Iterative Constructions and Private Data Release*

Anupam Gupta[1,**], Aaron Roth[2,***], and Jonathan Ullman[3,†]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
[2] Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA 19104
[3] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA

Abstract. In this paper we study the problem of approximately releasing the *cut function* of a graph while preserving differential privacy, and give new algorithms (and new analyses of existing algorithms) in both the interactive and non-interactive settings.

Our algorithms in the interactive setting are achieved by revisiting the problem of releasing differentially private, approximate answers to a large number of queries on a database. We show that several algorithms for this problem fall into the same basic framework, and are based on the existence of objects which we call *iterative database construction* algorithms. We give a new generic framework in which new (efficient) IDC algorithms give rise to new (efficient) interactive private query release mechanisms. Our modular analysis simplifies and tightens the analysis of previous algorithms, leading to improved bounds. We then give a new IDC algorithm (and therefore a new private, interactive query release mechanism) based on the Frieze/Kannan low-rank matrix decomposition. This new release mechanism gives an improvement on prior work in a range of parameters where the size of the database is comparable to the size of the data universe (such as releasing all cut queries on dense graphs).

We also give a non-interactive algorithm for efficiently releasing private *synthetic data* for graph cuts with error $O(|V|^{1.5})$. Our algorithm is based on randomized response and a non-private implementation of the SDP-based, constant-factor approximation algorithm for cut-norm due to Alon and Naor. Finally, we give a reduction based on the IDC framework showing that an efficient, private algorithm for computing sufficiently accurate rank-1 matrix approximations would lead to an improved efficient algorithm for releasing private synthetic data for graph cuts. We leave finding such an algorithm as our main open problem.

1 Introduction

Consider a graph representing the online communications between a set of individuals; each vertex represents a user, and an edge between two users indicates that they have corresponded by email. It might be useful to allow data analysts to mine this graph for statistical information. However, the graph is also composed of sensitive information,

* A full version appears at `http://arxiv.org/abs/1107.3731`

** Research was partly supported by NSF awards CCF-0964474 and CCF-1016799.

*** Work done at Microsoft Research, New England. Email: `aaroth@cis.upenn.edu`
† Supported by NSF grant CNS-0831289. Email: `jullman@seas.harvard.edu`

R. Cramer (Ed.): TCC 2012, LNCS 7194, pp. 339–356, 2012.

and we cannot release information that reveals much about the existence of specific edges. Thus we would like a way to analyze the structure of this graph while protecting the privacy of individual edges. Specifically we would like to guarantee *differential privacy* [7] (defined in Section 2), which, roughly, requires that our algorithms be randomized, and induce nearly the same distribution over outcomes when given two data sets (e.g. graphs) which differ in only a single point (e.g. an edge).

Table 1. Comparison of accuracy bounds for linear queries. The bounds in the first column are prior to this work, the second column are what we achieve in this work, and the last column are the new bounds instantiated for releasing all cut queries. The bounds listed here are approximate and hide the dependence on certain parameters, such as δ and β. n denotes database size, k denotes the total number of queries answered, and $\mathcal{X}$ represents the data universe. For a graph $G = (V, E)$, $n = n_2 = |E|$, $|\mathcal{X}| = \binom{|V|}{2}$, and for all cut queries, $k = 2^{2|V|}$. Previous efficient results do not achieve non-trivial ($\leq |E|$) error, while all of the new bounds do for sufficiently dense graphs.

	Previous Bounds	This Paper	
		General Bounds	All Cut Queries
Median Mech.[a] [19]	$\frac{n^{2/3}(\log k)(\log \lvert\mathcal{X}\rvert)^{1/3}}{\epsilon^{1/3}}$	$\frac{n^{1/2}(\log k)^{3/4}(\log \lvert\mathcal{X}\rvert)^{1/4}}{\epsilon^{1/2}}$	$\frac{\lvert E\rvert^{1/2}\lvert V\rvert^{3/4}(\log \lvert V\rvert)^{1/4}}{\epsilon^{1/2}}$
Online MW [15]	$\frac{n^{1/2}(\log k)(\log \lvert\mathcal{X}\rvert)^{1/4}}{\epsilon}$	$\frac{n^{1/2}(\log k)^{1/2}(\log \lvert\mathcal{X}\rvert)^{1/4}}{\epsilon^{1/2}}$	$\frac{\lvert E\rvert^{1/2}\lvert V\rvert^{1/2}(\log \lvert V\rvert)^{1/4}}{\epsilon^{1/2}}$
Frieze/Kannan IDC	New in this paper	$\frac{n_2^{1/4}(\log k)^{1/2}\lvert\mathcal{X}\rvert^{1/4}}{\epsilon^{1/2}}$ [b]	$\frac{\lvert E\rvert^{1/4}\lvert V\rvert}{\epsilon^{1/2}}$
K-Norm Mech.[16]	$\frac{\sqrt{k}}{\epsilon}\left(\log\left(\frac{\lvert\mathcal{X}\rvert}{k}\right)\right)^{1/2}$ [c]	Not in IDC Framework	Not Applicable

[a] The bounds listed here are for linear queries. The Median Mechanism more generally works for any set of low sensitivity queries $\mathcal{Q}$ that have an α-net of size $N_\alpha(\mathcal{Q})$. We improve the bound from the solution to $\alpha = \frac{\log(N_\alpha(\mathcal{Q}))\log^2 k}{\epsilon}$ to the solution to $\alpha = \frac{\sqrt{\log N_\alpha(\mathcal{Q})\log k}}{\epsilon}$.

[b] Here we use $n_2 = \|\mathcal{D}\|_2^2$, in contrast to other known IDCs, whose error is in terms of $n = \|\mathcal{D}\|_1$. Note that $n \leq n_2 \leq n^2$.

[c] For $k \leq |\mathcal{X}|/2$. This is an approximate bound on *average* per-query error. All other algorithms listed bound worst-case per-query error.

One natural objective is to provide private access to the *cut function* of this graph. That is, to provide a privacy preserving way for a data analyst to specify any two (of the exponentially many) subsets of individuals, and to discover (up to some error) the number of email correspondences that have passed between these two groups. There are two ways we might try to achieve this goal: We could give an *interactive* solution where we give the analyst private oracle access to the cut function. Here the user can write down any sequence of cut queries and the oracle will respond with private, approximate answers. We may also try for a stronger, *non-interactive* solution, in which we release a private *synthetic dataset*; a new, private graph that approximately preserves the cut function of the original graph.

The case of answering cut queries on a graph is just one instance of the more general problem of query release for exponentially sized families of *linear queries* on a data set. Although this problem has been extensively studied in the differential privacy literature, we observe that no previously known efficient solution is suitable for the case of releasing all cut queries on graphs. In the setting of cut queries on a graph, we use "efficient solution" roughly to mean one in which each query is answered in time $\mathrm{poly}(|V|)$, in the interactive setting, or one in which the whole construction runs in time $\mathrm{poly}(|V|)$, in the non-interactive setting. In this paper we provide both efficient interactive and non-interactive solutions for this problem.

We give a generic framework that converts objects we call *iterative database construction (IDC)* algorithms into private query release mechanisms in both the interactive and non-interactive settings. This framework generalizes the median mechanism [19], the online multiplicative weights mechanism [15], and the offline multiplicative weights mechanism [12, 14]. Our framework gives a simple, modular analysis of all of these mechanisms, which lead to tighter bounds in the interactive setting than those given in [19] and [15]. These improved bounds are crucial to our objective of giving non-trivial approximations to all possible cut queries. We also instantiate this framework with a new IDC algorithm for arbitrary linear queries that is based on the Frieze/Kannan low-rank matrix decomposition [10] and is tailored to releasing cut queries. This algorithm leads to a new online query release mechanism for linear queries that gives a better approximation in settings (such as we would encounter trying to answer all cut queries on a dense graph) where the database size is comparable to the size of the data universe. We summarize our bounds in Table 1.

