cup H_i}(S) \leq \mathrm{MC}(G \cup H_i) + O\left(\frac{\ln n}{\varepsilon}\right)$$

Proof. This proof is due to Gupta et al. [GRU11]. Assume that i is selected to satisfy the condition of Lemma 3.4. Then, by Theorem 3.1.1, the number of cuts in $G \cup H_i$ that have size at most $\mathrm{MC}(G \cup H_i) + t$ for some t is

$$c_t = n^{2\left(\frac{\mathrm{MC}(G \cup H_i) + t}{\mathrm{MC}(G \cup H_i)}\right)} = n^{2 + \frac{2t}{\mathrm{MC}(G \cup H_i)}} \leq n^2 \cdot n^{\frac{2t}{4\ln n/\varepsilon}} = n^2 e^{\frac{\varepsilon t}{2}} \tag{1}$$

We want to use this to bound the tail probability of the outputted cut size being too high, so we will consider each probability of the cut size being equal to a certain amount. Note that the number of cuts of size equal to $\mathrm{MC}(G \cup H_i) + t$ is $c_t - c_{t-1}$.

The n^7 cuts $\mathcal{S}$ outputted by Karger's algorithm must include the min-cut of $G \cup H_i$ with very high probability; indeed, the chance it does not include it is

$$\left(1 - \frac{1}{n^2}\right)^{n^7} \leq \frac{1}{e^{n^5}}$$

So for a cut $(S, V \setminus S)$ with size $\mathrm{MC}(G \cup H_i) + t$, the probability our algorithm outputs this cut is

$$\Pr[S] = \frac{\exp(-\varepsilon(\mathrm{MC}(G \cup H_i) + t)}{\sum_{S \in \mathcal{S}} \exp(-\varepsilon C_{G \cup H_i}(S))} \leq \frac{\exp(-\varepsilon(\mathrm{MC}(G \cup H_i) + t)}{\exp(-\varepsilon \mathrm{MC}(G \cup H_i)} = \exp(-\varepsilon t)$$

and so we can bound the tail probability by

$$\begin{aligned}
\Pr[C_{G \cup H_i}(S) \geq \mathrm{MC}(G \cup H_i) + t] &\leq \sum_{t' \geq t} (c_{t'} - c_{t'-1}) \exp(-\varepsilon t') \\
&= \sum_{t' > t} e^{-\varepsilon t'} c_{t'} - \sum_{t' \geq t-1} e^{-\varepsilon t' - \varepsilon} c_{t'} \\
&\leq \sum_{t' \geq t} e^{-\varepsilon t'} c_{t'} - \sum_{t' \geq t} e^{-\varepsilon t' - \varepsilon} c_{t'} \\
&= (1 - \exp(-\varepsilon)) \sum_{t' \geq t} e^{-\varepsilon t'} c_{t'} \\
&\leq (1 - \exp(-\varepsilon)) \sum_{t' \geq t} e^{-\varepsilon t'/2} \cdot n^2 \qquad \text{(by (1))} \\
&\leq n^2 (1 - \exp(-\varepsilon)) \left(\frac{e^{-\varepsilon t/2}}{1 - e^{-\varepsilon/2}}\right)
\end{aligned}$$

Letting $t = \frac{8 \ln n}{\varepsilon}$ gives

$$\begin{aligned}
&= n^2 \cdot n^{-4} \cdot \underbrace{\frac{1 - e^{-\varepsilon}}{1 - e^{-\varepsilon/2}}}_{\text{decreasing in } \varepsilon, \text{ so } O(1)} \\
&= \frac{1}{n^2}
\end{aligned}$$

and the claim follows, via the proof of Gupta et al. [GRU11]. □

Corollary 3.5.1 ([GRU11])**.** *With high probability, the algorithm outputs a cut S such that*

$$C_G(S) \leq MC(G) + O\left(\frac{\ln n}{\varepsilon}\right)$$

Proof. The proof is due to Gupta et al. [GRU11]. Lemma 3.4 tells us that

$$\mathrm{MC}(G \cup H_i) - \mathrm{MC}(G) < \frac{12 \ln n}{\varepsilon}$$

with high probability (namely, at least $1 - \frac{1}{n^2}$). Theorem 3.5 says that

$$C_{G \cup H_i}(S) - \mathrm{MC}(G \cup H_i) < \frac{8 \ln n}{\varepsilon}$$

Adding these two inequalities gives

$$C_{G \cup H_i}(S) - \mathrm{MC}(G) < \frac{20 \ln n}{\varepsilon} \implies C_G(S) - \mathrm{MC}(G) < \frac{20 \ln n}{\varepsilon}$$

since any cut S must have higher size in $G \cup H_i$ than G. This proves the claim, via the proof of Gupta et al. [GRU11]. □

Theorem 3.6 ([GRU11])**.** *The above algorithm is $(4\varepsilon, O(\frac{1}{n^2}))$-edge differentially private.*

Proof. This proof is due to Gupta et al. [GRU11]. For now, suppose that the algorithm $\mathcal{A}$ didn't run Karger's algorithm n^7 times and instead sampled from all min-cuts. This algorithm (call it $\mathcal{A}'$) is the composition of two applications of the exponential mechanism with privacy parameter $2\varepsilon\delta(q) = 2\varepsilon$, so the privacy of this algorithm is 4ε.

Now, consider an auxiliary algorithm $\mathcal{A}_{\text{fail}}$ that runs Karger's algorithm n^7 times, but then samples among all possible cuts in step 4, and outputs FAIL if the chosen cut is not among the n^7 computed cuts. We show that $\Pr[\mathcal{A}_{\text{fail}} \text{ outputs FAIL}] = O(\frac{1}{n^2})$.

First, note that by Lemma 3.4, $\mathrm{MC}(G \cup H_i) > \frac{4 \ln n}{\varepsilon}$ with probability at least $1 - \frac{1}{n^2}$. Thus, conditioning on this, let F be the event where the outputted cut has a size greater than $3\mathrm{MC}(G \cup H_i)$, and let E be the event that $\mathcal{A}_{\text{fail}}$ outputs FAIL.

$$\begin{aligned}\Pr[\mathcal{A}_{\text{fail}} \text{ outputs FAIL}] = \Pr[E] &= \Pr[E \mid F] \Pr[F] + \Pr[E \mid \bar{F}] \Pr[\bar{F}] \\ &\leq \Pr[F] + \Pr[E \mid \bar{F}]\end{aligned}$$

We saw that from Theorem 3.5, $\Pr[F] \leq \frac{1}{n^2}$. For $\Pr[E \mid \bar{F}]$, by Theorem 3.1, the probability that Karger's outputs a certain cut of size at most $3\mathrm{MC}(G \cup H_i)$ is at least $\frac{1}{n^6}$. Thus, the probability that n^7 runs of Karger's algorithm don't output that particular cut is at most

$$\left(1 - \frac{1}{n^6}\right)^{n^7} \leq e^{-\frac{n^7}{n^6}} = \frac{1}{e^n}$$

Thus, $\Pr[E \mid \bar{F}] \leq \frac{1}{e^n} = O(\frac{1}{n^2})$, and therefore $\Pr[\mathcal{A}_{\text{fail}} \text{ outputs FAIL}] = O(\frac{1}{n^2})$.

