Recursive Mechanism: Towards Node Differential Privacy and Unrestricted Joins [Full Version, Draft 0.1]*

Shixi Chen
Fudan University
chensx@fudan.edu.cn

Shuigeng Zhou
Fudan University
sgzhou@fudan.edu.cn

ABSTRACT

Existing studies on differential privacy mainly consider aggregation on data sets where each entry corresponds to a particular participant to be protected. In many situations, a user may pose a relational algebra query on a sensitive database, and desires differentially private aggregation on the result of the query. However, no known work is capable to release this kind of aggregation when the query contains unrestricted join operations. This severely limits the applications of existing differential privacy techniques because many data analysis tasks require unrestricted joins. One example is subgraph counting on a graph. Existing methods for differentially private subgraph counting address only edge differential privacy and are subject to very simple subgraphs. Before this work, whether any nontrivial graph statistics can be released with reasonable accuracy under node differential privacy is still an open problem.

In this paper, we propose a novel differentially private mechanism to release an approximation to a linear statistic of the result of some positive relational algebra calculation over a sensitive database. Unrestricted joins are supported in our mechanism. The error bound of the approximate answer is roughly proportional to the *empirical sensitivity* of the query — a new notion that measures the maximum possible change to the query answer when a participant withdraws its data from the sensitive database. For subgraph counting, our mechanism provides the first solution to achieve node differential privacy, for any kind of subgraphs.

1. INTRODUCTION

An important task in data privacy research is to develop mechanisms to publish useful results mined from sensitive database, without disclosing individual privacy. Most of existing techniques provide rather limited privacy protection, since they usually address specific attack models, or rely on specific assumptions about the prior knowledge the potential adversary may possess. In recent years, the paradigm of differential privacy has received increasing attention, because it can provide robust and quantitative privacy guarantee while making no assumptions about the prior knowledge

of the adversary. Data publishing algorithms that achieve differential privacy should guarantee that their outputs are randomized such that input databases differing in one participant are almost indistinguishable to the adversary. Therefore, participating in a database is unlikely to cause privacy breach.

Existing studies on differential privacy are mainly based on a simple data model, where the input database is a set of records, and each record corresponds to a participant. The output of a differentially private data publishing algorithm should have almost identical probability distributions for input data sets that differ in exactly one record. Various kinds of queries that compute aggregations on data sets have been considered, and much effort has been put to linear aggregations, on which more complex queries can be built.

The success of most existing differentially private mechanisms relies on the precondition that the maximum possible change to the query answer resulted from the change of one participant should be small and bounded. Such maximum possible change is called the *sensitivity* of the query, which determines the minimum magnitude of noise needed to introduce into the answer. In practice, however, many databases contain information about not only individual participants, but also relationships between them. The change of one participant may, in the worst case, have potentially unlimited impact on the database and the query answer. Queries on such databases are too complex to be tackled by existing techniques. In this paper, we try to relax the precondition by allowing potentially unbounded impact that may be incurred by new participants joining the database, and give an elegant solution.

1.1 Motivation

Subgraph counting is an important problem in data mining and social networks, which counts the number of occurrences of a given query subgraph in an input graph. Despite of the enormous works on anonymization schemes for private graphs, little has been down to provide quantitative guarantees of privacy and utility. In [12], subgraph counting is studied under a much weaker version of differential privacy. Their privacy guarantee protects only against a specific class of adversaries. The error of the approximate answer returned by their algorithm is large — the magnitude of noise grows exponentially with the number of edges in the subgraph. In [10] and [7], k-triangle and k-star counting are studied, and they achieve better privacy and utility guarantee. In particular, they achieve ϵ-differential privacy for k-star counting, and (ϵ, δ)-differential privacy, a weaker version of differential privacy, for k-triangle counting. However, their work cannot be extended to other kinds of subgraph. It is also worthy of mentioning the work in [5], which gives an algorithm for releasing an approximation to the degree distribution of a graph and achieves k-edge differential privacy.

A major problem of the above works is that they can only achieve edge privacy — each edge corresponds to a participant to be pro-

---

tected. But for many real-world data, such as social networks, each individual participant contributes to the graph a node rather than just an edge. We desire privacy protection based on nodes rather than edges. Unfortunately, it is difficult to achieve node differential privacy while obtaining reasonable query accuracy, because the maximum possible change to the query answer resulted from the change of one node (as well as all of its incident edges) is comparable to the graph size. Prior to our work, whether any nontrivial graph statistics can be released under node differential privacy with reasonable accuracy is still an open problem [7]. It was widely believed that algorithms achieving node differential privacy can only return query answers that are too noisy for practical applications [8, 5]. In this paper, we try to challenge this seemingly impossible task and give a general solution.

In reality, databases usually consist of a number of tables. A participant may contribute tuples to several tables, and a tuple can be contributed collectively by multiple participants. A user may want to issue a SQL query to the database to obtain an output table, then requests approximate statistic of the output table. Subgraph counting is, in fact, a special case of this general context, because every subgraph count can be written as a SELECT query. It will be quite useful if this kind of task can be solved under differential privacy. There have been at least two attempts in the literature [9, 11], which are based on bounding the *global sensitivity* of the query. However, these works support only restricted kinds of join operations, where one participant can affect only constant number of tuples in the output table. Even the most simple subgraph counting requires unrestricted joins where a participant can have unbounded impact on the query answer. Obviously, existing methods are unable to support this kind of joins.

We focuses on the case where the SQL query can be translated into a series of positive relational algebra calculation. We aim at releasing an approximation to a linear statistic of the output table with reasonable accuracy under differential privacy. Our solution covers subgraph counting. Both node and edge differential privacy are achievable, depending on the choice of user. Node differential privacy is stronger than edge differential privacy, but the latter can allow better query accuracy. When nodes or edges of the graph are associated with auxiliary information, our solution also allows arbitrary kinds of constraints imposed on any edges or nodes of the subgraph, which are not supported by prior works.

1.2 Contributions

To develop differentially private mechanisms that can support unrestricted joins, we face several difficulties. First, the problem we study allows one participant to have complex impact on the database. The data model assumed by existing differentially private mechanisms is too simple to suffice our need to express the complex relations between the database and the participants. Hence, new data model is needed to express how participants affect the database content. Second, existing notions of *sensitivity*, including global and local sensitivity, are no longer appropriate in our case, because a new participant joining the database can, in the worst case, have unlimited impact on the query answer, leading to unbounded sensitivity. Thus, it is impossible for us to calibrate the noise to such sensitivities. We need a new metric to measure the least magnitude of noise that is necessary to answer a query. Third, existing works for complex queries often compromise privacy guarantee, utility guarantee or efficiency guarantee. However, such compromise can lead to severe problem for practical use, which limit the applications of those techniques. It is a challenging task to develop mechanisms that can achieve all three guarantees.

Contributions of this paper are as follows:

1) We propose a general model of *sensitive databases*, which allows one participant to affect the database content in any possible way. By formalizing the definition of *neighborhood*, the notion of differential privacy on this data model is setting up such that privacy protection is based on individual participants.

2) We propose a new notion of sensitivity, called *empirical sensitivity*, that measures the maximum possible change to the query answer when a participant withdraws its data from the current database content. Empirical sensitivity is always bounded, and is often small. It gives a better measure of the least magnitude of noise that is necessary to answer a query.

3) We develop a general but inefficient mechanism to answer any monotonic query on a sensitive database. This mechanism guarantees ϵ-differential privacy, and the error bound is roughly proportional to the *global empirical sensitivity* of the query.

4) We propose a specific model of *sensitive databases* based on K-relation or c-table. Every tuple in a K-relation is annotated with a positive Boolean expression that specifies its condition of presence. K-relation is closed under positive relational algebra calculation. Hence it can be used to express the complex relations between the participants and the table output by a SQL query.