We also give a new algorithm (building on techniques for constructing private synthetic data in [2, 8]) in the non-interactive setting that efficiently generates private synthetic graphs that approximately preserve the cut function. Finally, we use our IDC framework to show that an efficient, private algorithm for privately computing good rank-1 approximations to matrices would automatically yield efficient private algorithms for releasing synthetic graphs with improved approximation guarantees.

1.1 Our Results and Techniques

Our main conceptual contribution is to define the abstraction of *iterative database construction (IDC)* algorithms (Section 3) and to show that an efficient IDC for any class of queries $\mathcal{Q}$ automatically yields an efficient private data release mechanism for $\mathcal{Q}$ in both the interactive and non-interactive settings. Informally, IDCs construct a data structure that can be used to answer all the queries in $\mathcal{Q}$ by iteratively improving a hypothesis data structure. Moreover, they update the hypothesis when given a query witnessing a significant difference between the hypothesis data structure and the underlying database.

In hindsight, this framework generalizes the median mechanism [19] and the online multiplicative weights mechanism [15]. It also generalizes the offline multiplicative weights mechanism [12, 14]. All of these mechanisms can be seen to use IDCs of the sort we define in this work. (In Appendix A we show how these algorithms fall into the IDC framework.)

Our generalization and abstraction also allows for a simple, modular analysis of mechanisms based on IDCs. Using this analysis, we are able to show improved bounds on the accuracy of both the median mechanism and multiplicative weights mechanism. These improved bounds are significant in our application of using an interactive mechanism to release a large number of cut queries and crucial if we want to answer all cut queries. When answering all cut queries, the previous bounds would not guarantee error that is $\leq |E|$, meaning that the error may be larger than the largest cut in the graph. Of course, we can privately guarantee error $\leq |E|$ simply by releasing the answer 0 for every cut query. Our new analysis shows that these mechanisms are capable of answering all $2^{2|V|}$ cut queries with error $o(|E|)$ on sufficiently dense graphs; e.g., multiplicative weights gives sublinear error for graphs with $|E| = \omega(|V|\sqrt{\log |V|})$.

Although it may seem unrealistic to answer all cut queries using an interactive mechanism, our new analysis allows us to give a best-of-both-worlds guarantee that we can answer each query efficiently with non-trivial accuracy without ever having to "shut off" the algorithm for answering too many queries. In practice it may be preferable to limit the number of queries the interactive mechanism will have to answer, in order to improve the accuracy of the responses. In this case our new bounds still offer significant improvements in accuracy.

We also define a new IDC based on the Frieze/Kannan low-rank matrix decomposition [10], which yields a private interactive mechanism for releasing linear queries. Our new mechanism outperforms previously known techniques when the size of the database is comparable to the size of the data universe, as is the case on a dense graph. The error for the Frieze/Kannan IDC is smaller than that for multiplicative weights for extremely dense graphs, where $|E| = \Omega(|V|^2/\log |V|)$.

We then consider the problem of efficiently releasing private synthetic data for the class of cut queries. We show that a technique based on randomized response efficiently yields a private data structure (but not a synthetic database) capable of answering any cut query on a graph with $|V|$ vertices up to maximum error $O(|V|^{1.5})$. (Note this error is independent of the density of the graph and the Frieze/Kannan and multiplicative weights IDCs introduce smaller error for sparser graphs.) We then show how to use this data structure to efficiently construct a synthetic database with only a small constant factor blowup in our error. Our algorithm is based on a technique for constructing synthetic data in [2, 8]. Their observation is that, for linear queries, the set of accurate synthetic databases is described by a (large) set of linear constraints. In the case of cut queries, we are able to use a constant-factor approximation to the cut-norm due to Alon and Naor [1] as the separation oracle to find a feasible solution (and thus a synthetic database) efficiently. Finally, we show how the existence of an efficient private algorithm for finding good low-rank approximations to matrices would imply the existence of an improved algorithm for privately releasing synthetic data for cut queries, using our IDC framework.

To summarize the results for cut queries: between the multiplicative weights IDC, the Frieze/Kannan IDC, and randomized response, the best mechanism depends on $|E|$. When $|E|$ is below $O(|V|^2/\log |V|)$, the multiplicative weights IDC introduces the least error. For $|E|$ lying between $O(|V|^2/\log |V|)$ and $O(|V|^2)$, the Frieze/Kannan IDC introduces the least error. Both IDC mechanisms have error increasing with $|E|$, finally matching the error for randomized response when $|E| = \Theta(|V|^2)$. When

answering k queries, the error for all three mechanisms depends on $\sqrt{\log k}$, so these thresholds are independent of the number of queries.

1.2 Related Work

Differential privacy, introduced in a series of papers [4, 6, 7] in the last decade, has become a standard solution concept for statistical database privacy. The first mechanism for simultaneously releasing the answers to exponentially large classes of statistical queries was given in [5]. They showed that the existence of small nets for a class of queries $\mathcal{Q}$ automatically yields a (computationally inefficient) non-interactive, private algorithm for releasing answers to all the queries in $\mathcal{Q}$ with low error. Subsequent improvements were given by Dwork et al. [8, 9].

Roth and Roughgarden [19] showed that large classes of queries could also be released with low error in the *interactive* setting, in which queries may arrive online, and the mechanism must provide answers before knowing which queries will arrive in the future. Subsequently, Hardt and Rothblum [15] gave improved bounds for the online query release problem based on the multiplicative weights algorithm. In hindsight, both of these algorithms follow the same basic framework, which is to use an IDC.

Gupta et al. [12] gave a *non-interactive* data release mechanism based on the multiplicative weights algorithm and an arbitrary agnostic learner for a class of queries. An instantiation of this algorithm (the *offline* multiplicative weights algorithm) using the generic agnostic learner of Kasiviswanathan et al. [17] (who use the exponential mechanism of [18]) was implemented and experimentally evaluated on the task of releasing small conjunctions to low error on real data by Hardt, Ligett, and McSherry [14]. This algorithm gives bounds comparable to those given in this paper, but it does not work in the interactive setting, and is not computationally efficient for settings in which the number of queries is exponentially larger than the database size (as is the case with graph cuts). We note in Section 7 that this generic algorithm can also be instantiated with any iterative database construction algorithm.

Hardt and Talwar [16] consider the setting where the number of queries is smaller than the universe size and introduced the K-Norm mechanism. Subsequent improvements were given by [3]. When the number of queries and the database size are comparable to the universe size (i.e. $|\mathcal{Q}| = \Omega(|\mathcal{X}|)$, $n \geq \Omega\left(|\mathcal{X}|/\log|\mathcal{X}|\right)$), the K-Norm mechanism gives *average error* that is smaller than the worst-case error promised by the online multiplicative weights mechanism. In this range of parameters the Frieze/Kannan IDC and the K-Norm mechanism both improve on the online multiplicative weights, and give roughly the same error. However, the Frieze/Kannan IDC has bounded worse-case error, as opposed to average-case error. In general the two mechanisms are incomparable, as the error of the Frieze/Kannan IDC has bounded worse-case error and applies even when $|Q| > |\mathcal{X}|$, but its error has polynomial, rather than logarithmic dependence on $|\mathcal{X}|$.

The Frieze-Kannan low-rank approximation (or the weak regularity lemma) shows that every matrix can be approximated by a sum of a small number of cut matrices [10, 11], and this fact has many important algorithmic applications. We also use the fact that the proof extends to more general settings, as was noted by [20].

2 Preliminaries

In this paper, we study datasets $\mathcal{D}$ that consist of collections of n elements from some universe $\mathcal{X}$. We can also write $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ when it is convenient to represent $\mathcal{D}$ as a histogram over $\mathcal{X}$. We say that two databases $\mathcal{D}$, $\mathcal{D}'$ are adjacent if they differ in only a single element. As histograms, they are adjacent if $\|\mathcal{D} - \mathcal{D}'\|_1 \leq 1$. We will require that our algorithms satisfy *differential privacy*:

Definition 1 (Differential Privacy). *A randomized algorithm $M : \mathbb{N}^{|\mathcal{X}|} \to R$ (for any abstract range R) satisfies (ϵ, δ)-differential privacy if for all adjacent databases $\mathcal{D}$ and $\mathcal{D}'$, and for all events $S \subseteq R$, $\Pr[M(\mathcal{D}) \in S] \leq \exp(\epsilon) \Pr[M(\mathcal{D}') \in S] + \delta$*

We will generally think of ϵ as being a small constant, and δ as being negligibly small – i.e. smaller than any inverse polynomial function of n.