Finally, when we consider the three algorithms $\mathcal{A}$, $\mathcal{A}$ and $\mathcal{A}_{\text{fail}}$, we note that the difference between the probabilities of outputting a particular cut in the pair $(\mathcal{A}', \mathcal{A}_{\text{fail}})$ is at most $O(\frac{1}{n^2})$, and similarly for $(\mathcal{A}_{\text{fail}}, \mathcal{A})$. Thus, the probabilities in the output distributions of $(\mathcal{A}', \mathcal{A})$ must differ by $O(\frac{1}{n^2})$; namely:

$$|\Pr[\mathcal{A}(G) = S] - \Pr[\mathcal{A}'(G) = S]| \leq O\left(\frac{1}{n^2}\right)$$

so we can conclude that, for $G \sim_e G'$,

$$\begin{aligned}
\Pr[\mathcal{A}(G) = S] &\leq \Pr[\mathcal{A}'(G) = S] + O\left(\frac{1}{n^2}\right) \\
&\leq e^{4\varepsilon} \Pr[\mathcal{A}'(G')] + O\left(\frac{1}{n^2}\right) \\
&\leq e^{4\varepsilon} \left(\Pr[\mathcal{A}(G') = S] + O\left(\frac{1}{n^2}\right)\right) + O\left(\frac{1}{n^2}\right) \\
&= e^{4\varepsilon} \Pr[\mathcal{A}(G') = S] + O\left(\frac{1}{n^2}\right)
\end{aligned}$$

and $\mathcal{A}$ is $(4\varepsilon, O(\frac{1}{n^2}))$-edge differentially private, by the proof of Gupta et al. [GRU11]. □

Thus, we have a nice implementation of a private minimum cut algorithm. Gupta et al. [GRU11] use this similar technique of proving a lower bound to guarantee privacy for several other combinatorial optimization problems (not necessarily graph-theoretic).

3.2 MST Cost

Another way of achieving a private algorithm in optimization is by employing similar mechanisms to that of the Laplace mechanism, but find a way to reduce the noise significantly in a way that still makes the mechanism private. In this section, we investigate a differentially private algorithm from Nissim et al. [NRS07] that departs from a simple application of one of the aforementioned algorithm. Suppose we now have a weighted graph $G = (V, E)$ with an weight function $w : E \to \mathbb{R}$. For simplicity, assume that the weights of the edges are bounded above by some constant K.

Recall that there are efficient algorithms to compute the cost of an MST, such as Kruskal's [Kru56] or Prim's [Pri57] algorithm. However, outputting the exact cost of the MST would yield a leak of privacy. Therefore, our goal is to perturb the cost of the MST by a sufficient amount as to preserve privacy. We consider an edge differentially private notion of privacy here, with the additional condition that two graphs can also be neighbors if exactly one of the weights of their edges is different.

So, we consider two graphs to be edge-neighbors if their weight functions have Hamming distance 1. Note that if we simply apply the Laplace mechanism to this, the global sensitivity of the MST cost function, f_{MST}, would be K, since for a complete graph with all edge weights equal to K, we can change its weight to 0 and change the MST cost by K [NRS07]. We will see that we can reduce this noise by a significant margin!

We refer to the approach given in Nissim et al. [NRS07]. The high level idea of their approach is to introduce an alternative notion of sensitivity known as *smooth sensitivity*.

3.2.1 Smooth Sensitivity

Global sensitivity provides too much leeway for the output of a private function due to the fact that it considers the worst-case scenario across all possible instances of our dataset [NRS07]. As we saw, this worst case scenario isn't very fruitful for our algorithm in the vast majority of instances. To curb this issue, we can look at the other extreme of sensitivity– what if we considered perturbation around the specific dataset our algorithm is operating on? Instead of looking at all pairs of differing datasets, we can fix one of the elements of the pair and consider perturbing around that:

Definition 3.7 ([NRS07])**.** *The local sensitivity of $f : \mathcal{D} \to \mathbb{R}^d$ is defined to be*

$$LS_f(D) = \max_{D':\ D \sim D'} \|f(D) - f(D')\|_1$$

So why don't we just use noise according to local sensitivity rather than global sensitivity? The intuitive answer is that changing the noise based on the example may reveal some characteristics of the original dataset; particular inputs may be more sensitive than others and thus will result in drastically different outputs [NRS07]. Thus, much of the existing literature tries to find an appropriate compromise between global and local sensitivity.

Smooth sensitivity, among other measures of sensitivity, are inspired by the high global sensitivity of graph statistics; the removal of a single vertex or edge can have a large impact on the structure of a graph.

Definition 3.8 ([NRS07])**.** *A β-smooth upper bound S_f for LS_f satisfies the following properties:*

1. *For all $D \in \mathcal{D}$, $S_f(D) \geq LS_f(D)$*
2. *For all $D, D' \in \mathcal{D}$ such that $D \sim D'$, $S_f(D) \leq e^{\beta} S_f(D')$.*

Definition 3.9 ([NRS07])**.** *The smooth sensitivity of $f : \mathcal{D} \to \mathbb{R}$, parameterized by β, is defined to be*

$$S^*_{f,\beta}(D) = \max_{D'} \left(LS_f(D') e^{-\beta d(D,D')} \right)$$

To understand the motivations behind these definitions, observe that the smooth upper bound on LS_f seeks to be a compromise between local and global sensitivity [NRS07]. The second condition of its definition is to make sure that our noise doesn't change too much between neighboring inputs [NRS07]. Since our upper bound is directly related to the smooth sensitivity, this ensures there is a constant upper bound to the change between small perturbations.

The definition of smooth sensitivity is very similar to that of global sensitivity, except with an added exponential term. This allows global sensitivity to consider all pairs of datasets, but additionally weighting distant datasets less.