5) We develop an efficient mechanism to answer any linear query to a sensitive K-relation. This mechanism guarantees ϵ-differential privacy, and the error bound is roughly proportional to the *universal empirical sensitivity* of the query. The computation cost is in a polynomial of the size of K-relation. Our mechanism is the first solution to the problem of subgraph counting for any subgraphs, which can achieve either node differential privacy or edge differential privacy, and the error bound is roughly proportional to the *local empirical sensitivity* of the query.

6) We conduct extensive experiments to evaluate the proposed mechanism. Experimental results validate the effectiveness and efficiency of the new mechanism.

In Fig. 1 we present a brief comparison between our mechanism and existing mechanisms.

2. PRELIMINARIES

2.1 Privacy and Utility

In this work, we will use differential privacy [2], a state-of-the-art paradigm for privacy preserving data publishing. A randomized algorithm is differentially private if it yields nearly identical distributions over its outcomes when running on neighboring databases.

DEFINITION 1 (DIFFERENTIAL PRIVACY). *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for any pair of neighboring databases D, D', and for any set of possible outputs $S \subseteq Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(D) \in S] \le e^{\epsilon} \cdot \Pr[\mathcal{A}(D') \in S] + \delta \qquad (1)$$

where the probability is taken over the randomness of $\mathcal{A}$. When $\delta = 0$, the algorithm is ϵ-differentially private.

All algorithms presented in this paper satisfy ϵ-differential privacy.

The definition of *neighboring* depends on the context or application. Usually, D and D' are said to be neighboring if they differ only by one participant. In this case, a differentially private algorithm can protect against disclosure of any participant. In the literature, the database D is often considered as a multiset of records, where each record corresponds to a particular participant, then D and D' are neighboring if $|D - D'| + |D' - D| = 1$.

Queries	Our mechanism	Existing mechanisms
Monotonic query on a sensitive database	$\widetilde{O}(\widetilde{GS}_q/\epsilon)$ error, $\mathrm{Exp}(|P|)$ time	None
Linear statistic of the output of a SQL query	$\widetilde{O}(\widetilde{US}_q/\epsilon)$ error, $\mathrm{Poly}(|P|,|R|)$ time	$O(US_q/\epsilon)$ error and $O(1)$ time if there are no unrestricted joins [9, 11] Not solvable if there are unrestricted joins because $US_q \geq GS_q = +\infty$
triangle counting ($\triangle$)	$\widetilde{O}(\widetilde{LS}_q/\epsilon)$ error, $\mathrm{Poly}(k,|R|)$ time	$O(LS_q/\epsilon + 1/\epsilon^2)$ error; $O(|V|\cdot|E|)$ time; only achieve differential privacy based on edges [10]
k-star counting (e.g., 3-star)	$\widetilde{O}(\widetilde{LS}_q/\epsilon)$ error, $\mathrm{Poly}(k,|R|)$ time or $\mathrm{Poly}(|V|,|E|,k)$ time	$O(LS_q/\epsilon)$ error if $1/\epsilon = O(d_{\max}/k)$; $O(|V|\cdot|E|)$ time; only achieve differential privacy based on edges [7]
k-triangle counting (e.g., 3-triangle)	$\widetilde{O}(\widetilde{LS}_q/\epsilon)$ error, $\mathrm{Poly}(k,|R|)$ time or $\mathrm{Poly}(|V|,|E|,k)$ time	$O(LS_q/\epsilon)$ error if $\ln(1/\delta)/\epsilon = O(a_{\max})$; $O(|V|\cdot|E|)$ time; only achieve (ϵ,δ)-differential privacy based on edges [7]
k-node l-edge connected subgraph counting	$\widetilde{O}(\widetilde{LS}_q/\epsilon)$ error, $\mathrm{Poly}(k,l,|R|)$ time	$\Theta((kl^2 \log|V|)^{l-1}/\epsilon)$ error; $O(1)$ time; only achieve adversary privacy based on edges w.r.t. a specific class of adversaries [12]

Figure 1: Comparison between our mechanism and existing mechanisms. $\widetilde{O}$ means that logarithmic factors are omitted. For sensitive database, $|P|$ denotes the number of participants and $|\mathrm{supp}(R)|$ denotes the number of tuples returned by the SQL query. For subgraph counting, $|V|$ and $|E|$ denote the number of nodes and edges in the graph, and $|R| = |\mathrm{supp}(R)|$ denotes the true query answer. $d_{\max}$ denotes the maximum degree of a node, and $a_{\max}$ denotes the maximum number of common neighbors of a pair of nodes. GS, LS, US, $\widetilde{GS}$, $\widetilde{LS}$ and $\widetilde{US}$ are explained in Sec. 2 and Sec. 3. We have $\widetilde{LS}_q \leq LS_q$ and $\widetilde{US}_q \leq US_q$. Note that we do not take account of the time needed for generating the output table or the list of matched subgraphs in the computation cost. For subgraph counting our solution can achieve differential privacy based on either nodes or edges, depending on the choice of user.

We are interested in queries that are real-valued functions of the database (though other kinds of queries are also important). A differentially private algorithm must introduce randomness to its output, and the answer is never exact. The utility of the algorithm is measured by how accurate its answer is.

DEFINITION 2 ((ϵ, δ)-ACCURATE). *For a database D, a query q and the true answer $q(D)$, we say that the answer returned by an algorithm $\mathcal{A}$ is (ϵ, δ)-accurate if*

$$\Pr[|\mathcal{A}(D) - q(D)| > \epsilon] \leq \delta \qquad (2)$$

2.2 Global Sensitivity

A well known approach to achieve differential privacy is Laplace mechanism [2], which introduces i.i.d. noises into the query answers. The magnitude of noise is calibrated to the *sensitivity* of the query — a property of the query that measures the maximum possible change to the true answer caused by a small change in the database.

DEFINITION 3 (GLOBAL SENSITIVITY). *For a real-valued function $q : \mathbb{D} \to \mathbb{R}^m$, the (global) sensitivity of q is*

$$GS_q = \max_{D,D' \in \mathbb{D}} \|q(D) - q(D')\|_1 \qquad (3)$$

where the maximum is taken over all pairs of neighboring databases D, D'.

Given a database $D \in \mathbb{D}$, a query sequence $q : \mathbb{D} \to \mathbb{R}^m$ and a parameter $\epsilon > 0$, Laplace mechanism $\mathcal{A}$ returns $\mathcal{A}(D) = q(D) + (Y_1, \ldots, Y_m)$, where Y_i are i.i.d. random variables that

follow Laplace distribution $\mathrm{Lap}(GS_q/\epsilon)$, which has the following probability density function

$$\mathrm{Lap}(y|b) = \frac{1}{2b} \exp(-\frac{|y|}{b}) \qquad (4)$$

Laplace mechanism satisfies ϵ-differential privacy. It is easy to verify that Laplace mechanism returns $(cGS_q/\epsilon, e^{-c})$-accurate answer to each query in the sequence q, for any $c > 0$.

2.3 Local Sensitivity and Smooth Sensitivity

In Laplace mechanism, the magnitude of noise depends on GS_q and the parameter ϵ, but not on the database D. Since the global sensitivity GS_q measures the impact of a participant on the true answer in the worst case, this often introduces unnecessarily large noise. In [10], a local measure of sensitivity was proposed

DEFINITION 4 (LOCAL SENSITIVITY). *For a real-valued function $q : \mathbb{D} \to \mathbb{R}^m$ and a database $D \in \mathbb{D}$, the local sensitivity of q at D is*

$$LS_q(D) = \max_{D'} \|q(D) - q(D')\|_1 \qquad (5)$$

where the maximum is taken over the neighborhood of D.