We note that when we will discuss interactive mechanisms, we must view the output of a mechanism as the *transcript* of an interaction between an adaptive adversary who supplies questions about the database based on previous outcomes of the mechanism, and the mechanism itself. For clarity, in this paper we will elide specifics about the model of adaptive private composition. For a detailed treatment of this issue, see [9].

A useful distribution is the *Laplace* distribution.

Definition 2 (The Laplace Distribution). *The Laplace Distribution with mean 0 and scale b is the distribution with probability density function: $\mathrm{Lap}(x|b) = \frac{1}{2b} \exp(-\frac{|x|}{b})$. We will sometimes write Lap(b) to denote the Laplace distribution with scale b, and will sometimes abuse notation and write $\mathrm{Lap}(b)$ simply to denote a random variable $X \sim \mathrm{Lap}(b)$.*

A fundamental result in data privacy is that perturbing low sensitivity queries with Laplace noise preserves $(\epsilon, 0)$-differential privacy.

Theorem 1 ([7]). *Suppose $Q : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is a function such that for all adjacent databases $\mathcal{D}$ and $\mathcal{D}'$, $\|Q(\mathcal{D}) - Q(\mathcal{D}')\|_1 \leq 1$. Then the procedure which on input $\mathcal{D}$ releases $Q(\mathcal{D}) + (X_1, \ldots, X_k)$, where each X_i is an independent draw from a Lap$(1/\epsilon)$ distribution, preserves $(\epsilon, 0)$-differential privacy.*

It will be useful to understand how privacy parameters for individual steps of an algorithm compose into privacy guarantees for the entire algorithm. The following useful theorem is due to Dwork, Rothblum, and Vadhan:

Theorem 2 ([9]). *Let $0 \leq \epsilon \leq 1$ be a parameter. Let P, Q be probability measures supported on a set $\mathcal{S}$ such that $\max_{s \in S} |\log (P(s)/Q(s))| \leq \epsilon$. Then*

$$\mathbb{E}_P \left[\log \left(P(s)/Q(s)\right)\right] \leq 2\epsilon^2.$$

We are interested in privately releasing accurate answers to large collections of queries. Queries are functions $Q : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}$, and we denote collections of queries by $\mathcal{Q}$. We write $k = |\mathcal{Q}|$ to denote the cardinality of the set of queries.

A common type of queries are *linear queries*. A linear query Q has a representation as a vector $[0, 1]^{|\mathcal{X}|}$, and can be evaluated on a database by taking the dot product between the query and the histogram representation of the database: $Q(\mathcal{D}) = Q \cdot \mathcal{D}$.

Definition 3 (Accuracy). *Let $\mathcal{Q}$ be a set of queries. A mechanism $M : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}$ is* (α, β)-accurate for $\mathcal{Q}$ *if there exists a function* $\mathrm{Eval} : \mathcal{Q} \times \mathcal{R} \to \mathbb{R}$ *s.t. for every database $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$, with probability at least $1 - \beta$ over the coins of M, $M(\mathcal{D})$ outputs $r \in \mathcal{R}$ such that $\max_{Q \in \mathcal{Q}} |Q(\mathcal{D}) - \mathrm{Eval}(Q, r)| \leq \alpha$. We will abuse notation and write $Q(r) = \mathrm{Eval}(Q, r)$.*

We say that an algorithm M *releases synthetic data* (as is the case for our new IDC, as well as the multiplicative weights IDC [15]) if $\mathcal{R} = \mathbb{N}^{|\mathcal{X}|}$ In this case, $M(\mathcal{D}) = \widehat{\mathcal{D}} \in \mathbb{N}^{|\mathcal{X}|}$ and $\mathrm{Eval}(\widehat{\mathcal{D}}, Q) = Q(\widehat{\mathcal{D}})$. We say that a synthetic data release algorithm is *efficient* if it runs in time polynomial in $n = \|\mathcal{D}\|_1$, the size of the data set. Note that if $n \ll |\mathcal{X}|$, efficient algorithms will have to input and output concise representations of the dataset (i.e., as collections of items from the universe) instead of using the histogram representation. Nevertheless, it will be convenient to think of datasets as histograms.

We say an algorithm efficiently releases k queries from a class $\mathcal{Q}$ in the interactive setting if on an arbitrary, adaptively chosen stream of queries $Q_1, \ldots, Q_k$, it outputs answers $a_1, \ldots, a_k$. The algorithm must output each a_i after receiving query Q_i but before receiving Q_{i+1}, and is only allowed $\mathrm{poly}(n)$ run time per query. We are typically interested in the case when k can be exponentially large in n. Note that as far as computational efficiency is concerned, releasing synthetic data for a class of queries k is at least as difficult as releasing queries from k in the interactive setting, since we can use the synthetic data to answer queries interactively.

Graphs and Cuts. When we consider datasets that represent graphs $G = (V, E)$, we think of the database as being the edge set $\mathcal{D}_G = E$, and the data-universe being the collection of all possible edges in the complete graph: $|\mathcal{X}| = \binom{|V|}{2}$. That is, we consider the vertex set to be common among all graphs, which differ only in their edge sets. One example we care about is approximating the cut function of a sensitive graph G.

For any real-valued matrix $A \in \mathbb{R}^{m \times m'}$, for $S \subseteq [m]$ and $T \subseteq [m']$, we define $A(S, T) := \sum_{s \in S, t \in T} A_{st}$. The *cut norm of the matrix* A is now defined as $\|A\|_C := \max_{S \subseteq [m], T \subseteq [m']} |A(S, T)|$. A graph G can be represented as its adjacency matrix $A_G \in \{0, 1\}^{|V| \times |V|}$. In this paper, a *cut* in a graph G is defined by any two subsets of vertices $S, T \subseteq V$. We write the value of an S, T cut in G as $G(S, T) := A_G(S, T)$, where A_G is the adjacency matrix of G. Similarly, we extend the definition of *cut norm* to n vertex graphs naturally by defining $\|G\|_C := \|A_G\|_C = \max_{S,T \subseteq V} |G(S, T)|$ and $\|G - H\|_C := \|A_G - A_H\|_C$. The class of cut queries $\mathcal{Q}_{\text{Cut}} = \{Q_{S,T} : S, T \subseteq V\}$, where $Q_{S,T}(G) = A_G(S, T)$. Note that cut queries are an example of a class of linear queries, because we can represent them as a vector in which $Q_{S,T}[i, j] = 1$ if $i \in S, j \in T$ and 0 otherwise, and evaluate $Q_{S,T}(G) = \sum_{i,j \in V} Q_{S,T}[i, j] \cdot A_G[i, j]$.

Note that as linear queries, we can write cut queries as the outer product of two vectors: $Q_{S,T} = \chi_S \cdot \chi_T^T$, where $\chi_S, \chi_T \in \{0, 1\}^{|V|}$ are the characteristic vectors of the sets S and T respectively. Let us define a more general class of *rank-1* queries on graphs to be a subset of all linear queries: $\mathcal{Q}_{r1} = \{Q \in [0, 1]^{|V| \times |V|}$ such that $Q = u \cdot v^T$ for some vectors $u, v \in [0, 1]^{|V|}\}$. Of course the set of rank-1 queries includes the set of cut queries, and any mechanism that is accurate with respect to rank-1 queries is also accurate with respect to cut queries.

Proofs. Because of space constraints, many of the proofs in this paper have been omitted. The interested reader can see full proofs in the full version of this paper: [13].