It turns out that this form for the smooth sensitivity has the following property:

Theorem 3.10 ([NRS07])**.** *The smooth sensitivity $S^*_{f,\beta}$ is a β-smooth upper bound for LS_f. Furthermore, $S^*_{f,\beta}$ is the minimum such upper bound, i.e. for all β-smooth upper bounds S, we have that $S^*_{f,\beta}(D) \leq S(D)$.*

Proof. This proof is due to Nissim et al. [NRS07]. It is easy to see that $S^*_{f,\beta}(D) \geq LS_f(D')$ by letting $D' = D$. To show β-smoothness, suppose $D \sim D'$. Let D^* be the dataset that achieves the maximum in $S^*_{f,\beta}(D)$. Then,

$$\begin{aligned}
S^*_{f,\beta}(D) &= LS_f(D^*)e^{-\beta d(D,D^*)} && \text{(by definition of } D^*) \\
&\leq LS_f(D^*)e^{-\beta(d(D',D^*)-d(D,D'))} && \text{(by the triangle inequality)} \\
&= LS_f(D^*)e^{-\beta(d(D',D^*)-1)} \\
&= e^{\beta} LS_f(D^*)e^{-\beta d(D',D^*)} \\
&\leq e^{\beta} S^{(}_{f,\beta} D')
\end{aligned}$$

thus proving the claim, via the proof from Nissim et al. [NRS07]. □

Theorem 3.11 ([NRS07])**.** *Suppose $\mathcal{A}'$ is an algorithm created via perturbing the output of $\mathcal{A}$ by:*

$$\mathcal{A}'(D) = \mathcal{A}(D) + \text{Lap}\left(\frac{2S_{f,\beta}(D)}{\varepsilon}\right)$$

where $\beta = \frac{\varepsilon}{2\ln(1/\delta)}$. Then, $\mathcal{A}'$ is (ε, δ)-differentially private.

The above theorem shows that if we can efficiently compute the smooth sensitivity for any problem we wish to apply our modified Laplace mechanism to, we can greatly reduce the noise involved and still maintain privacy.

It will become more useful, however, to rewrite the smooth sensitivity as in Nissim et al. [NRS07]:

$$\begin{aligned}
S^*_{f,\beta}(D) &= \max_{D'} \left(LS_f(D')e^{-\beta d(D,D')}\right) \\
&= \max_{k\in[0..n]} \left(e^{-\beta k} \max_{D':\ d(D,D')=k} LS_f(D')\right)
\end{aligned}$$

so we define the *sensitivity at distance k* [[NRS07]] to be

$$A^{(k)}_f(D) = \max_{D':\ d(D,D')\leq k} LS_f(D')$$

and thus we can rewrite the smooth sensitivity as

$$S^*_{f,\beta}(D) = \max_{k\in[0..n]} e^{-\beta k} A^{(k)}_f(D)$$

Notice that this is an equality rather than an equality since even if $A^{(k)}_f(D)$ achieved the maximum at some $k' < k$, it would also be shadowed by $A^{(k')}_f(D)$ and thus does not matter.

This means that if we can compute the function $A_f^(k)$, we can compute the smooth sensitivity of f. We call this function the *sensitivity at distance k* [NRS07]. This function is a lot easier to reason about for specific problems since it removes the exponential term and leaves only the local sensitivity terms, which are easier to work with.

3.2.2 Computing smooth sensitivity

Before we prove a result about the smooth sensitivity of the MST, we first show the smooth sensitivity of the minimum function, which is involved in the computation of MST sensitivity. Assume that the input to the minimum function is sorted: $0 \leq x_1 \leq \cdots \leq x_n \leq K$.

Lemma 3.12 ([NRS07])**.** *The sensitivity at distance k of the minimum function, f_{min}, is*

$$A^{(k)}(x) = \max(x_{k+1}, x_{k+2} - x_1)$$

Proof. The proof is due to Nissim et al. [NRS07]. Note that the local sensitivity of the sequence $x_1, \ldots, x_n$ is given by

$$LS_{f_{\min}}(x_1, \ldots, x_n) = \max(x_1, x_2 - x_1)$$

since only two perturbations can change the minimum: changing x_1 to 0, or increasing x_1 past x_2. Using the same logic, we can look at the following two vectors that are a distance k away from $x_1, \ldots, x_n$:

$$0, 0, \ldots, 0, x_{k+1}, \ldots, x_n \qquad \text{and} \qquad x_1, 0, 0, \ldots, 0, x_{k+2}, \ldots, x_n$$

The first sequence has a local sensitivity of x_{k+1}, and the second has a local sensitivity of $x_{k+2} - x_1$. Thus the claim holds, from the proof of [NRS07]. □

The main result we will prove in this section is the expression for $A^{(k)}_{\text{MST}}$, the local sensitivity at distance k for the MST cost function:

Theorem 3.13 ([NRS07])**.** *Let $G = (V, E)$ be a graph with weight function w. The local sensitivity at distance k for the MST cost function has the form*

$$A_f^{(k)}(G) = \max_{S \subset V} A^{(k)}_{f_{\min}}(\{w(e) \mid e \in \text{Cut}_G(S)\})$$

where $f_{\min}$ denotes the minimum function.

Sketch of proof. It will suffice to prove that

$$LS_f(G) = \max_{S \subset V} LS_{f_{\min}}(\{w(e) \mid e \in \text{Cut}_G(S)\})$$

The proof for this stems from the following fact: for any MST T of G and any cut $S \subset V$ of G, T must contain the minimum edge crossing the cut $(S, V \setminus S)$ (otherwise, one could form a smaller MST by using the minimum weight edge across the cut). Thus a change in

the minimum weight edge across a cut will affect the cost of the MST, and so changing the MST cost by changing the weight of an edge will be determined by the maximum change across all cuts of the minimum weight edge.

To prove the original claim, we show the proof from Nissim et al. [NRS07]:

$$\begin{aligned} A_f^{(k)}(G) &= \max_{G':\ d(G,G')\le k} LS_f(G') \\ &= \max_{G':\ d(G,G')\le k} \max_{S\subset V(G')} LS_{f_{\min}}(\{w'(e) \mid e \in \mathrm{Cut}_{G'}(S)\}) \end{aligned}$$

Since G and G' are on the same set of vertices, we can exchange the order of the maxima:

$$\begin{aligned} &= \max_{S\subset V} \max_{G':\ d(G,G')\le k} LS_{f_{\min}}(\{w'(e) \mid e \in \mathrm{Cut}_{G'}(S)\}) \\ &= \max_{S\subset V} A_{f_{\min}}^{(k)}(\{w(e) \mid e \in \mathrm{Cut}_G(S)\} \end{aligned}$$

proving the claim via the proof from NIssim et al. [NRS07]. □

This gives us a nice way to compute the smooth sensitivity and give a better privacy result for the MST! The only thing we have to show is the efficiency of implementing this procedure, since naïvely implementing the computation of $A^{(k)}(G)$ would be highly efficient as we will be iterating through all the cuts of the graph. For the details of such computation, see Nissim et al. [NRS07]. We note that the computation of the sensitivity of $f_{\min}$ takes constant time, so this is not the bottleneck of computation.