Observing that $GS_q = \max_D LS_q(D)$, we know that $LS_q(D)$ never exceeds GS_q. Ideally, we would like to release $q(D)$ with noise magnitude proportional to $LS_q(D)$, but the noise magnitude might leak information and differential privacy is not satisfied. [10] proposed that the noise magnitude should be calibrated to a smooth upper bound S on the local sensitivity, namely, a function S that is an upper bound on LS_f at all point and such that $\ln(S(\cdot))$ has low global sensitivity. [10] presents algorithms to compute the optimal S, called the *smooth sensitivity* of q, for a variety of queries.

2.4 K-Relation and c-Table

Our work addresses aggregation on relations where each tuple could be contributed by multiple participants, and each participant could contribute multiple tuples. To track which participants contribute a tuple and how they contribute, we use K-relation [4] or c-table [6], a model proposed in the field of uncertain databases, where tuples are annotated (tagged) with their provenance information, and positive relational algebra is generalized to such tagged-tuple relations. Here we briefly review K-relation and c-table.

Let U be a finite set of attributes and $\mathbb{C}$ a domain of values, then each tuple is a function $t : U \to \mathbb{C}$. The set of all such U-tuples is denoted by U-Tup. Relations without annotations are just subsets of U-Tup. Tuples in a K-relation are annotated with elements from a semiring $(K, +, \cdot, 0, 1)$. A K-relation over U is a function $R : U\text{-Tup} \to K$ with a finite support $\text{supp}(R) = \{t | R(t) \neq 0\}$. The operations of positive algebra on K-relation are defined as follows [4]:

empty relation For any set of attributes U, there is $\emptyset : U\text{-Tup} \to K$ such that $\emptyset(t) = 0$ for all t.

union For $R_1, R_2 : U\text{-Tup} \to K$, $R_1 \cup R_2 : U\text{-Tup} \to K$ is defined by

$$(R_1 \cup R_2)(t) = R_1(t) + R_2(t)$$

projection For $R : U\text{-Tup} \to K$ and $V \subseteq U$, $\pi_V R : V\text{-Tup} \to K$ is defined by

$$(\pi_V R)(t) = \sum_{t = t' \text{ on } V \text{ and } R(t') \neq 0} R(t')$$

selection For $R : U\text{-Tup} \to K$ and a selection predicate $P : U\text{-Tup} \to \{0, 1\}$, $\sigma_P R : U\text{-Tup} \to K$ is defined by

$$(\sigma_P R)(t) = R(t) \cdot P(t)$$

natural join For $R_i : U_i\text{-Tup} \to K$, $i = 1, 2$, $R_1 \bowtie R_2 : (U_1 \cup U_2)\text{-Tup} \to K$ is defined by

$$(R_1 \bowtie R_2)(t) = R_1(t_1) \cdot R_2(t_2)$$

where $t_1 = t$ on U_1 and $t_2 = t$ on U_2.

renaming For $R : U\text{-Tup} \to K$ and a bijection $\beta : U \to U'$, $\rho_\beta R : U'\text{-Tup} \to K$ is defined by

$$(\rho_\beta R)(t) = R(t \circ \beta)$$

Intersection and *cartesian product* are just special cases of natural join. But *difference* is not supported in positive relational algebra.

We study differentially private aggregation on a c-table, which is a special case of K-relation where K makes up of positive Boolean expressions over some set B of variables. The term *positive* means that the expressions do not involve negation ($\neg$), but only disjunction ($\vee$), conjunction ($\wedge$) and constants True and False. In our work, each variable in B may correspond to a (potential) participant being protected, then the Boolean expression annotated with a tuple t gives the condition of t being presented in the relation when some participants may opt out.

In c-table or K-relation, expressions that yield the same truth-value for all valuation of variables in B are considered equivalent. But this is not applicable to our work. An expression $(b_1 \vee b_2) \wedge (b_1 \vee b_3)$ cannot be simply rewritten into $b_1 \vee (b_2 \wedge b_3)$. Such rewriting could make our mechanism fail to satisfy differential privacy. We will review this issue later.

3. PROBLEM FORMULATION

3.1 Sensitive Databases and Monotonic Queries

In the literature of differential privacy, a *sensitive database* is typically considered as a multiset of records, and the privacy is defined by the indistinguishability between data sets that differ by only one record. But this definition of privacy is no longer appropriate in our case, where each participant could have complex effect on the database. To achieve differential privacy in our setting, we need to know about not only the content of the database, but also how it changes if some participants withdraw their data. A sensitive database being released should contain such self-descriptive information. We propose a new definition of *sensitive database*, as below, which is more general.

DEFINITION 5 (SENSITIVE DATABASE). *A sensitive database is an ordered pair (P, M), where P is finite set of participants contributing the data, and M is a function $M : \mathcal{P}(P) \to \mathbb{D}$ such that $M(P')$ is the content of the database if only participants in P' contribute their data.*

Once sensitive databases are formalized, we are ready to adapt the notion of differential privacy to them by making clear what sensitive databases are considered neighboring with each other. We say that two sensitive databases are neighboring if one database can be obtained from the other by one participant withdrawing its data.

DEFINITION 6 (NEIGHBORING). *Two sensitive databases (P_1, M_1) and (P_2, M_2) are neighboring if $|P_1 - P_2| + |P_2 - P_1| = 1$ and $M_1(P') = M_2(P')$ for all $P' \subseteq P_1 \cap P_2$.*

DEFINITION 7 (ANCESTOR). *We say that (P_1, M_1) is an ancestor of (P_2, M_2), denoted by $(P_1, M_1) \preceq (P_2, M_2)$, if $P_1 \subseteq P_2$ and $M_1(P') = M_2(P')$ for all $P' \subseteq P_1$.*

We postulate a class Ω of sensitive databases, such that every possible sensitive database being considered is an element of Ω. Moreover, if $(P, M) \in \Omega$, then all ancestors of (P, M) are also elements of Ω. We make a further assumption that there is a special element D_0 in $\mathbb{D}$ such that $M(\emptyset) = D_0$ for all $(P, M) \in \Omega$ (otherwise, Ω comprises disconnected parts).

For a sensitive database (P, M), a query q takes as input $M(P)$, the current content of the database, and outputs $q(M(P))$. In this paper, we address queries that output a real number and are monotonic.

DEFINITION 8 (MONOTONIC QUERY). *For a class Ω of sensitive databases, a query $q : \mathbb{D} \to \mathbb{R}$ is monotonic if both of following hold:*

- $q(D_0) = 0$

- $q(M_1(P_1)) \leq q(M_2(P_2))$ *for all $(P_1, M_1) \preceq (P_2, M_2)$*

If the global sensitivity of a query is low, then Laplace mechanism can still be applied to obtaining differentially private answer with reasonable accuracy. In many applications, however, the change of a participant could, in the worst case, incur excessive or even unlimited impact on the database content as well as the query answer. No existing differentially private techniques can process queries with unbounded global/local sensitivity. Hence, global/local sensitivity is no longer an appropriate quantity to measure the necessary amount of noise introduced into the query answer. We propose a new notion of sensitivity, *empirical sensitivity*, which suffices our need.

DEFINITION 9 (LOCAL EMPIRICAL SENSITIVITY). *For a real-valued function $q : \mathbb{D} \to \mathbb{R}^m$ and a sensitive database (P, M), the local empirical sensitivity of q at (P, M) is*

$$\widetilde{LS}_q(P, M) = \max_{p \in P} \|q(M(P)) - q(M(P - \{p\}))\|_1 \quad (6)$$

If $P = \emptyset$, then $\widetilde{LS}_q(P, M) = 0$.