3 Iterative Database Constructions

In this section we define the abstraction of *iterative database constructions* that includes our new Frieze/Kannan construction and several existing algorithm [19, 15] as a special case. Roughly, each of these mechanisms works by maintaining a sequence of data structures $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots$ that give increasingly good approximations to the input database $\mathcal{D}$ (in a sense that depends on the IDC). Moreover, these mechanisms produce the next data structure in the sequence by considering only one query Q that *distinguishes* the real database in the sense that $Q(\mathcal{D}^{(t)})$ differs significantly from $Q(\mathcal{D})$.

Syntactically, we will consider functions of the form $\mathbf{U} : \mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{R}_{\mathbf{U}}$. The inputs to $\mathbf{U}$ are a data structure in $\mathcal{R}_{\mathbf{U}}$, which represents the current data structure $\mathcal{D}^{(t)}$; a query Q, which represents the distinguishing query, and may be restricted to a certain set $\mathcal{Q}$; and also a real number. which estimates $Q(\mathcal{D})$. Formally, we define a *database update sequence* , to capture the sequence of inputs to $\mathbf{U}$ used to generate the database sequence $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots$.

Definition 4 (Database Update Sequence). *Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database and let $\left\{(\mathcal{D}^{(t)}, Q^{(t)}, \widehat{A}^{(t)})\right\}_{t=1,\ldots,C} \in (\mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R})^C$ be a sequence of tuples. We say the sequence is an $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, C)$*-database update sequence *if it satisfies the following properties:*

1. $\mathcal{D}^{(1)} = \mathcal{D}(\emptyset, \cdot, \cdot)$,
2. *for every* $t = 1, 2, \ldots, C$, $\left|Q^{(t)}(\mathcal{D}) - Q^{(t)}(\mathcal{D}^{(t)})\right| \geq \alpha$,
3. *for every* $t = 1, 2, \ldots, C$, $\left|Q^{(t)}(\mathcal{D}) - \widehat{A}^{(t)}\right| < \alpha$,
4. *and for every* $t = 1, 2, \ldots, C-1$, $\mathcal{D}^{(t+1)} = \mathbf{U}(\mathcal{D}^{(t)}, Q^{(t)}, \widehat{A}^{(t)})$.

We note that for all of the iterative database constructions we consider, the approximate answer $\widehat{A}^{(t)}$ is used only to determine the *sign* of $Q^{(t)}(\mathcal{D}) - Q^{(t)}(\mathcal{D}^{(t)})$, which is the motivation for requiring that $\widehat{A}^{(t)}$ have error smaller than α. The main measure of efficiency we're interested in from an iterative database construction is the maximum number of updates we need to perform before the database $\mathcal{D}^{(t)}$ approximates $\mathcal{D}$ well with respect to the queries in $\mathcal{Q}$. To this end we define an iterative database construction as follows:

Definition 5 (Iterative Database Construction). *Let $\mathbf{U} : \mathcal{R}_{\mathbf{U}} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{R}_{\mathbf{U}}$ be an update rule and let $B : \mathbb{R} \to \mathbb{R}$ be a function. We say $\mathbf{U}$ is a $B(\alpha)$*-iterative database construction for query class $\mathcal{Q}$ *if for every database $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$, every $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, C)$-database update sequence satisfies $C \leq B(\alpha)$.*

Note that, by definition, if $\mathbf{U}$ is a $B(\alpha)$-iterative database construction, then given any maximal $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, C)$-database update sequence, the final database $\mathcal{D}^{(C)}$ must satisfy $\max_{Q \in \mathcal{Q}} \left|Q(\mathcal{D}) - Q(\mathcal{D}^{(C)})\right| \leq \alpha$ or else there would exist another query satisfying property 2 of Definition 4, and thus there would exist a $(\mathbf{U}, \mathcal{D}, \mathcal{Q}, \alpha, C+1)$-database update sequence, contradicting maximality.

4 Query Release from Iterative Database Construction

In this section we describe an interactive algorithm for releasing linear queries using an arbitrary iterative database construction.

Algorithm 1. Online Query Release Mechanism

$\mathcal{M}^{\mathbf{U}}(\mathcal{D}, \epsilon, \delta, \alpha, \beta, k)$**:**

Input: A database $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$, a parameter $\alpha \in \mathbb{R}$, parameters $\epsilon, \delta, \beta \in [0, 1]$, and the number of queries $k \in \mathbb{N}$. Oracle access to $\mathbf{U}$, a $B = B(\alpha)$-iterative database construction for $\mathcal{Q}$.

Parameters:

$$\sigma = \sigma(\alpha) := \frac{1000\sqrt{B(\alpha)} \cdot \log(4/\delta)}{\epsilon} \qquad T = T(\alpha) := 4\sigma(\alpha) \cdot \log(2k/\beta).$$

Set $\mathcal{D}^{(1)} := \mathbf{U}(\emptyset, \cdot, \cdot)$, $C = 0$.

For: $t = 1, 2, \ldots, k$

1. Receive a query $Q^{(t)} \in \mathcal{Q}$ and compute

$$Z^{(t)} \sim \mathrm{Lap}(\sigma) \qquad A^{(t)} = Q^{(t)}(\mathcal{D}) \qquad \widehat{A}^{(t)} = Q^{(t)}(\mathcal{D}) + Z^{(t)} \qquad \Lambda^{(t)} = Q^{(t)}(\mathcal{D}^{(t)})$$

2. **If:** $|\widehat{A}^{(t)} - \Lambda^{(t)}| \leq T$ **then:** output $\Lambda^{(t)}$ and set $\mathcal{D}^{(t+1)} = \mathcal{D}^{(t)}$
   Else: output $\widehat{A}^{(t)}$, set $\mathcal{D}^{(t+1)} = \mathbf{U}\left(\mathcal{D}^{(t)}, Q^{(t)}, \widehat{A}^{(t)}\right)$, and set $C = C + 1$.
3. **If:** $C = B(\alpha)$ **then:** terminate.

4.1 Privacy Analysis

Theorem 3. *Algorithm 1 is (ϵ, δ)-differentially private.*

Proof (Proof Sketch). Our privacy analysis follows the approach of [15] straightforwardly. The details appear in the full version of the paper. Intuitively, we will try to classify the answers to the queries by the amount of "information leaked about the database." This classification will lead to a bound on the total amount of information leaked, and a tighter bound can be deduced using Theorem 2.

At a very high level, the argument can be thought of in two steps. The first is to argue that the noise we add has large enough magnitude that the information leaked in the (small number of) "update rounds" is small. This step is simple and follows from the bound on the number of update rounds and the well-known properties of the Laplace distribution. The second step is to argue that the *location* of the update rounds also leaks little information. This second step is more difficult, and requires reasoning carefully about rounds that are "close to update rounds."

More specifically, though still informally we will consider three possible ranges for the value of the *noise* $Z^{(t)}$ in each round $t = 1, 2, \ldots, k$.. Intuitively the three cases are as follows: 1) The noise is sufficiently small that there would never be an update, even if the input database were exchanged with an adjacent one. Here we argue no information is leaked. 2) The noise is sufficiently large that there would always be an

update, even if the database were exchanged with an adjacent one. In these rounds there is information leaked, but we also increment C, and thus there cannot be too many of these before terminating. 3) The noise is intermediate, such that we do not do an update and increment C, but might if we switched to an adjacent database. In principle there may be as many as k such rounds, however it will turn out with high probability the number of such rounds is not much bigger than B.

We then complete the proof by applying Theorem 2 to bound the *expected* privacy loss over the course of all the rounds, and apply Azuma's inequality to argue that except with probability δ, the total privacy loss does not exceed ϵ.

4.2 Utility Analysis

Theorem 4. *Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database. And $\mathbf{U}$ be a $B(\alpha)$-iterative database construction for query class $\mathcal{Q}$. Then for any $\beta, \epsilon, \delta > 0$, Algorithm 1 is $\left(\frac{5T(\alpha)}{4}, \beta\right)$-accurate for $\mathcal{Q}$, as long as $T(\alpha) \in [4\alpha/3, 2\alpha]$.*

Proof (Proof sketch). Roughly, the argument is as follows: Assume we did not add any noise to the queries. Then we would answer each query with the exactly-correct answer $A^{(t)}$ or with $\Lambda^{(t)}$ so long as $\Lambda^{(t)}$ is sufficiently close to $A^{(t)}$. Essentially, all we do in the proof is show that this intuition remains correct when noise is added.