Due to this algorithm, the sensitivity no longer directly relies on the maximum weight edge in the graph; instead, smooth sensitivity makes it such that the maximum weight edges are given less weight in the overall maximum due to the $e^{-\beta d(D,D')}$ term. Thus, our smooth sensitivity will be much lower than the global sensitivity.

This is one of the earliest results in reducing the noise for a differentially private algorithm. More recent results [Sea15, Pin18] attempt to do more than just release the MST cost; Sealfon [Sea15] releases a spanning tree (close in weight to the actual MST) in a privacy setting in which edges are publicly known and weights are unknown, and neighboring graphs differ by magnitude 1 in the L_1-norm. Pinot [Pin18] uses a slightly different neighbor metric for weighted graphs and use an algorithm similar to Prim's algorithm in order to compute a private MST.

4 Projection-based methods for graph differential privacy

Upon until results about smooth sensitivity and its applications to edge differential privacy were published, there was not a lot of literature regarding vertex differentially private algorithms [KNRS13]. These algorithms would be tougher to devise due to the more disruptive nature of removing a vertex versus removing an edge.

In this section, we introduce projection-based methods, a set of more recent techniques that are useful in designing vertex-differentially private algorithms (and can also be applied to edge-differentially algorithms as well). Recent literature around designing differentially private algorithms for graph problems has revolved this same central theme. The motivation behind their shared ideas comes from highlighting which graphs are the primary troublemakers in adding sensitivity-based noise. In particular, graphs with vertices of high degree will have high global sensitivity with the removal of a vertex for many different graph statistics, outlined in Kasiviswanathan et al. [KNRS13].

4.1 Restricted sensitivity

4.1.1 Background

Rather than looking at the whole class of graphs, Blocki et al. [BBDS12] temporarily restricts its view to low-degree graphs; in particular graphs whose vertices all have degree at most some constant. In fact, this is a good assumption; in practice, many large graphs and social networks are sparse, i.e. the degree of a vertex is much smaller than the total number of vertices [BBDS12]. Blocki et al. [BBDS12] use the examples of telephone calls, the Internet graph, and Facebook's friendships to illustrate that this assumption is valid. Let

$$\mathcal{G}_D = \{G = (V, E) \mid \deg(v) \leq D, \forall v \in V\} \subset \mathcal{G}$$

The key insight here is that within these set of low-degree graphs, functions that are computed on these graphs will (theoretically) satisfy some nice properties. Using this idea, Blocki et al. [BBDS12] introduce a new notion of sensitivity in order look solely at nice, low-degree graphs:

Definition 4.1 ([BBDS12])**.** *The restricted sensitivity of f under the set $\mathcal{G}_D \subset \mathcal{G}$ is defined to be*

$$RS_f(\mathcal{G}_D) = \max_{G_1, G_2 \in \mathcal{G}_D} \frac{|f(G_1) - f(G_2)|}{d(G_1, G_2)}$$

Restricted sensitivity will allow us to narrow our scope of graphs to a much smaller one while simultaneously allowing us to use much smaller noise; in fact, for most graphs $G \in \mathcal{G}_D$, we have that $LS_f(G) \gg RS_f(\mathcal{G}_D)$ [BBDS12].

The goal here is to find an algorithm that gives good guarantees on graphs in $\mathcal{G}_D$, and then extend this behavior to the rest of the graphs in $\mathcal{G}$. In particular, we wish to achieve good accuracy still on sparse graphs.

We also introduce some other definitions used by Blocki et al. [BBDS12]:

Definition 4.2 ([RS15])**.** *Let $(X, d_X), (Y, d_Y)$ be metric spaces. A function $f : X \to Y$ is Lipschitz continuous*[1] *with Lipschitz constant c (a c-Lipschitz function) if*

$$d_Y(f(x_1), f(x_2)) \leq c \cdot d_X(x_1, x_2)$$

[1]The original paper uses the term "c-smooth" to refer to this notion; we instead refer to it via Lipschitz continuity in order to not clash with the terminology of smooth sensitivity.

for all $x_1, x_2 \in X$.

What Lipschitz continuity intuitively says is that the proximity of points in a space should not change by more than a constant multiplicative factor. This ensures that points that are close by aren't shifted around too much when mapped to a new space. Since our metric spaces are discrete in the case of the space of graphs (endowed with the graph neighbor metric), we would like neighboring graphs to not result in a drastically different results under our projection operator μ [BBDS12].

4.1.2 Restricted sensitivity for edge privacy

In the world of edge-differential privacy, restricted sensitivity actually works quite nicely. Blocki et al. [BBDS12] define a projection operator $\mu : \mathcal{G} \to \mathcal{G}_D$ that converts an arbitrary graph to one of constant max degree D. This projection operator is meant to be used in conjunction with a function f that operates well on graphs in $\mathcal{G}_D$ (by which we mean, gives good privacy guarantees). Their work, along with others, make heavy use of Lipschitz (equivalently, Lipschitz continuous) functions:

Fortunately, Blocki et al. [BBDS12] prove the following results regarding projection operators:

Lemma 4.3 ([BBDS12])**.** *Let* $(\mathcal{G}, d_E)$ *be a metric space of the set of graphs equipped with the edge neighbor metric* d_E*. Then there exists a 3-Lipschitz projection operator* $\mu_E : \mathcal{G} \to \mathcal{G}_D$*.*

The intuition for constructing the projection operator μ_E is to fix some ordering of the edges, and for each vertex with degree at least D, deleting those edges exceeding some index in the ordering that makes the degree at most D.

Theorem 4.4 ([BBDS12])**.** *Let* $f : \mathcal{G} \to \mathbb{R}$*. The algorithm*

$$\mathcal{A}_f(G) = f(\mu_E(G)) + \text{Lap}\left(\frac{3 \cdot RS_f(\mathcal{G}_D)}{\varepsilon}\right)$$

is ε*-edge differentially private.*

This gives a very clean result for providing edge privacy on graphs; simply project our graph in question, and add noise to the computation on this reduced graph.

4.1.3 Restricted sensitivity for vertex privacy

The situation is different in the world of vertex privacy. In particular, because of the increased sensitivity when removing a vertex, the construction of our projection operator is a little more involved in this case. We leave out most of the details out in this paper since the expression for the noise addition is rather complex, but we will summarize the results here.