DEFINITION 10 (GLOBAL EMPIRICAL SENSITIVITY). *For a real-valued function $q : \mathbb{D} \to \mathbb{R}^m$ and a sensitive database (P, M), the global empirical sensitivity of q at (P, M) is*

$$\widetilde{GS}_q(P, M) = \max_{(P', M') \preceq (P, M)} \widetilde{LS}_q(P', M') \quad (7)$$

Empirical sensitivity measures the maximum possible change to the query answer when a participant opts out. It is obvious that $\widetilde{LS}_q(P, M) \leq LS_q(M(P)) \leq GS_q$ and $\widetilde{LS}_q(P, M) \leq \widetilde{GS}_q(P, M) \leq GS_q$.

3.2 Linear Queries on Sensitive Relations

Although the model of sensitive databases and monotonic queries is general, it may be too general to allow efficient mechanism for obtaining differentially private answer. We are in particular interested in a special class of monotonic queries that compute linear aggregation on a relation, and the relation is itself a function of the sensitive database.

DEFINITION 11. *A linear query q on sensitive database is a function $q = q_+ \circ q_*$, where $q_* : \mathbb{D} \to \mathcal{P}(U\text{-Tup})$ and $q_+ : \mathcal{P}(U\text{-Tup}) \to \mathbb{R}$, such that q_* transforms a database $D \in \mathbb{D}$ into a finite set of tuples (e.g. by some relational algebra calculation), and q_+ is a linear function: $q_+(T) = \sum_{t \in T} q_+(t)$.*

Note that the output of q_* must be finite, although the space U-Tup can be infinite.

To ensure that a linear query q is monotonic, we pose some limitations on the functions q_* and q_+. First, we require that introducing a new participant into a sensitive database never results in removal of any tuple from the relation output by q_*. Second, we assume that q_+ is nonnegative.

DEFINITION 12. *A linear query $q = q_+ \circ q_*$ on sensitive database is monotonic if the following hold:*

- $q_*(M_1(P_1)) \subseteq q_*(M_2(P_2))$ *for all* $(P_1, M_1) \preceq (P_2, M_2)$
- $q_+(T) \geq 0$ *for all finite* $T \subseteq U$-Tup

If we want to answer a linear function q_+ that may yield negative output, we can decompose it into two nonnegative components and compute them individually: $q_+(t) = \max(0, q_+(t)) - \max(0, -q_+(t))$.

Because we focus on a single query, where q_* is fixed, we can construct a class of virtual sensitive databases $\Omega' = \{(P, M')\}$, such that each (P, M) in Ω is mapped into a virtual one (P, M') where $M' = q_* \circ M$. Then $M'(P)$ is a set of tuples and the query $q = q_+ \circ q_*$ is just a linear function that computes $q_+(M'(P))$. The monotonicity of q_* transmits to the monotonicity of M'. We call such (P, M') a *sensitive relation*.

DEFINITION 13. *A sensitive relation (P, M) is a sensitive database with $M : \mathcal{P}(P) \to \mathcal{P}(U\text{-Tup})$, and $M(P)$ must be finite. A class Ω of sensitive relations is monotonic if $M_1(P_1) \subseteq M_2(P_2)$ for all $(P_1, M_1) \preceq (P_2, M_2)$ in Ω.*

In this subsection and most parts of this paper, we study nonnegative linear queries for a monotonic class of sensitive relations.

To obtain a differentially private answer to a query q on a relation $T = M(P)$, it should specify how the relation T is affected by its contributors P. In particular, we want to know for each tuple in T the condition of its presence if some participants may opt out. The definition of the function M is too general to be efficiently handled in practice. Therefore, we propose to represent M as a c-table or K-relation R, where each tuple t is annotated with a positive Boolean expression $R(t)$ that specifies its condition of presence. Each variable p in an expression indicates whether the participant $p \in P$ would contribute its data. A sensitive relation represented as a K-relation is called a sensitive K-relation, denoted by (P, R).

For a query q, an algorithm may first transform the original sensitive database (P, M) into a sensitive K-relation (P, R) in a flexible way. For the correctness of the differentially private mechanism, however, the transformation should guarantee that for any neighboring sensitive databases the corresponding sensitive K-relations are also neighboring. The concept of *neighboring* for sensitive K-relations is defined by as follows.

DEFINITION 14. *Given an equivalence relation $\sim$ on K, two sensitive K-relations (P_1, R_1) and (P_2, R_2), where $P_2 = P_1 \cup \{p\}$, $p \notin P_1$, are neighboring if $R_1(t) \sim R_2(t)_{|p \to \text{False}}$ for all $t \in U\text{-Tup}$, where $R_2(t)_{|p \to \text{False}}$ denotes an operation that replaces all occurrences of the variable p in $R_2(t)$ with constant False.*

An issue in the above definition is that it does not specify what kinds of Boolean expressions in K are equivalent. A necessary condition for two expressions being equivalent is that they must yield the same truth-value for all valuation of variables. The way we write the expressions may, or may not matter, depending on the particular algorithms being used. For example, the inefficient mechanism presented in Sec. 4.2 is independent of the form of expressions, so expressions that yield the same truth table are equivalent. On the other hand, the efficient mechanism presented in Sec. 5 relies on the way we write an expression. We will discuss this in Sec. 5.

In Fig. 2 we present simple examples of K-relations that are produced by different queries to a graph. Fig. 2(a) is a subgraph counting, while Fig. 2(b) is a more complicated query.

Finally, we introduce a variant of empirical sensitivity, which is relevant to the error bound of our mechanism.

DEFINITION 15 (IMPACT). *For a sensitive K-relation (P, R) and a participant $p \in P$, the impact of p at R is*

$$\text{impact}(p, R) = \{t : R(t) \not\sim R(t)_{|p \to \text{False}}\} \quad (8)$$

DEFINITION 16 (UNIVERSAL EMPIRICAL SENSITIVITY). *For a sensitive K-relation (P, R), a participant $p \in P$ and a nonnegative linear query q, the universal empirical sensitivity of q for a participant p at R is*

$$\widetilde{US}_q(p, R) = \sum_{t \in \text{impact}(p, R)} q(t) \quad (9)$$

For a sensitive K-relation (P, R) and a nonnegative linear query q, the universal empirical sensitivity of q at (P, R) is

$$\widetilde{US}_q(P, R) = \max_{p \in P} \widetilde{US}_q(p, R) \quad (10)$$

When $q(t) = 1$ for all t, $\widetilde{US}_q(p, R)$ measures how many tuples in R have p appearing in their annotated expressions. The error bound of our mechanism presented in Sec. 5 is roughly proportional to the universal empirical sensitivity $\widetilde{US}_q$.