When adding noise we answer with either $A^{(t)} + Z^{(t)}$ or $\Lambda^{(t)}$, so long as $\Lambda^{(t)}$ is sufficiently close to $A^{(t)} + Z^{(t)}$. It is not hard to argue that $Z^{(t)}$ remains small in every round, and thus the answers in the latter case are not much less accurate than the answers in the former case.

What remains to be shown is that the mechanism does not terminate early due to the condition $C = B$. In order to do this, we show that the sequence of updates forms a database update sequence, and thus cannot be too long if $\mathbf{U}$ is an efficient iterative database construction. In order to do this, we argue that $Z^{(t)}$ is sufficiently small that the condition for performing an update ($|A^{(t)} + Z^{(t)} - \Lambda^{(t)}| \geq T$) is sufficient to ensure that the query is a good distinguisher ($|A^{(t)} - \Lambda^{(t)}| \geq \alpha$).

In order to get the best accuracy parameters, one can just solve for the equation $\alpha = 3T(\alpha)/4$; substituting for $T(\cdot)$, this is the same as solving the following equation for α: $\alpha = \frac{96\sqrt{B(\alpha)}\log(4/\delta)\log(k/\beta)}{\epsilon}$. Using this method we obtain bounds on the error for various IDCs, which are summarized both in Table 1 and in the full version.

5 An Iterative Database Construction Based on Frieze/Kannan

In this section we describe and analyze an iterative database construction based on the Frieze/Kannan "cut decomposition" [10]. Although the style of analysis we use was originally applied specifically to cuts in [10], their argument generalizes to arbitrary linear queries. To our knowledge, such a generalization was first observed in [20].

Note that the sum in Algorithm 2 denotes entrywise vector addition.

Theorem 5. *Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be a dataset. For any $\alpha > 0$, $\mathbf{U}_\alpha^{FK}$ is a $B(\alpha)$-iterative database construction for a class of linear queries $\mathcal{Q}$, where $B(\alpha) = \frac{\|\mathcal{D}\|_2^2|\mathcal{X}|}{\alpha^2}$.*

Algorithm 2. The Frieze/Kannan-based IDC

$\mathbf{U}_\alpha^{FK}(\mathcal{D}, Q, \widehat{A})$**:**
If: $\mathcal{D} = \emptyset$ **then:** output $\mathcal{D}' = \emptyset$
Else if: $Q(\mathcal{D}) - \widehat{A} > 0$ **then:** output $\mathcal{D}' = \mathcal{D} - \frac{\alpha}{|\mathcal{X}|} \cdot Q$
Else if: $Q(\mathcal{D}) - \widehat{A} < 0$ **then:** output $\mathcal{D}' = \mathcal{D} + \frac{\alpha}{|\mathcal{X}|} \cdot Q$

Proof (Proof sketch). Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database and let $\left\{(\mathcal{D}^{(t)}, Q^{(t)}, \widehat{A}^{(t)})\right\}_{t=1,\ldots,C}$ be $(\mathbf{U}_\alpha^{FK}, \mathcal{D}, \mathcal{Q}, \alpha, B)$-database update sequence (Definition 4). We want to show that $C \leq \|\mathcal{D}\|_2^2|\mathcal{X}|/\alpha^2$. Specifically, after $\|\mathcal{D}\|_2^2|\mathcal{X}|/\alpha^2$ invocations of $\mathbf{U}_\alpha^{FK}$, the database $\mathcal{D}^{(\|\mathcal{D}\|_2^2|\mathcal{X}|/\alpha^2)}$ is $(\alpha, \mathcal{Q})$-accurate for $\mathcal{D}$, and thus there cannot be a sequence of longer than $\|\mathcal{D}\|_2^2|\mathcal{X}|/\alpha^2$ queries that satisfy property 2 of Definition 4.

In order to formalize this intuition, we use a potential argument as in [10] to show that for every $t = 1, 2, \ldots, B$, $\mathcal{D}^{(t+1)}$ is significantly closer to $\mathcal{D}$ than $\mathcal{D}^{(t)}$. Specifically, our potential function is the L_2^2 norm of the database $\mathcal{D} - \mathcal{D}^{(t)}$, defined as $\|\mathcal{D}\|_2^2 = \sum_{i \in \mathcal{X}} \mathcal{D}(i)^2$. Observe that $\|\mathcal{D} - \mathcal{D}^{(1)}\|_2^2 = \|\mathcal{D}\|_2^2$, and $\|\mathcal{D}\|_2^2 \geq 0$. Thus it will suffices to show, as we do in the full proof, that in every step, the potential decreases by $\alpha^2/|\mathcal{X}|$.

Corollary 1. *Let* $\gamma = O\left(\epsilon^{-1/2} n_2^{1/4} |\mathcal{X}|^{1/4} \sqrt{\log(k/\beta)}\right)$*. Then Algorithm 1, instantiated with* $\mathbf{U}_\gamma^{FK}$ *is* (ϵ, δ)*-differentially private and an* (α, β)*-accurate interactive release mechanism for query set* $\mathcal{Q}$ *with* $\alpha = O\left(\frac{n_2^{1/4}|\mathcal{X}|^{1/4}\sqrt{\log(k/\beta)\log(1/\delta)}}{\sqrt{\epsilon}}\right)$ *where* $n_2 = \|\mathcal{D}\|_2^2$*. Note that for databases that are subsets of the data universe (rather than multisets),* $n_2 = n$*.*

Remark 1. For the setting in which the database represents a graph and the queryset contains all cut queries, this bounds is $O(|V||E|^{1/4}/\sqrt{\epsilon})$. This improves on the accuracy of the multiplicative weights IDC for dense graphs with $|E| \geq \Omega(|V|^2/\log|V|)$.

6 Results for Synthetic Data

In this section, we consider the more demanding task of efficiently releasing *synthetic data* for the class of cut queries on graphs. Our algorithm is simple, and is based on releasing a noisy histogram. Note that for a graph, $|\mathcal{X}| = \binom{|V|}{2}$, and $\mathcal{D} = E$, so as long as $|E| = \Omega(|V|)$, the universe is at most a polynomial in the database size. (Moreover, it is easy to show that there does not exist any $(\epsilon, 0)$-private mechanism that has error $o(|V|)$, so the only interesting cases are when $|E| = \Omega(|V|)$.)

Consider a database whose elements are drawn from $\mathcal{X}$; we represent this as a vector (histogram) $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$. Let $\widehat{\mathcal{D}} = \mathcal{D} + (Y_1, \ldots, Y_{|\mathcal{X}|})$ be a "noisy" database, where each $Y_i \sim \text{Lap}(1/\varepsilon)$ is an independent draw from the Laplace distribution. Note that by Theorem 1, the procedure which on input $\mathcal{D}$ releases the noisy database $\widehat{\mathcal{D}}$ preserves $(\epsilon, 0)$-differential privacy. This follows because the histogram vector can be viewed as simply the evaluation of the identity query $Q : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{N}^{|\mathcal{X}|}$, which can be easily seen

to be 1-sensitive. At this stage, we could release $\widehat{\mathcal{D}}$ and be satisfied that we have designed a private algorithm. There are two issues: first, we must analyze the utility guarantees that $\widehat{\mathcal{D}}$ has with respect to our query set $\mathcal{Q}$. Second, $\widehat{\mathcal{D}}$ is not quite synthetic data. It will be a vector with possibly negative entries, and so does not represent a histogram. Interpreted as a graph, it will be a weighted graph with negative edge weights. Such an answer may be insufficient for some applications, so in Section 6.1 we show how to convert such an answer into $[0,1]$ weighted graph with similar accuracy guarantees.