The main result that prevents us from using a similar projection scheme is the following hardness result:

Theorem 4.5 ([BBDS12]). *Unless $P = NP$, no efficient projection operator $\mu_V : \mathcal{G} \to \mathcal{G}_D$ exists that gives an $O(\ln D)$-approximation to the distance from a graph G to $\mathcal{G}_D$:*

$$d_V(G, \mu_V(G)) \leq O(\ln D) d_V(G, \mathcal{G}_D)$$

This theorem tells us that we can't design an $O(1)$-Lipschitz projection operator under the vertex neighbor metric. If such an efficient operator μ_V existed (with constant $c = O(1)$), we could take a sequence of graphs $G = G_0 \to G_1 \to G_2 \to \cdots \to G_t \in \mathcal{G}_D$ such that $G_i \sim_v G_{i+1}$ and $t = d_V(G, \mathcal{G}_D)$ [BBDS12]. Then, since μ_V is c-Lipschitz, we have that $d_V(\mu_V(G_i), \mu_V(G_{i+1})) \leq c$, and thus by the triangle inequality,

$$d_V(\mu(G_0), \mu(G_t)) = d_V(\mu(G), G_t) \leq ct$$

and so [BBDS12],

$$d_V(G, \mu_V(G)) \leq d_V(G, G_t) + d_V(G_t, \mathcal{G}_D) \leq ct + t = O(1) d_V(G, \mathcal{G}_D$$

To get around this, Blocki et al. [BBDS12] instead allow our projection operator to map to graphs in $\mathcal{G}_{2D}$ and equip this to a *distance estimator* $\hat{d}$ which, roughly, is a c-Lipschitz function taking values between $d(G, \mu_V(G))$ and $c \cdot d(G, \mathcal{G}_D)$.

Then, Blocki et al. [BBDS12] create a vertex-differentially private algorithm by adding noise proportional to the following β-smooth upper bound on $f \circ \mu_V$:

$$S_{f,\beta}(G) = \max_{d \in \mathbb{Z}_{\geq \hat{d}(G)}} \exp\left\{ -\frac{\beta}{c}(d - \hat{d}(G)) \right\} (2d + c + 1) R S_f(\mathcal{G}_{2D})$$

Blocki et al. [BBDS12] proves that a 4-Lipschitz distance estimator exists. Using this distance estimator, we can finally create our vertex-differentially private algorithm:

Theorem 4.6 ([BBDS12]). *Let $f : \mathcal{G} \to \mathbb{R}$. The algorithm*

$$\mathcal{A}_f(G) = f(\mu_E(G)) + \text{Lap}\left(\frac{2 \cdot S_{f,\varepsilon/2\ln(1/\delta)}(G)}{\varepsilon} \right)$$

is (ε, δ)-vertex differentially private.

4.2 Lipschitz extensions for higher-dimensional vertex privacy

In this section, we look at a different approach for designing vertex differentially private algorithms, via a concept called *Lipschitz extensions* [RS15].

The idea behind Lipschitz extensions is not too far from the idea of projections. We will again focus on a specific subset of graphs, specifically graphs of bounded degree. However, the concept of Lipschitz extensions are applicable to queries taking values in $\mathbb{R}^d$ for $d \geq 2$ (restricted sensitivity is only applicable to scalar-valued queries). However, we will see that not all queries in $\mathbb{R}^d$ admit Lipschitz extensions [RS15]. In contrast, Lipschitz extensions always exist for queries in $\mathbb{R}$ [McS34].

4.2.1 Background

We first define Lipschitz extensions:

Definition 4.7 ([RS15])**.** *Let $X \subset X^*$ be two sets. Let $f : X \to Y$ be a c-Lipschitz function. A Lipschitz extension $f^* : X^* \to Y$ of f with stretch s is a function that satisfies the following conditions:*

- *For all $x \in X$, $f(x) = f^*(x)$.*
- *f^* is $s \cdot c$-Lipschitz.*

We can see how these would be useful for designing differentially private algorithms; their motivation is taking advantage of low global sensitivity within a particular subset of our domain X^*. Lipschitz extensions with stretch s scale the sensitivity by a factor of s, so finding constant stretch extensions is extremely useful since, when used in privacy-preserving mechanisms, they keep the privacy parameter small [RS15].

As a concrete example, suppose we want to compute the number of edges $f_E(G)$ in a graph G. The global sensitivity of this function is $O(n)$ in an arbitrary graph, since the removal of a vertex will remove at most $n - 1$ edges [KNRS13]. On the other hand, when considering degree-bounded graphs, the global sensitivity is only $O(D)$. With a constant stretch Lipschitz extension f_E^* of f_E and the Laplace mechanism, we now have an algorithm that releases the number of edges with noise proportional to $O(D)/\varepsilon$ instead of $O(n)/\varepsilon$.

It's important to note in the above example that the result is not necessarily accurate on high degree graphs; all this guarantees us is that on sparse graphs (which are more common) our result is released with much smaller noise. Furthermore, this algorithm is still ε-differentially private, so we don't compromise any degree of privacy. A common workaround for this is to apply the standard Laplace mechanism for graphs that yield a high-valued statistic (measured with the L_1-norm) and choosing a sufficient threshold to run this on in order to maintain privacy [KNRS13].

Thus, the remainder of this section will be focused on designing such extensions.

4.2.2 Lipschitz extensions for scalar functions

If our query function is a scalar function, it turns out that a Lipschitz extension of stretch 1 always exists. This theorem is due to McShane [McS34].

Theorem 4.8 ([McS34])**.** *Let (X^*, d) be a metric space. Given a c-Lipschitz function $f : X \to Y$, the function $f^* : X^* \to Y$ such that*

$$f^*(x^*) = \sup_{x \in X}(f(x) - c \cdot d(x, x^*))$$

is a stretch 1 Lipschitz extension of f.

Proof. Due to [McS34]. If $x^* \in X$, then by Lipschitz continuity, we have that

$$f(x) - f(x^*) \leq |f(x) - f(x^*)| \leq c \cdot d(x, x^*)$$
$$f(x) - c \cdot d(x, x^*) \leq f(x^*)$$

for all $x \in X$, and note that equality is achieved when $x = x^*$. So,

$$f^*(x^*) = \sup_{x \in X}(f(x) - c \cdot d(x, x^*)) = f(x^*)$$

This proves the first property of Lipschitz extensions. Now we prove that

$$|f^*(x_2^*) - f^*(x_1^*)| \leq c \cdot d(x_1^*, x_2^*)$$

for all $x_1^*, x_2^* \in X^*$. WLOG suppose that $f^*(x_1^*) \leq f^*(x_2^*)$. Then,

$$\begin{aligned}
f^*(x_2^*) - f^*(x_1^*) &= \sup_{x \in X}(f(x) - c \cdot d(x, x_2^*)) - \sup_{x \in X}(f(x) - c \cdot d(x, x_1^*)) \\
&\leq \sup_{x \in X}[(f(x) - c \cdot d(x, x_2^*)) - (f(x) - c \cdot d(x, x_1^*))] \\
&\qquad\qquad \text{(using } \sup f(x) + \sup g(x) \geq \sup(f(x) + g(x))\text{)} \\
&= \sup_{x \in X}[c(d(x, x_1^*) - d(x, x_2^*))] \\
&\leq c \cdot d(x_1^*, x_2^*) \qquad \text{(by the triangle inequality)}
\end{aligned}$$

proving the claim from [McS34]. □

This gives a nice analytic form for the Lipschitz extension; however, in practice, naïvely computing this exact expression is not necessarily efficient, as we need to iterate through all elements of X in order to compute the value of f^*. In practice, this form is not used often and query-specific constructions are used, but this at least shows that it is possible to have a stretch 1 function for any scalar function regardless of computation bounds.