(a) social network (graph) — nodes b, d, f (top row); a, c, e (bottom row)

node differential privacy

t	$R(t)$
abc	$a \wedge b \wedge c$
bcd	$b \wedge c \wedge d$
cde	$c \wedge d \wedge e$

edge differential privacy

t	$R(t)$
abc	$e_{ab} \wedge e_{ac} \wedge e_{bc}$
bcd	$e_{bc} \wedge e_{bd} \wedge e_{cd}$
cde	$e_{cd} \wedge e_{ce} \wedge e_{de}$

(a) How many triangles in a social network

node differential privacy

t	$R(t)$
ab	$a \wedge b \wedge c$
ac	$a \wedge c \wedge b$
bc	$b \wedge c \wedge (a \vee d)$
bd	$b \wedge d \wedge c$
cd	$c \wedge d \wedge (b \vee e)$
ce	$c \wedge e \wedge d$
de	$d \wedge e \wedge c$

edge differential privacy

t	$R(t)$
ab	$e_{ab} \wedge e_{ac} \wedge e_{bc}$
ac	$e_{ac} \wedge e_{ab} \wedge e_{bc}$
bc	$e_{bc} \wedge ((e_{ab} \wedge e_{ac}) \vee (e_{bd} \wedge e_{cd}))$
bd	$e_{bc} \wedge e_{bd} \wedge e_{cd}$
cd	$e_{cd} \wedge ((e_{bc} \wedge e_{bd}) \vee (e_{ce} \wedge e_{de}))$
ce	$e_{ce} \wedge e_{cd} \wedge e_{de}$
de	$e_{de} \wedge e_{cd} \wedge e_{ce}$

(b) How many pairs of friends that have a common friend

Figure 2: Examples of K-relations

4. THE RECURSIVE MECHANISM FRAMEWORK

In this section, we first present the framework of a novel differential privacy mechanism, *recursive mechanism*, which can answer any monotonic queries on any sensitive databases. Then, we give a general but inefficient implementation of the mechanism.

4.1 The Basic Framework

Our mechanism is based on two special sequences, $H_0(P, M) \cdots H_{|P|}(P, M)$ and $G_0(P, M) \cdots G_{|P|}(P, M)$, as functions of the sensitive database (P, M) in Ω. We call H a recursive sequence, which should satisfy the conditions given by the following definition.

DEFINITION 17 (RECURSIVE SEQUENCE). *A sequence, $H_0(P, M) \dots H_{|P|}(P, M)$, as a function on Ω, is called a recursive sequence if the following conditions hold:*

- $H_0(P, M) = 0$ *for all* $(P, M) \in \Omega$

- *(Recursive Monotonicity)* $H_i(P_2, M_2) \leq H_i(P_1, M_1) \leq H_{i+1}(P_2, M_2)$ *for all neighboring* $(P_1, M_1) \preceq (P_2, M_2)$ *in* Ω *and* $0 \leq i \leq |P_1|$

We call G a bounding sequence of H, which is also a recursive sequence but satisfies some additional condition.

DEFINITION 18 (BOUNDING SEQUENCE). *For a recursive sequence H and $g \geq 1$, a sequence, $G_0(P, M) \dots G_{|P|}(P, M)$, as a function on Ω, is called a g-bounding sequence of H, if the following conditions hold:*

- G *is a recursive sequence*

- $H_j(P, M) \leq H_i(P, M) + (|P| - i)G_k(P, M)$ *for all* $(P, M) \in \Omega$ *and all* $0 \leq i \leq j \leq |P|$ *and* $k = |P| - \lfloor (|P| - j)/g \rfloor$

If $g = 1$, we simply say G is a bounding sequence of H.

The framework of our mechanism consists of three steps

1. For a monotonic query q, we construct a recursive sequence H and a g-bounding sequence G of H such that $H_{|P|}(P, M) = q(M(P))$ for all $(P, M) \in \Omega$.

2. Based on G, find a quantity Δ such that Δ approximates $G_{|P|}(P, M)$ or the empirical sensitivity of q, and $\ln \Delta$ has low global sensitivity, then we add multiplicative noise to Δ, obtaining $\widehat{\Delta}$, which satisfies differential privacy.

3. Based on H, find a quantity X such that X approximates the true answer $H_{|P|}(P, M)$, and X has global sensitivity $\widehat{\Delta}$, then we add Laplace noise to X, obtaining $\widehat{X}$, which satisfies differential privacy.

The concrete construction of H and G are omitted here. We focus on Step 2 and 3 in this subsection. In the remainder of this paper, we will omit the argument (P, M) when the context is clear.

For a sensitive database (P, M) and parameters $\beta > 0$ and $\theta > 0$, we compute Δ as following

$$\Delta = \min\{e^{i\beta}\theta : G_{|P|-i} \leq e^{i\beta}\theta\} \qquad (11)$$

We can observe several important properties of Δ. In the sequel, all proofs of lemmas and theorems are moved to the appendix.

LEMMA 1. $GS_{\ln \Delta} \leq \beta$.

LEMMA 2. $\Delta \leq \max\{\theta, e^{\beta}G_{|P|}\}$.

LEMMA 3. $G_{|P|-\ln(\frac{\Delta}{\theta})/\beta} \leq \Delta$.

Because $\ln \Delta$ has low global sensitivity, we can add Laplace noise to $\ln \Delta$ to obtain a noisy version $\widehat{\Delta}$ that satisfies differential privacy. For parameter $\epsilon_1 > 0$ and $\mu > 0$, we compute $\widehat{\Delta} = e^{\mu+Y}\Delta$, where $Y \sim \text{Lap}(\beta/\epsilon_1)$. This finishes Step 2, and $\widehat{\Delta}$ has several properties.

LEMMA 4. *The release of $\widehat{\Delta}$ satisfies ϵ_1-differential privacy.*

LEMMA 5. $\Pr[\widehat{\Delta} > e^{\mu+c}\Delta] \leq \frac{1}{2}e^{-c\epsilon_1/\beta}$ *for any $c > 0$.*

LEMMA 6. $\Pr[\widehat{\Delta} < \Delta] \leq \frac{1}{2}e^{-\mu\epsilon_1/\beta}$.

In Step 3, we first find a quantity X such that X approximates $H_{|P|}(P, M)$ and $GS_X \leq \widehat{\Delta}$. We compute X as

$$X = \min\{H_i + (|P| - i)\widehat{\Delta} : 0 \leq i \leq |P|\} \qquad (12)$$

We have several properties of X.

LEMMA 7. *For any fixed $\widehat{\Delta} \geq 0$, $GS_X \leq \widehat{\Delta}$.*

LEMMA 8. *If $\widehat{\Delta} \geq \Delta$, then $H_{|P|-g \ln(\frac{\widehat{\Delta}}{\theta})/\beta} \leq X \leq H_{|P|}$.*

For parameter $\epsilon_2 > 0$, our mechanism releases $\widehat{X} = X + Y$, where $Y \sim \text{Lap}(\widehat{\Delta}/\epsilon_2)$. We give the privacy and utility guarantees in the following theorem.

THEOREM 1. *For parameters $\epsilon_1 > 0$, $\epsilon_2 > 0$, $\beta > 0$, $\theta > 0$ and $\mu > 0$, recursive mechanism, as described above, satisfies $(\epsilon_1 + \epsilon_2)$-differential privacy, and is $(e^{2\mu}\Delta^* c/\epsilon_2 + g\lceil \ln(\frac{\Delta^*}{\theta})/\beta \rceil G_{|P|}, e^{-\mu\epsilon_1/\beta} + e^{-c})$-accurate for any $c > 0$, where $\Delta^* = \max\{\theta, e^\beta G_{|P|}\}$. If $\epsilon_1 = \Theta(\epsilon)$, $\epsilon_2 = \Theta(\epsilon)$, $\beta = \epsilon_1/k$, and θ and μ are constants, then the mechanism is $(O(k\ln(G_{|P|})G_{|P|}/\epsilon), 2e^{-k\mu})$-accurate as $\epsilon \to 0, k \to \infty$ and $G_{|P|} \to \infty$.*

The error bound of recursive mechanism is roughly proportional to $G_{|P|}$. Hence, the most important thing in a concrete implementation of recursive mechanism is to find sequences H and G with $G_{|P|}$ as small as possible.

4.2 A General but Inefficient Implementation

Now we present a general but inefficient implementation of the recursive mechanism, which can answer any monotonic queries on sensitive databases. For a monotonic query q, we construct H and G as follows:

$$H_i(P, M) = \min_{(P', M') \preceq (P, M), |P'|=i} q(M'(P')) \qquad (13)$$

$$G_i(P, M) = \min_{(P', M') \preceq (P, M), |P'|=i} \widetilde{GS}_q(P', M') \qquad (14)$$

$$(15)$$

Then we can show that the above H and G are what we want.