The utility guarantee of this procedure over the collections $\mathcal{Q}$ of linear queries is also not difficult; i.e., each query $Q \in \mathcal{Q}$ is a vector in $[0,1]^{|\mathcal{X}|}$, and on any database $\mathcal{D}$ evaluates to $Q(\mathcal{D}) = \langle Q, \mathcal{D}\rangle$.

Lemma 1. *Suppose that $\mathcal{Q} \subseteq [0,1]^{|\mathcal{X}|}$ is some collection of linear queries. For the case $|\mathcal{Q}| \leq (\beta/2)\, 2^{|\mathcal{X}|/6}$, it holds that with probability at least $1-\beta$, for every query $Q \in \mathcal{Q}$, $|Q(\mathcal{D}) - Q(\widehat{\mathcal{D}})| \leq \varepsilon^{-1}\sqrt{6|\mathcal{X}|\log(|\mathcal{Q}|/\beta)}$. For general $\mathcal{Q}$, the error bound is $O(\varepsilon^{-1}\sqrt{|\mathcal{X}|\log(|\mathcal{X}|/\beta)\log(|\mathcal{Q}|/\beta)})$.*

The proof of this lemma uses standard moment-generating function techniques and is deferred to the full version.

In summary, the bounds on the error are $\approx \varepsilon^{-1}\sqrt{|\mathcal{X}|\log|\mathcal{Q}|}$, with some correction terms depending on whether the size of the query set is at most $2^{O(|\mathcal{X}|)}$ or larger.

6.1 Randomized Response and Synthetic Data for Cut Queries

For the case of cuts in graph on a vertex set V, the database is a vector in $\{0,1\}^{\binom{|V|}{2}}$, and the noisy database just adds independent $\mathrm{Lap}(1/\varepsilon)$ noise to each bit value. Since the query set $\mathcal{Q}_{cuts}$ has size $2^{2|V|}$, (namely it consists of all (S,T) pairs), we have $|\mathcal{Q}_{cuts}| \ll (\beta/2)2^{|\mathcal{X}|/6}$ for all reasonable β and $|V|$, we can use the randomized response analysis above to get accuracy

$$O\left(\left(\tbinom{|V|}{2}\log(|\mathcal{Q}_{cuts}|/\beta)\right)^{1/2}/\varepsilon\right) = O((|V|^{3/2} + |V|\log 1/\beta)/\varepsilon)$$

with probability at least $1-\beta$. In fact, one can give a slightly tighter analysis where the accuracy depends on the size of the sets S, T—by observing that the number of random variables participating in a cut query (S,T) is exactly $|S||T|$, one can show that the accuracy for all cuts is whp $O(\varepsilon^{-1}\sqrt{|V||S||T|})$.

Viewing the noisy database $\widehat{\mathcal{D}}$ as a weighted graph $\widehat{G}$, where the weight of (u,v) is $\mathbf{1}_{(u,v)\in E(G)} + \mathrm{Lap}(1/\varepsilon)$, note that $\widehat{G}$ has negative weight edges and hence cannot be considered synthetic data. We can remedy the situation (using the idea of solving a suitable linear program [2, 8]):

Lemma 2 (Synthetic Data for Cuts). *There is a computationally efficient $(\varepsilon, 0)$-differentially private randomized algorithm that takes a unweighted graph G and outputs a synthetic graph G' such that, with high probability, $\|G - G'\|_C \leq O(|V|^{3/2}/\varepsilon)$—all cuts in G and G' are within $O(|V|^{3/2}/\varepsilon)$ additive error.*

The proof is deferred to the full version, but the idea is straightforward: we write a linear program with exponentially many constraints to solve for a synthetic database, and use an SDP-based approximation algorithm of [1] for the cut-norm problem as an approximate separation oracle to solve the LP.

7 Towards Improving on Randomized Response for Synthetic Data

In this section, we consider one possible avenue towards giving an efficient algorithm for privately generating synthetic data for graph cuts that improves over randomized response. We first show how generically, any efficient Iterative Database Construction algorithm can be used to give an efficient *offline* algorithm for privately releasing synthetic data when paired with an efficient *distinguisher*. The analysis here follows the analysis of [12], who analyzed the corresponding algorithm when instantiated with the multiplicative weights algorithm, rather than a generic Iterative Database Construction algorithm.

We will pair an Iterative Database Construction algorithm for a class of queries $\mathcal{C}$ with a corresponding *distinguisher*.

Definition 6 (**$(F(\epsilon),\gamma)$-Private Distinguisher).** *Let $\mathcal{Q}$ be a set of queries, let $\gamma \geq 0$ and let $F(\epsilon) : \mathbb{R}^+ \to \mathbb{Z}$ be a function. An algorithm Distinguish$_\epsilon : \mathbb{N}^{|\mathcal{X}|} \times \mathbb{N}^{|\mathcal{X}|} \to \mathcal{Q}$ is an $(F(\epsilon),\gamma)$-Private Distinguisher for $\mathcal{Q}$ if for every setting of the privacy parameter ϵ, it is ϵ-differentially private with respect to $\mathcal{D}$ and if for every $\mathcal{D}, \mathcal{D}' \in \mathbb{N}^{|\mathcal{X}|}$ it outputs a $Q^* \in \mathcal{Q}$ such that $|Q^*(\mathcal{D}) - Q^*(\mathcal{D}')| \geq \max_{Q\in\mathcal{Q}} |Q(\mathcal{D}) - Q(\mathcal{D}')| - F(\epsilon)$ with probability at least $1 - \gamma$.*

We present the algorithm in the full version, but the idea is very simple. Rather than waiting for a query to arrive online that induces an update step, we find queries which will induce update steps using the distinguisher. The IDC algorithm will guarantee that there will not be too many update steps, and so an efficient distinguisher will yield an efficient algorithm for releasing synthetic data.

Theorem 6. *There is an (ϵ,δ)-differentially private mechanism for releasing synthetic data such that given an $(F(\epsilon),\gamma)$-private distinguisher and a $B(\alpha)$-IDC, it is (α,β)-accurate for:*

$$\alpha \geq \max\left[\frac{16\sqrt{B(\alpha)\log(1/\delta)}\log(2B(\alpha)/\beta)}{\epsilon}, 2F\left(\frac{\epsilon}{4\sqrt{B(\alpha)\log(1/\delta)}}\right)\right]$$

as long as $\gamma \leq \beta/(2B(\alpha))$

We defer the proof until the full version. Note that the running time of the algorithm is dominated by the running time of the IDC algorithm and of the distinguishing algorithm: efficient IDC algorithms paired with efficient distinguishing algorithms for a class of queries $\mathcal{Q}$ automatically correspond to efficient algorithms for privately releasing synthetic data useful for $\mathcal{Q}$. For the class of graph cut queries, both the multiplicative weights IDC and the Frieze/Kannan IDC are computationally efficient. Therefore,

one approach to finding a computationally efficient algorithm for releasing synthetic data useful for cut queries is to find an efficient private distinguisher for cut queries.

One curious aspect of this approach is that it might in fact be computationally easier to release a *larger* class of queries than cut queries, even though this is a strictly more difficult task from an information theoretic perspective. For example, solving the distinguishing problem for cut queries on graphs $\mathcal{D}$ and $\mathcal{D}'$ is equivalent to finding a pair of sets (S, T) which witness the cut-norm on the graph $\mathcal{D} - \mathcal{D}'$. On the other hand, solving the distinguishing problem for rank-1 queries (which include cut queries, and are a larger class) is equivalent to finding the best rank-1 approximation to the adjacency matrix $\mathcal{D} - \mathcal{D}'$. The former problem is NP-hard, whereas the latter problem can be quickly solved non-privately using the singular value decomposition.