On the contrary, Raskhodnikova and Smith [RS15] prove an impossibility result regarding the existence of higher-dimensional Lipschitz extensions:

Lemma 4.9 ([RS15])**.** *$\exists c$ such that for all $p \geq 3$, there exist functions $f : \mathcal{G}_D \to \mathbb{R}^p$ (with the L_1-norm) that do not have stretch-$O(1)$ Lipschitz extensions to $\mathcal{G}$.*

4.2.3 Lipschitz extension for degree list

We show a Lipschitz extension for the degree list of a graph, based on [RS15]. The degree list of a graph $G = (V, E)$ is

$$\text{deg-list}(G) = \mathit{sort}([\deg(v)]_{i=1}^{|V|}) \in \mathbb{R}^*$$

Without loss of generality, we will assume the degree list to be sorted. Note that the sensitivity of the degree list with the vertex neighbor metric is $2D$, since removing a vertex

from will result in one term in the degree list going to zero, and all other non-zero terms decreasing by at most one [RS15]. Sorting will also not increase the L_1-norm between the degree lists; for a proof of this see Lemma A.1.

We will show a function that has stretch $\frac{3}{2}$, i.e. the resulting sorted degree lists from neighboring graphs have L_1 distance at most $3D$ [RS15].

To do so, Raskhodnikova and Smith [RS15] and Kasiviswanathan et al. [KNRS13] make use of a so-called *flow graph* of our graph:

Definition 4.10 ([RS15])**.** *The flow graph $FG(G)$ of a graph $G = (V, E)$ is defined as follows: let $V' = V^\ell \cup V^r \cup \{s, t\}$, where V^ℓ and V^r are copies of V, and let E' be the following set:*

$$E' = \{(s, v^\ell) \mid v^\ell \in V^\ell\} \cup \{(v_i^\ell, v_j^r) \mid (v_i, v_j) \in E\} \cup \{(v^r, t) \mid v^r \in V^r$$

Let all edges going out of s have capacity D as well as all edges going into t. Let all other edges have unit capacity. V' and E' form a directed graph denoting the flow graph $FG(G)$ of G.

We can use this flow graph in order to construct several Lipschitz extensions. Kasiviswanathan et al. [KNRS13] use this flow graph to construct an extension of the edge count in a graph.

In this paper, we will focus on the degree list. For the degree list, in addition to maximizing the s-t flow in the flow graph, we will supplement this with a minimization problem in order to obtain a unique max-flow [RS15]. We define our Lipschitz extension f^* as follows [RS15]:

1. Fix a canonical ordering of V, and construct the flow graph of G, $FG(G)$.
2. Let f_s be a vector of the flows going out of s (so $f_s \in \mathbb{R}^n$) and let f_t be a vector of the flows going into t. Let $f_{st} = f_s \circ f_t$ be their concatenation.
3. Define our objective function Φ as:

   $$\Phi(f) = \|f_{st} - \vec{D}_{2n}\|_2^2$$

   where $\vec{D}_{2n}$ is a vector of entries D of length $2n$.
4. Find a flow $\hat{f}$ in $FG(G)$ that minimizes the value of $\Phi(\hat{f}_{st})$.
5. Output $\text{sort}(\hat{f}_s)$, the sorted list of flows going out of s.

One property of Φ we will be using often is that it is strictly convex [RS15]. The minimization of Φ allows us to deterministically select a *unique* max-flow, since Φ is strictly convex and must have a single global minimum over the set of flows [RS15]. It turns out that minimizing Φ actually leads to a max-flow over the flow graph:

Theorem 4.11 ([RS15])**.** *If f is a flow that optimizes Φ, then f is a max-flow in $FG(G)$.*

Proof. Due to Raskhodnikova and Smith [RS15]. If f was not a max-flow, then in the residual graph of $FG(G)$, there exists some $s \rightsquigarrow t$ path. This path would use two edges (s, w) and (x, t) that are the only edges contributing to the vector f_{st}. By augmenting the flow f to f', we get that $\Phi(f') < \Phi(f)$ since Φ is stricty decreasing in f_{st} (as the entries of f_{st} must be nonnegative and upper-bounded by D). But this contradicts the fact that f achieves the optimal value of Φ. This proof [RS15] thus proves the claim. □

Now we are ready to prove that this extension is actually a Lipschitz extension:

Theorem 4.12 ([RS15])**.** *$f^* : \mathcal{G} \to \mathbb{R}^*$ is a Lipschitz extension of $f_{\text{deglist}} : \mathcal{G}_D \to \mathbb{R}^*$ of stretch $\frac{3}{2}$, i.e.*

- *$f^*(G) = f_{\text{deglist}}(G)$ for all graphs $G \in \mathcal{G}_D$.*
- *f^* is $3D$-Lipschitz:*

$$\|f^*(G_2) - f^*(G_1)\|_1 \leq 3D \cdot d_V(G_1, G_2) \quad \forall G_1, G_2 \in \mathcal{G}$$

Proof. Due to Raskhodnikova and Smith [RS15]. Consider two neighboring graphs $G_1 \sim_v G_2$, where $V(G_2) = \{v'\} \sqcup V(G_1)$. Let f_1 and f_2 be the max-flows in $FG(G_1)$ and $FG(G_2)$, respectively, that optimize Φ. Then the flow $f = f_2 - f_1$ is feasible in the residual graph of $FG(G_2)$ for f_1. Therefore, we have that

$$\|f^*(G_2) - f^*(G_1)\|_1 = \|f_s\|_1$$

so it suffices to show that $\|f_s\| \leq 3D$.

The way we achieve this is via the Flow Decomposition Theorem [Eri19], by decomposing f into a set of $s \rightsquigarrow t$ paths and cycles. We partition this decomposition into three separate flows:

- Ψ^s will contain all paths and cycles of the decomposition that use the edge (s, v') in $FG(G_2)$.
- Ψ^t will contain all paths and cycles that do not use (s, v') but use (v', t) in $FG(G_2)$.
- Ψ^0 will contain all other paths and cycles.

We prove bounds on each of these flows, which will suffice to proving our claim.