THEOREM 2. *The sequence H is a recursive sequence, and the sequence G is a bounding sequence of H.*

Because $G_{|P|}(P, M) = \widetilde{GS}_q(P, M)$, the error bound of recursive mechanism using these H and G is roughly proportional to the global empirical sensitivity of q. The main disadvantage of this implementation is the expensive computation cost for H and G.

5. EFFICIENT RECURSIVE MECHANISM

In this section, we present an efficient recursive mechanism, which takes polynomial computation cost and can answer linear queries on sensitive K-relations.

5.1 Recursive Mechanism with Relaxation

The central idea of the efficient recursive mechanism is relaxation, which introduces a mapping $\phi : K \to [0,1]^{[0,1]^P}$ that maps each Boolean expression in K into a $[0,1]$-valued expression $\phi_k : [0,1]^P \to [0,1]$. The detail of ϕ will be discussed in the next subsection. Now, we first give some required properties of ϕ.

For simplifying notations, we let True = 1 and False = 0. For $f : P \to [0,1]$, we define $|f| = \sum_p f(p)$. By $f \leq g$ we mean $f(p) \leq g(p)$ for all p. The mapping ϕ has the following properties.

Correctness For any $k \in K$ and any Boolean assignment $f : P \to \{0,1\}$, $\phi_k(f) = k(f)$.

Naturalness For any $k \in K$, any real assignment $f : P \to [0,1]$ and any $p \in P$, if $f(p) = 0$, then $\phi_k(f) = \phi_{k|_{p\to\text{False}}}(f)$, and if $f(p) = 1$, then $\phi_k(f) = \phi_{k|_{p\to\text{True}}}(f)$.

Monotonicity For any $k \in K$ and any real assignments $f, g : P \to [0,1]$, if $f \leq g$, then $\phi_k(f) \leq \phi_k(g)$.

Convexity For any $k \in K$, ϕ_k is a convex function.

Truncated Linearity Define $\psi(x) = \min(1, x)$ and $\phi_k^*(f) = 1 - \phi_k(1 - \psi \circ f)$. For any $k \in K$, $f : P \to [0,1]$ and $c \geq 1$, $\phi_k^*(cf) = \min(1, c\phi_k^*(f))$

Then, we introduce the notion of *equivalence* — two Boolean expressions in K are equivalent if their relaxed functions under ϕ are the same. This completes Definition 14 for neighboring sensitive K-relations.

DEFINITION 19 (EQUIVALENCE). *For any $k_1, k_2 \in K$, k_1 and k_2 are equivalent, denoted by $k_1 \sim k_2$, if $\phi_{k_1} = \phi_{k_2}$.*

Equivalence of two expressions implies that they yield the same truth table. But expressions that yield the same truth table are not necessarily equivalent. We will explain this in the next subsection.

Provided a nonnegative linear query $q : U\text{-Tup} \to \mathbb{R}$ and mapping ϕ, we construct the recursive sequence H as

$$H_i(P, R) = \min_{f \in [0,1]^P, |f|=i} \sum_t q(t)\phi_{R(t)}(f) \qquad (16)$$

Note that the sum is finite since R has finite support.

THEOREM 3. *The sequence H is a recursive sequence, and $H_{|P|}(P, R) = q(\text{supp}(R))$.*

To construct the bounding sequence of H, we also require that an auxiliary quantity $S_{k,p}$ is provided for each $k \in K$ and $p \in P$, which bounds the maximum change of $\phi_k(f)$ caused by a small change to $f(p)$. Formally, for all $f, g \in [0,1]^P$, if $f \leq g$, and $f(p') = g(p')$ for all $p' \in P - \{p\}$, then

$$\phi_k(g) - \phi_k(f) \leq (g(p) - f(p))S_{k,p} \qquad (17)$$

$S_{k,p}$ can be seen as the upper bound of the partial derivative of ϕ_k w.r.t. p. We call $S_{k,p}$ the ϕ-sensitivity of the expression k for p. We can observe the following fact.

LEMMA 9. *For any $f \leq g$ in $[0,1]^P$, and any $k \in K$,*

$$\phi_k(g) - \phi_k(f) \leq \sum_p (g(p) - f(p))S_{k,p} \leq |g - f| \max_p S_{k,p} \quad (18)$$

Assuming that all ϕ-sensitivities $S_{k,p}$ are known, we construct a 2-bounding sequence G of H as

$$G_i(P, R) = 2 \min_{f \in [0,1]^P, |f|=i} \max_{p \in P} \sum_t q(t)\phi_{R(t)}(f)S_{R(t),p} \quad (19)$$

THEOREM 4. *The sequence G is a 2-bounding sequence of H.*

5.2 The Mapping ϕ

Here we discuss the mapping ϕ, the issues about annotation of Boolean expressions, and the utility guarantee of the recursive mechanism. For an expression k, we define ϕ_k in a recursive way, as follows:

- $\phi_{\text{False}}(f) = 0$ and $\phi_{\text{True}}(f) = 1$ for all f

- $\phi_p(f) = f(p)$ for all $p \in P$

- $\phi_{x \wedge y}(f) = \max\{0, \phi_x(f) + \phi_y(f) - 1\}$ and $\phi_{x \vee y}(f) = \max\{\phi_x(f), \phi_y(f)\}$ for all expressions x and y

It can be shown that the above ϕ is just what we need.

THEOREM 5. *The mapping ϕ, defined above, have the desired properties of correctness, naturalness, monotonicity, convexity, and truncated linearity.*

The output of mapping ϕ is invariant under certain kinds of transformations of the input expressions.

Identity $\phi_{x \wedge \text{True}} = \phi_x$, $\phi_{x \vee \text{False}} = \phi_x$

Annihilator $\phi_{x \wedge \text{False}} = \phi_{\text{False}}$, $\phi_{x \vee \text{True}} = \phi_{\text{True}}$

Associativity $\phi_{x \wedge (y \wedge z)} = \phi_{(x \wedge y) \wedge z}$, $\phi_{x \vee (y \vee z)} = \phi_{(x \vee y) \vee z}$

Distributivity of $\wedge$ over $\vee$ $\phi_{x \wedge (y \vee z)} = \phi_{(x \wedge y) \vee (x \wedge z)}$

Two expressions are equivalent if one can be obtained from another via a series of above transformations. Because ϕ is defined recursively, the above transformations can be applied to any place of an expression k without changing ϕ_k.

Before invoking our mechanism, one needs to first generate a sensitive K-relation from the sensitive database and then issue a monotonic query. To satisfy differential privacy, it is important to ensure that for any neighboring sensitive databases the resulting sensitive K-relations are still neighboring, according to Definition 14. Hence, when we annotate tuples with expressions that specify their conditions of presence, we should take care of the way we write the expressions. Specifically, if a tuple t is annotated with expression k, then we should ensure that when any participant p opts out, the new expression k' annotated with t can be obtained from $k_{|p \rightarrow \text{False}}$ via a series of invariant transformations. If this is guaranteed, then we say the annotation is safe. Fortunately, safe annotation is often easy to achieve. For positive relational algebra queries, the annotation provided in Sec. 2.4 is always safe. Moreover, two expressions in disjunctive normal form are equivalent if and only if they produce the same truth table. Therefore, if we always expand all expressions into disjunctive normal form, then the annotation is always safe.