Corollary 2. *An efficient $(F(\epsilon), \gamma)$-distinguisher for the class of rank-1 queries for $F(\epsilon) = T/\epsilon$ would yield an (α, β)-accurate mechanism for releasing synthetic data for graph cuts (and all rank-1 queries) for any $\beta \geq \Omega(\exp(-\epsilon T))$ and: $\alpha_{MW} = 2\sqrt[4]{2}\epsilon^{-1/2}\sqrt{Tm}\left(\log|V|\log(1/\delta)\right)^{1/4}$ using the multiplicative weights IDC, or: $\alpha_{FK} \geq 2\epsilon^{-1/2}(m\log(1/\delta))^{1/4}\sqrt{|V|T}$ using the Frieze/Kannan IDC*

The proof, deferred to the full version, only requires plugging in the parameters for these two IDC algorithms. We remark that for the class of rank-1 queries, an efficient $(F(\epsilon), \gamma)$-distinguisher with $F(\epsilon) = \tilde{O}\left(\frac{|V|}{\epsilon}\right)$ would be sufficient to yield an efficient algorithm for releasing synthetic data useful for cut queries, with guarantees matching those of the best known algorithms for the interactive case, as listed in Table 1. For graphs for which the size of the edge set $m \leq \Omega(n^2)$, this would yield an improvement over our randomized response mechanism, which is the best mechanism currently for privately releasing synthetic data for graph cuts. We observe that such a distinguisher is information-theoretically possible, and the only question is whether such a private distinguisher exists that is also computationally efficient. To see this, observe that an $O(|V|)$-net for the set of all rank-1 queries can be constructed by considering all pairs of vectors $x, y \in \{0, 1/|V|, 2/|V|, \ldots, 1\}^{|V|}$ and their associated outer-products $x \cdot y^T$. Since there are at most $|V|^{2|V|}$ such pairs, the exponential mechanism serves as an inefficient $F(\epsilon)$ distinguisher for $F(\epsilon) = O(|V|\log|V|/\epsilon)$.

We note that a distinguisher for rank-1 queries must simply give a good rank-1 approximation to the matrix $\mathcal{D} - \mathcal{D}'$, which will always be symmetric in this setting (because both the hypothesis is at every step simply the adjacency matrix for an undirected graph, as of course is the private database), and hence an algorithm for finding accurate rank-1 approximations merely for symmetric matrices would already yield an algorithm for releasing synthetic data for cuts! Unlike classes of queries like conjunctions, for which their are imposing barriers to privately outputting useful synthetic data [21, 12], there are as far as we know no such barriers to improving our randomized-response based results for synthetic data for graph cuts. We leave finding such an algorithm, for privately giving low rank approximations to matrices, as an intriguing open problem.

Acknowledgements. This paper benefited from interactions with many people. We particularly thank Moritz Hardt and Kunal Talwar for extensive, enlightening discussions. In particular, the observation that randomized response leads to a data structure

for graph cuts with error $O(|V|^{1.5})$ is due to Kunal Talwar. We thank Salil Vadhan for helpful discussions about the Frieze/Kannan low-rank matrix decomposition, and Frank McSherry and Adam Smith for helpful discussions about algorithms for computing low-rank matrix approximations. We thank Cynthia Dwork for always fruitful conversations.

References

[1] Alon, N., Naor, A.: Approximating the cut-norm via Grothendieck's inequality. SIAM J. Comput. 35(4), 787–803 (2006) (electronic)
[2] Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: PODS, pp. 273–282 (2007)
[3] Bhaskara, A., Krishnaswamy, R., Talwar, K.: Unconditional differentially private mechanisms for linear queries (2011) (manuscript)
[4] Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SuLQ framework. In: PODS, pp. 128–138 (2005)
[5] Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: STOC, pp. 609–618 (2008)
[6] Chawla, S., Dwork, C., McSherry, F., Smith, A., Wee, H.: Toward Privacy in Public Databases. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 363–385. Springer, Heidelberg (2005)
[7] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
[8] Dwork, C., Naor, M., Reingold, O., Rothblum, G., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC, pp. 381–390 (2009)
[9] Dwork, C., Rothblum, G., Vadhan, S.: Boosting and differential privacy. In: FOCS, pp. 51–60 (2010)
[10] Frieze, A., Kannan, R.: Quick approximation to matrices and applications. Combinatorica 19(2), 175–220 (1999)
[11] Frieze, A., Kannan, R.: A simple algorithm for constructing Szemerédi's regularity partition. Electron. J. Combin. 6, Research Paper 17, 7 (1999)
[12] Gupta, A., Hardt, M., Roth, A., Ullman, J.: Privately Releasing Conjunctions and the Statistical Query Barrier. In: STOC. ACM, New York (2011)
[13] Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release, Arxiv preprint arXiv:1107.3731 (2011)
[14] Hardt, M., Ligett, K., McSherry, F.: A simple and practical algorithm for differentially private data release, Arxiv preprint arXiv:1012.4763 (2011)
[15] Hardt, M., Rothblum, G.: A multiplicative weights mechanism for privacy-preserving data analysis. In: FOCS, pp. 61–70 (2010)
[16] Hardt, M., Talwar, K.: On the Geometry of Differential Privacy. In: STOC (2010)
[17] Kasiviswanathan, S., Lee, H., Nissim, K., Raskhodnikova, S., Smith, A.: What Can We Learn Privately? In: FOCS, pp. 531–540 (2008)
[18] McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS (2007)
[19] Roth, A., Roughgarden, T.: Interactive Privacy via the Median Mechanism. In: STOC (2010)
[20] Trevisan, L., Tulsiani, M., Vadhan, S.P.: Regularity, boosting, and efficiently simulating every high-entropy distribution. In: CCC (2009)
[21] Ullman, J., Vadhan, S.: PCPs and the Hardness of Generating Private Synthetic Data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)

A Other Iterative Database Construction Algorithms

In this section, we demonstrate how the median mechanism and the multiplicative weights mechanism fit into the IDC framework. These mechanisms apply to general classes of linear queries $\mathcal{Q}$.

A.1 The Median Mechanism

In this section, we show how to use the median database subroutine as an Iterative Database Construction.

Definition 7 (Median Datastructure). *A median datastructure* $\mathbf{D}$ *is a collection of databases* $\mathbf{D} \subset \mathbb{N}^{|\mathcal{X}|}$. *Any query can be evaluated on a median datastructure as follows:* $Q(\mathbf{D}) = \mathit{Median}(\{Q(\mathcal{D}') : \mathcal{D}' \in \mathbf{D}\})$.

Algorithm 3. The Median Mechanism (MM) Algorithm

$\mathbf{U}^{MM}_{k,\alpha}(\mathbf{D}^t, Q^{(t)}, \widehat{A}^{(t)})$

If: $\mathbf{D}^t = \emptyset$ **then:** output $\mathbf{D}^0 = \{\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|} : |\mathcal{D}| = n^2 \log k/\alpha^2\}$

Else if: $Q^{(t)}(\mathbf{D}^t) - \widehat{A}^{(t)} > 0$ **then:** output $\mathbf{D}' = \mathbf{D}' \setminus \{\mathcal{D} \in \mathbf{D} : Q^{(t)}(\mathcal{D}) \geq Q^{(t)}(\mathbf{D})\}$

Else if: $Q^{(t)}(\mathbf{D}^t) - \widehat{A}^{(t)} < 0$ **then:** output $\mathbf{D}' = \mathbf{D}' \setminus \{\mathcal{D} \in \mathbf{D} : Q^{(t)}(\mathcal{D}) \leq Q^{(t)}(\mathbf{D})\}$

Theorem 7. *The Median Mechanism algorithm is a* $B(\alpha) = n^2 \log |\mathcal{X}| \log k/\alpha^2$ *iterative database construction algorithm for every class of* k *linear queries* $\mathcal{Q}$.