First, we claim that $\|\Psi^s_s\|_1 \leq 2D$. Note that each path in Ψ^s will contribute the value of its own flow to Ψ^s and each cycle starting from s will contribute at most twice its flow (for the edge going out of s and the one going into s). Since the total flow going through (s, v') is at most D by construction, the claim holds.

Next, we claim that $\|\Psi^t_s\|_1 \leq D$. In the same logic as above, each path to t will contribute its own value. However, each cycle contributes 0 to Ψ^t_s, since the cycle would have to go through

s, and the $t \rightsquigarrow s$ portion of the cycle would allow us to augment the flow Ψ^t, contradicting the optimality of f_2. Thus, by similar logic, $\|\Psi^t_s\|_1 \leq D$.

Finally, we claim that $\|\Psi^0_s\|_1 = 0$. Suppose by way of contradiction that $\|\Psi^0_s\|_1 > 0$. Let $\mathcal{F}_1$ be a set containing all feasible flows in $FG(G_1)$, endowed with the L_2-norm, which includes f_1. $\mathcal{F}_1$ is a convex set, since any convex combination of feasible flows is also a feasible flow. Note that $f_1 + \Psi_0 \in \mathcal{F}_1$ since Ψ^0 has no paths that use v'. Furthermore, since f_1 minimizes Φ, f_1 is the closest point in $\mathcal{F}_1$ to $\vec{D}_{2n}$ (since it minimizes the L_2-norm to $\vec{D}_{2n}$). We make use of Theorem A.2 to conclude that

$$\begin{aligned}\langle (f_1 + \Psi^0) - f_1, \vec{D}_{2n} - f_1 \rangle &\leq 0 \\ \langle \Psi^0, \vec{D}_{2n} - f_1 \rangle &\leq 0\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. Similarly, letting $\mathcal{F}_2$ be the set of feasible flows in $FG(G_2)$, and using similar logic,

$$\begin{aligned}\langle (f_2 - \Psi^0) - f_2, \vec{D}_{2n} - f_2 \rangle &\leq 0 \\ -\langle \Psi^0, \vec{D}_{2n} - f_2 \rangle &\leq 0\end{aligned}$$

Adding these two inequalities gives us

$$\begin{aligned}\langle \Psi^0, \vec{D}_{2n} - f_1 \rangle - \langle \Psi^0, \vec{D}_{2n} - f_2 \rangle &\leq 0 \\ \langle \Psi^0, f_2 - f_1 \rangle &\leq 0 \\ \langle \Psi^0, \Psi^s + \Psi^t + \Psi^0 \rangle &\leq 0 \\ \langle \Psi^0, \Psi^s + \Psi^t \rangle &\leq 0\end{aligned}$$

However, the only way for this to happen is if $\Psi^0 = \vec{0}$, since Ψ^0 and $\Psi^s + \Psi^t$ cannot have opposite signs in any entry since they are both subflows of f. Therefore, the claim holds.

Putting these three claims together, we have that $\|f_s\| \leq 3D$ [RS15]. □

This gives a full Lipschitz extension of the degree list. We note that this can be efficiently computed via an efficient linear program solver, since the problem of minimizing Φ is simply a convex optimization problem over the set of feasible flows for a flow graph with polynomially many constraints [RS15].

Rashkhodnikova and Smith [RS15] use this Lipschitz extension to devise a few more things:

- This extension is employed to create a *degree distribution* Lipschitz extension, a histogram of the degrees of every vertex. This is achieved by defining a stretch 1 function that takes as input the degree list and outputs the cumulative degree histogram (CDH), counts of vertices of degree greater than a certain amount [RS15]. Then, transforming this cumulative histogram to a degree histogram has stretch 2 (since to obtain the degree histogram we subtract consecutive elements of the CDH). Thus, this results in a stretch 3 Lipschitz extension for the degree histogram [RS15].

- One lingering question we have left unanswered is how to select an appropriate value of D. Rashkhodnikova and Smith [RS15] create a modified version of the exponential mechanism for Lipschitz extensions. The current issue with the exponential mechanism is that q-functions with large sensitivities are not penalized enough in order to output with small error, since, if we let $\varepsilon' = 2\varepsilon\Delta(q)$ in the exponential mechanism, the error term in Theorem 2.10 relies on the maximum sensitivity of q over all values of r. This new exponential mechanism releases an output whose error is proportional to the sensitivity of the optimal r [RS15].

5 Conclusion

In this paper, we review the multitude of theory developed for designing private algorithms on graphical data. We see that in the design of the techniques described in this paper, the high sensitivity and sparsity of most graph enable us to create more useful algorithms for operating in this domain. These properties drove the creation of two key ideas: adding noise reduction for high-sensitivity queries as well as designing functional extensions of well-behaved subsets. In particular, these techniques we discussed are not only applicable to graphs, but for any generic collection of datasets. Therefore, these techniques are not solely tailored towards usage in graph problems; they comprise an artifact of the investigation of fundamental issues in conventional private graph algorithm design.

We remark on some potential areas of future work based on the results presented above. Most of the above constructions were not very efficient compared to their non-private counterparts. A great deal of research could be spent to come up with viable algorithms that push the efficiency further. Furthermore, frameworks such as the Lipschitz extension frameworks are not very easily generalizable; as we saw, not all functions yield a Lipschitz extension in higher dimensions [RS15]. The ideal scenario would be to have nontrivial necessary and sufficient conditions for when such an extension exists, as well as a more generalizable framework for constructing such extensions when those conditions are satisfied.

References

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. *CoRR*, abs/1208.4586, 2012.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[DRS19] Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy. *CoRR*, abs/1905.02383, 2019.

[Dwo06] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, July 2006.

[EK72] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[Eri19] Jeff Erickson. *Algorithms*. 2019.

[FF09] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. In *Classic papers in combinatorics*, pages 243–248. Springer, 2009.

[GRU11] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. *CoRR*, abs/1107.3731, 2011.

[HLMJ09] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

[Kar93] David Karger. Global min-cuts in *rnc* and other ramifications of a simple mincut algorithm. pages 21–30, 01 1993.

[KNRS13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer, 2013.

[Kru56] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[KRWY15] Michael J. Kearns, Aaron Roth, Zhiwei Steven Wu, and Grigory Yaroslavtsev. Privacy for the protected (only). *CoRR*, abs/1506.00242, 2015.

[Lap] Issie Lapowsky. How cambridge analytica sparked the great privacy awakening.

[Lav19] Mikhail Lavrov. Chapter 5, lecture 1: The obtuse angle criterion, March 2019.

[McS34] Edward James McShane. Extension of range of functions. *Bulletin of the American Mathematical Society*, 40(12):837–842, 1934.

[Mir17] Ilya Mironov. Renyi differential privacy. *CoRR*, abs/1702.07476, 2017.