The ϕ-sensitivities $S_{k,p}$, which bound the partial derivative of ϕ_k w.r.t. p, are also computed in a recursive way

- $S_{\text{True},p} = S_{\text{False},p} = 0$ and $S_{p,p} = 1$

- $S_{x \wedge y,p} = S_{x,p} + S_{y,p}$ and $S_{x \vee y,p} = \max\{S_{x,p}, S_{y,p}\}$

We can observe several properties of ϕ-sensitivities: 1) $S_{k,p}$ is not greater than the number of occurrences of p in expression k; 2) $S_{k,p}$ is at most one plus the the number of occurrences of $\wedge$ in k; 3) if k is written in disjunctive normal form (e.g., the case of subgraph counting), then $S_{k,p} \leq 1$; 4) for positive relational algebra query, if each tuple in the input tables is associated with at most one participant, and we use the approach described in Sec. 2.4 to annotate the tuples in the output table with expressions, then $S_{k,p}$ is at most one plus the number of operations in the positive relational algebra query. In Fig. 3 we present several examples of ϕ-sensitivity.

If we take the maximum S of $S_{k,p}$ over all $k \in \{R(t)\}$ and $p \in P$, then we can find that $G_{|P|}(P, R) \leq 2S \cdot \widetilde{US}_q(P, R)$. Hence, we conclude that the error bound of our mechanism is roughly proportional to S times the universal empirical sensitivity of q. In general, S is linear in the length of the positive relational algebra query. If all expressions are converted to disjunctive normal form, then S is just a constant 1. In particular, for subgraph counting we have $\widetilde{US}_q = \widetilde{GS}_q = \widetilde{LS}_q$. Thus the error bound is roughly proportional to the local empirical sensitivity of q.

5.3 Computation Cost

Note that the computation for each H_i and G_i can be encoded into a linear program with $O(L)$ variables, where L denotes the total length of all annotated expressions $R(t)$ for $t \in \text{supp}(R)$. Therefore, our mechanism can run in polynomial time.

expression k	ϕ-sensitivities of k
$a \wedge b \wedge c$	$S_{k,a} = S_{k,b} = S_{k,c} = 1$
$(a \vee b) \wedge (a \vee c) \wedge (b \vee d)$	$S_{k,a} = S_{k,b} = 2, S_{k,c} = S_{k,d} = 1$
$(a \wedge b) \vee (a \wedge c) \vee (b \wedge d)$	$S_{k,a} = S_{k,b} = S_{k,c} = S_{k,d} = 1$

Figure 3: Examples of ϕ-sensitivities

A simple algorithm that computes all H_i and G_i will need to solve $O(|P|)$ linear programs. We can improve this by utilizing the monotonicity of G and the convexity of H.

LEMMA 10. *(Convexity of H) $H_{i+1} - H_i \leq H_{i+2} - H_{i+1}$ for all $0 \leq i \leq |P| - 2$.*

Let $j = \arg\min\{e^{j\beta}\theta : G_{|P|-j} \leq e^{j\beta}\theta\}$, then $\Delta = e^{j\beta}\theta$. We can observe that $j = \ln(\frac{\Delta}{\theta})/\beta \leq 1 + \ln(\frac{G_{|P|}}{\theta})/\beta$. Hence, Δ can be computed with access to the last $O(\ln(G_{|P|})/\beta)$ entries of G. Furthermore, because $G_{|P|-j} - e^{j\beta}\theta$ is monotonously decreasing, we can use binary search to find j, with access to only $O(\ln(\ln(G_{|P|})/\beta))$ entries of G.

Given Δ and $\widetilde{\Delta}$, we then compute $X = H_i + (|P|-i)\widehat{\Delta}$, where $i = \arg\min\{H_i + (|P|-i)\widehat{\Delta} : 0 \leq i \leq |P|\}$. To do this, we compute

$$i' = \arg\min_{i' \in [0, |P|]} H_{i'} + (|P|-i')\widehat{\Delta} \qquad (20)$$

In the above formula, the range of i' is a real interval rather than an integer, and the definition of $H_{i'}$ is the same as Eq. 16. So i' can be computed by solving a linear program. Due to the convexity of H, we also know that $\lfloor i' \rfloor \leq i \leq \lceil i' \rceil$. Hence, i can then be computed with access to only two entries of H.

THEOREM 6. *Efficient recursive mechanism can run in $O(\ln(\ln(G_{|P|})/\beta)T(L))$ time, where $T(L)$ denotes time needed to solve a linear program with $O(L)$ variables, and L denotes the total length of all annotated expressions $R(t)$ for $t \in \text{supp}(R)$.*

6. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the performance of our mechanism. We first compare our mechanism with existing mechanisms for answering subgraph counting queries, then we use our mechanism to process more general K-relations.

For each experiment, we generate several different graphs (or K-relations) by random, and for each graph we run every mechanism many times to obtain a series of answers. We measure the accuracy of mechanisms by median relative error, that is, the median of the ratios between the absolute errors and the true answers. This measure of accuracy is consistent with the work [7].

6.1 Subgraph Counting

For subgraph counting, we compare the accuracy of our mechanism with the following existing mechanisms:

Local sensitivity mechanisms include the triangle algorithm of [10], the k-star algorithm and the k-triangle mechanism of [7]. All algorithms are based on the local sensitivity of the query. The k-triangle algorithm achieves only (ϵ, δ)-differential privacy, while the others can achieve ϵ-differential privacy.

RHMS mechanism of [12] can process subgraph counting for any connected subgraphs. It achieves only (ϵ, γ)-adversarial privacy for a specific class of adversaries.

8

(a) Comparison on graphs with various number of nodes. avgdeg = 10.

(b) Comparison on graphs with various average degrees. $|V| = 200$.

(c) Comparison with various ϵ. $|V| = 200$ and avgdeg = 10.
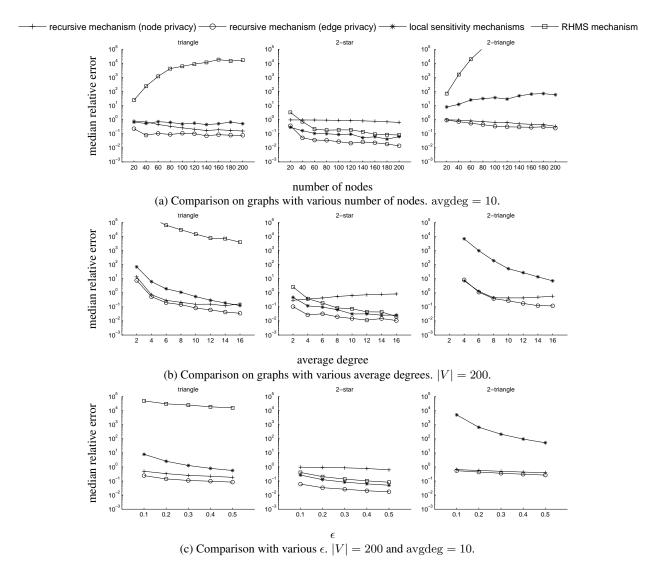
Figure 4: Comparing accuracy of different mechanisms in various settings.

We set $\epsilon = 0.5$ and $\delta = \gamma = 0.1$, which follows the parameter setting of [7].[1] Our mechanism can achieve ϵ-differential privacy, which is much stronger than the corresponding (ϵ, δ)-differential privacy and (ϵ, γ)-adversarial privacy. We test two versions of our mechanism, one provides node privacy, and the other provides edge privacy. Because node privacy requires that the released answer must be insensitive to the change of one node and all of its incident edges, it needs to introduce noise of much greater magnitude into the answer. Note that all other mechanisms in comparison can only provide edge privacy. For our mechanism, we simply set $\theta = 1$, $\beta = \epsilon/5$ and $\mu = 0.5$, and we set $\mu = 1$ for node differential privacy.