Proof. Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database and consider a $(\mathbf{U}^{MM}_k, \mathbf{D}^*, \mathcal{Q}, \alpha, B)$-database update sequence, $\left\{(\mathbf{D}^t, Q^{(t)}, \widehat{A}^{(t)})\right\}_{t=1,\ldots,B}$. It will be sufficient if we can show that $B(\alpha) \leq n^2 \log |\mathcal{X}| \log k/\alpha^2$. Specifically, that after $n^2 \log |\mathcal{X}| \log k/\alpha^2$ invocations of $\mathbf{U}^{MM}_{k,\alpha}$, the median datastructure $\mathbf{D}^{n^2 \log |\mathcal{X}| \log k/\alpha^2}$ is $(\alpha, \mathcal{Q})$-accurate for $\mathcal{D}$. The argument is simple. First, we have a simple fact from [5]:

Claim. For any set of k linear queries $\mathcal{Q}$ and any database $\mathcal{D}$ of size n, there is a database $\mathcal{D}'$ of size $|\mathcal{D}'| = n^2 \log k/\alpha^2$ so that $\mathcal{D}'$ is α-accurate for $\mathcal{D}$ with respect to $\mathcal{Q}$.

From this claim, we have that $|\mathbf{D}^t| \geq 1$ for all t, and so can always be used to evaluate queries. On the other hand, each update step eliminates half of the databases in the median datastructure: $|\mathbf{D}^t| = |\mathbf{D}^{t-1}|/2$. This is because the update step eliminates every database either above or below the median with respect to the last query. Initially $|\mathbf{D}^0| = |\mathcal{X}|^{n^2 \log k/\alpha^2}$, and so there can be at most $B(\alpha) \leq \log n^2 |\mathcal{X}| \log k/\alpha^2$ update steps before we would have $|\mathbf{D}^B| < 1$, a contradiction.

A.2 The Multiplicative Weights Mechanism

In this section we show how to use the multiplicative weights subroutine as an Iterative Database Construction. The analysis of the multiplicative weights algorithm is not new, and follows [15]. It will be convenient to think of our databases in this section as

probability distributions, i.e. normalized so that $||\mathcal{D}||_1 = 1$. Note that if we are α/n accurate for the normalized database, we are α-accurate for the un-normalized database with respect to any set of linear queries.

Algorithm 4. The Multiplicative Weights (MW) Algorithm

$\mathbf{U}_{\alpha}^{MW}(\mathcal{D}^t, Q^{(t)}, \widehat{A}^{(t)})$:

Let $\eta \leftarrow \alpha/(2n)$.

If: $\mathcal{D}^t = \emptyset$ **then:** output $\mathcal{D}' = \mathcal{D} \in \mathbb{R}^{|\mathcal{X}|}$ such that $D_i^0 = 1/|\mathcal{X}|$ for all i.

if $\widehat{A}^{(t)} < Q^{(t)}(\mathcal{D}^t)$ **then**

Let $r_t = Q^{(t)}$

else

Let $r_t = 1 - Q^{(t)}$

end if

Update: For all $i \in [|\mathcal{X}|]$ Let

$$\hat{\mathcal{D}}_i^{t+1} = \exp(-\eta r_t(\mathcal{D}_i^t)) \cdot \mathcal{D}_i^t$$

$$\mathcal{D}_i^{t+1} = \frac{\hat{\mathcal{D}}_i^{t+1}}{\sum_{j=1}^{|\mathcal{X}|} \hat{\mathcal{D}}_j^{t+1}}$$

Output $\mathcal{D}^{t+1}$.

Theorem 8. *The Multiplicative Weights algorithm is a $B(\alpha) = 4n^2 \log|\mathcal{X}|/\alpha^2$ iterative database construction algorithm for every class of linear queries $\mathcal{Q}$.*

Proof. Let $\mathcal{D} \in \mathbb{N}^{|\mathcal{X}|}$ be any database and consider a $(\mathbf{U}^{MW}, \mathcal{D}^*, \mathcal{Q}, \alpha, B)$-database update sequence, $\left\{(\mathcal{D}^{(t)}, Q^{(t)}, \widehat{A}^{(t)})\right\}_{t=1,\ldots,B}$. It will be sufficient if we can show that $B(\alpha) \leq 4n^2 \log|\mathcal{X}|/\alpha^2$. Specifically, that after $4n^2 \log|\mathcal{X}|/\alpha^2$ invocations of $\mathbf{U}^{MW}$, the database $\mathcal{D}^{(4n^2 \log|\mathcal{X}|/\alpha^2)}$ is $(\alpha, \mathcal{Q})$-accurate for $\mathcal{D}$. First let $\hat{\mathcal{D}} \in \mathbb{R}^{|X|}$ be a normalization of the database $\mathcal{D}$: $\hat{\mathcal{D}}_i = \mathcal{D}_i/\|\mathcal{D}\|_1$. Note that for any linear query, $Q(\mathcal{D}) = n \cdot Q(\hat{\mathcal{D}})$. We define:

$$\Psi_t \stackrel{\text{def}}{=} D(\hat{\mathcal{D}}||D^t) = \sum_{i=1}^{|\mathcal{X}|} \hat{\mathcal{D}}_i \log\left(\frac{\hat{\mathcal{D}}_i}{D_i^t}\right)$$

We begin with a simple fact:

Claim ([15]). For all t: $\Psi_t \geq 0$, and $\Psi_0 \leq \log|\mathcal{X}|$.

We will argue that in every step for which $|Q^{(t)}(\mathcal{D}) - Q^{(t)}(\mathcal{D}^t)| \geq \alpha/n$ the potential drops by at least $\alpha^2/4n$. Because the potential begins at $\log|\mathcal{X}|$, and must always be non-negative, we know that there can be at most $B(\alpha) \leq 4n^2 \log|X|/\alpha^2$ steps before the algorithm outputs a database $\mathcal{D}^t$ such that $\max_{Q \in \mathcal{Q}} |Q(\mathcal{D}) - Q(\mathcal{D}^t)| < \alpha/n$, which is exactly the condition that we want.

Lemma 3 ([15])

$$\Psi_t - \Psi_{t+1} \geq \eta \left(r_t(\mathcal{D}^t) - r_t(\mathcal{D}) \right) - \eta^2$$

Proof

$$\begin{aligned}
\Psi_t - \Psi_{t+1} &= \sum_{i=1}^{|\mathcal{X}|} \hat{\mathcal{D}}_i \log \left(\frac{D_i^{t+1}}{D_i^t} \right) \\
&= -\eta r_t(\mathcal{D}) - \log \left(\sum_{i=1}^{|\mathcal{X}|} \exp(-\eta r_t(x_i)) D_i^t \right) \\
&\geq -\eta r_t(\mathcal{D}) - \log \left(\sum_{i=1}^{|\mathcal{X}|} D_i^t (1 + \eta^2 - \eta r_t(x_i)) \right) \\
&\geq \eta \left(r_t(\mathcal{D}^t) - r_t(\mathcal{D}) \right) - \eta^2
\end{aligned}$$

The rest of the proof now follows easily. By the conditions of an iterative database construction algorithm, $|\widehat{A}^{(t)} - Q^{(t)}(\mathcal{D})| \leq \alpha/(2n)$. Hence, for each t such that $|Q^{(t)}(\mathcal{D}) - Q^{(t)}(\mathcal{D}^t)| \geq \alpha/n$, we also have that $Q^{(t)}(\mathcal{D}) > Q^{(t)}(\mathcal{D}_t)$ if and only if $\widehat{A}^{(t)} > Q^{(t)}(\mathcal{D}_t)$. In particular, $r_t = Q^{(t)}$ if $Q^{(t)}(\mathcal{D}^t) - Q^{(t)}(\mathcal{D}) \geq \alpha/n$, and $r_t = 1 - Q^{(t)}$ if $Q^{(t)}(\mathcal{D}) - Q^{(t)}(\mathcal{D}^t) \geq \alpha/n$. Therefore, by Lemma 3 and the fact that $\eta = \alpha/2n$:

$$\Psi_t - \Psi_{t+1} \geq \frac{\alpha}{2n} \left(r_t(\mathcal{D}^t) - r_t(\mathcal{D}) \right) - \frac{\alpha^2}{4n^2} \geq \frac{\alpha}{2n} \left(\frac{\alpha}{n} \right) - \frac{\alpha^2}{4n^2} = \frac{\alpha^2}{4n^2}$$