[MM09] Ilya Mironov and Frank McSherry. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 627–636. Association for Computing Machinery, Inc., June 2009.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, page 94–103, USA, 2007. IEEE Computer Society.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[NS06] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[Pin18] Rafael Pinot. Minimum spanning tree release under differential privacy constraints. *CoRR*, abs/1801.06423, 2018.

[Pri57] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[PSU88] Anthony L Peressini, Francis E Sullivan, and J Jerry Uhl. *The mathematics of nonlinear programming*. Springer-Verlag New York, 1988.

[Rot20] Aaron Roth. Email correspondence, 2020.

[RS15] Sofya Raskhodnikova and Adam D. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *CoRR*, abs/1504.07912, 2015.

[Sea15] Adam Sealfon. Shortest paths and distances with differential privacy. *CoRR*, abs/1511.04631, 2015.

[XCT14] Qian Xiao, Rui Chen, and Kian-Lee 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, page 911–920, New York, NY, USA, 2014. Association for Computing Machinery.

A Appendix

Invariance to sorting of the degree list

Lemma A.1. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ be two sequences of real numbers. Then $\|sort(\mathbf{b}) - sort(\mathbf{a})\|_1 \leq \|\mathbf{b} - \mathbf{a}\|_1$.*

Proof. Define a *parallel pair* of vectors $\mathbf{a}', \mathbf{b}'$ with respect to $\mathbf{a}, \mathbf{b}$ if there exists some permutation σ such that $\sigma(\mathbf{a}) = \mathbf{a}'$ and $\sigma(\mathbf{b}) = \mathbf{b}'$.

Note that permuting two vectors under the same permutation does not change the L_1 norm between them. Thus, we can prove the stronger claim: any parallel pair $\mathbf{x}, \mathbf{y}$ with respect to $sort(\mathbf{a}), sort(\mathbf{b})$ has minimum L_1 distance, i.e. for any permutations σ_1, σ_2,

$$\|\mathbf{y} - \mathbf{x}\|_1 \leq \|\sigma_2(\mathbf{b}) - \sigma_1(\mathbf{a})\|_1$$

Now, note that $\mathbf{x}, \mathbf{y}$ is a parallel pair to $sort(\mathbf{a}), sort(\mathbf{b})$ if and only if for all i, j, we have that either $x_i \leq x_j$ and $y_i \leq y_j$, or $x_i \geq x_j$ and $y_i \geq y_j$. It just suffices to find a parallel pair that has minimum distance, since all parallel pairs have the same distance.

Suppose by way of contradiction that no parallel pair $\mathbf{x}, \mathbf{y}$ achieves the minimum; then consider an arbitrary non-parallel pair $\mathbf{x}', \mathbf{y}'$ to $sort(\mathbf{a}), sort(\mathbf{b})$ that does achieve the minimum L_1 distance out of all permutations σ_1, σ_2; then for some i, j, the pairs (x'_i, x'_j) and (y'_i, y'_j) have opposite orderings. WLOG, we assume that $x'_i < x'_j$ and $y'_i > y'_j$ (we will consider the case when one pair is equal later). One can verify the different cases of the relative orderings of $\{x'_i, x'_j, y'_i, y'_j\}$ to show that by swapping y'_i and y'_j, the L_1 distance can only decrease. The resulting vectors cannot parallel to $sort(\mathbf{a}), sort(\mathbf{b})$ by our initial assumption. If the L_1 distance decreases, this contradicts the minimality of $\mathbf{x}', \mathbf{y}'$. Thus, the L_1 distance must stay the same. This process can be iterated until such i, j do not exist in the vectors, but this would contradict the fact that parallel pairs cannot achieve the minimum L_1 distance.

Therefore, every parallel pair achieves the minimum, and in particular, $sort(\mathbf{a}), sort(\mathbf{b})$ achieves the minimum. □

Notes on convex optimization

We prove a result about convex optimization that we make use of in Theorem 4.12.

Theorem A.2 (Obtuse angle theorem [PSU88])**.** *Let $\mathcal{C} \subseteq \mathbb{R}^k$ be a convex set. Let $\mathbf{y} \notin \mathcal{C}$ be some point outside of our convex set. Then the point $\hat{\mathbf{x}} \in \mathcal{C}$ is the closest point to $\mathbf{y}$ in $\mathcal{C}$ if and only if*

$$\langle \mathbf{y} - \hat{\mathbf{x}}, \mathbf{x} - \hat{\mathbf{x}} \rangle \leq 0$$

for all $\mathbf{x} \in \mathcal{C}$.

Proof. The proof is from [Lav19]. Fix an arbitrary $\mathbf{x} \in \mathcal{C}$, and consider the set of points on the line between $\mathbf{x}$ and $\hat{\mathbf{x}}$, parameterized by t. Define ϕ to be the squared distance between a point on this line and $\mathbf{y}$:

$$\phi(t) = \|t\mathbf{x} + (1-t)\hat{\mathbf{x}} - \mathbf{y}\|^2 \qquad t \in [0,1]$$

Notice that

$$\begin{aligned}
\phi(t) &= \|t\mathbf{x} + (1-t)\hat{\mathbf{x}} - \mathbf{y}\|^2 \\
&= \|(\hat{\mathbf{x}} - \mathbf{y}) + t(\mathbf{x} - \hat{\mathbf{x}})\|^2 \\
&= \|\hat{\mathbf{x}} - \mathbf{y}\|^2 - 2t\langle \mathbf{y} - \hat{\mathbf{x}}, \mathbf{x} - \hat{\mathbf{x}}\rangle + t^2\|\mathbf{x} - \hat{\mathbf{x}}\|^2 \\
\phi'(t) &= 2t\|\mathbf{x} - \hat{\mathbf{x}}\|^2 - 2\langle \mathbf{y} - \hat{\mathbf{x}}, \mathbf{x} - \hat{\mathbf{x}}\rangle \\
&= 2t\|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \phi'(0)
\end{aligned}$$

Now note the following:

- If $\phi'(0) \geq 0$, then $\phi'(t) \geq 0$ for all t, and so ϕ is increasing w.r.t. t and thus $t = 0$ achieves the minimum.
- If $\phi'(0) < 0$, then for some small value of t, $\phi'(t') < 0$ for all $t' \in [0, t]$, and thus $\phi(0) > \phi(t)$, and thus $t = 0$ does not yield the minimum.

From this, we conclude that $\hat{\mathbf{x}} \in \mathcal{C}$ is the closest point to $\mathbf{y}$ if and only if $\phi'(0) \geq 0$, or equivalently, $\langle \mathbf{y} - \hat{\mathbf{x}}, \mathbf{x} - \hat{\mathbf{x}}\rangle \leq 0$, concluding the proof from [Lav19]. □