We first perform experiments on synthetic graphs that are generated by random. We generate graphs with various numbers of nodes and average degree avgdeg. Each edge in the graph appears inde-
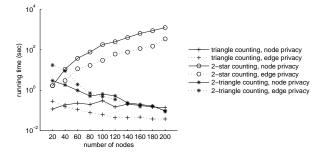

Figure 5: Running time of recursive mechanism, avgdeg = 10.

pendently with probability avgdeg $/(|V| - 1)$. The experimental results are presented in Fig. 4.

It can be observed that RHMS mechanism does not yield meaningful answers for triangle counting and 2-triangle counting. This is because that its error bound grows exponentially with the number of edges in the subgraph. In some experiments, the relative errors of

---

[1]It is widely believed by researchers that to provide useful privacy guarantee δ should be a negligible function of database size [10, 1, 3, 13] (i.e., δ is asymptotically smaller than any inverse polynomial: $\delta = 1/|P|^{\omega(1)}$). However, the k-triangle algorithm [7] yields too noisy answers for such small δ.

	netscience	power	1138_bus	bcspwr10	gemat12	ca-GrQc	ca-HepTh
$|V|$	1589	4941	1138	5300	4929	5242	9877
$|E|$	2742	6594	2596	13571	33111	14496	25998
total number of triangles							
	3764	651	128	721	592	48260	28339
running time of recursive mechanism in seconds (node privacy)							
	8.924	0.468	0.078	0.655	0.640	3940.552	788.160
running time of recursive mechanism in seconds (edge privacy)							
	19.188	0.374	0.063	0.406	0.483	54236.462	2104.818

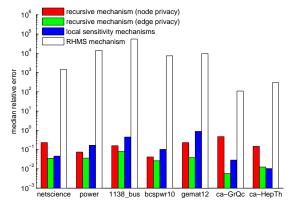**Figure 6: Sizes of real graphs and running time of our mechanism for triangle counting.**

Figure 7: Comparing accuracy of different mechanisms for triangle counting on different graphs.

Figure 8: Evaluating recursive mechanism on K-relations with various length of expressions, $|\operatorname{supp}(R)| = 1000$.

Figure 9: Evaluating recursive mechanism on K-relations of various sizes, each expression has 3 clauses.

RHMS mechanism are extremely high and the curves do not show in the figures. Moreover, the errors of local sensitivity mechanisms are also too high to be useful for triangle counting and 2-triangle counting when the graph is very sparse, because the smooth upper bound of local sensitivity is often high (relative to the true answer) for triangle counting on sparse graphs.

Our mechanism, when providing edge privacy (the same as other compared mechanisms), always yield the most accurate answers. When providing node privacy, our mechanism has high relative error for 2-star counting and 2-triangle counting, this is because the change of one node can affect a large number of 2-stars and 2-triangles in the graph. Nonetheless, the relative error of our mechanism decreases while the size of graph grows.

In Fig. 5, we present the running time of our mechanism. Because each matched subgraph found in the whole graph contributes a tuple into the K-relation, the computation cost of our mechanism grows polynomially with the true answer. Since the average degree is fixed, the number of triangles and 2-triangles often decreases when the graph enlarges, hence our mechanism runs faster for large sparse random graphs. On the other hand, the number of 2-stars is roughly proportional to the number of nodes, so the running time of our mechanism grows with the graph size[2].

We also evaluate the mechanisms on several real datasets[3]. Experimental results are shown in Fig. 6 and 7. We can see that our mechanism are often superior to the other mechanisms. This validates the practical usage of our mechanism.

6.2 Processing K-Relations

---

[2]When the degrees of nodes are large, the number of k-stars and k-triangles can grow exponentially with k. One may think that our mechanism has exponential computation cost in this situation. Actually, the algorithm can be improved by a clever construction of K-relation, such that the size of K-relation is asymptotically independent of k. Due to limitation of space, we cannot present the details in this paper.
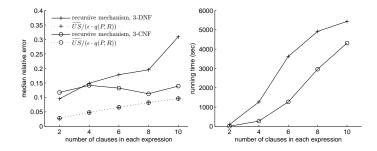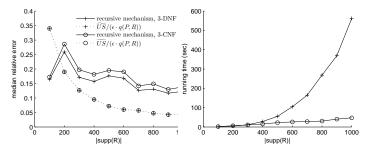
[3]Available at http://www.cise.ufl.edu/research/sparse/matrices/

Finally, we evaluate the performance of our mechanism for processing more general queries. Because there are many different kinds of positive relational algebra queries, we directly generate K-relations that could be produced by some relational queries. In particular, we consider two kinds of K-relations: K-relations in which every tuple is annotated with a 3-DNF Boolean expression, and K-relations in which every tuple is annotated with a 3-CNF Boolean expression. A 3-DNF K-relation can be produced by a union of many join results, and a 3-CNF K-relation can be produced by a join of many unions of tables. We simply generate all expressions by random, but ensure that all annotated expressions have the same length. We also make $|P|$, the total number of variables, equal to $|\operatorname{supp}(R)|$, the size of the K-relation. We let $q(t) = 1$, that is, the true answer is just $|\operatorname{supp}(R)|$. The performance of our mechanism is shown in Fig. 8 and 9. We do not present experimental results for different kinds of $q(t)$ because the curves are almost the same.

The dotted curves in the figures denote the relative error if the absolute error exactly matches $\widetilde{US}_q/\epsilon$, where $\widetilde{US}_q$ is the maximum number of tuples that have at least one common participant appearing in their annotated expressions. The error of our mechanism is nearly linear in $\widetilde{US}_q/\epsilon$, as shown in the figures. The empirical sensitivity $\widetilde{US}_q$ is insensitive to the increase of the number of participants and the number of tuples in R. Hence the relative error of our mechanism can gradually decrease if more data are available. In terms of computation cost, the running time of our mechanism grows polynomially with $|\operatorname{supp}(R)|$ and the length of expressions.

7. RELATED WORK

Since differential privacy was introduced [2], it has gained considerable attention, and many techniques were developed for private data analysis. Dwork *et al.* [2] showed that differential privacy

can be achieved if we calibrate the noise to the global sensitivity of the query, and proposed the Laplace mechanism. The noise yielded by Laplace mechanism is independent of the database instance. Due to simplicity and wide applicability of Laplace mechanism, many succeeding work for various query tasks were built upon Laplace mechanism. These include the relevant work [9, 11] that studied relational algebra queries under differential privacy.

Laplace mechanism fails to provide useful answers for queries that have large global sensitivity. Nissim *et al.* [10] introduced the notion of local sensitivity, and proposed to calibrate the noise to a smooth upper bound of the local sensitivity. This leads to the idea of instance-dependent noise. They also gave algorithms for computing smooth sensitivity of triangle count as well as some other statistics in a variety of domains. However, there are no general way to compute the smooth sensitivity of a given query.

Inspired by the work [10], Karwa *et al.* [7] studied the problem of k-star counting and k-triangle counting, which were based on the local sensitivity of the query. Rastogi *et al.* [12] addressed counting of general subgraphs, which achieves utility better than global sensitivity based mechanism by relaxing the privacy guarantee. These approaches only provide edge privacy guarantee.

There were also extensive studies on privacy in graph data beyond the scope of differential privacy, but most do not provide qualitative privacy and utility guarantee. Readers can refer to the survey [14] for techniques that are based on k-anonymity.

8. CONCLUSION

In this paper, we have presented a novel differentially private mechanism for releasing an approximation to a linear statistic of a table output by some positive relational algebra query to a database. It turns subgraph counting as a special case, and can provide guarantee of either node differential privacy or edge differential privacy. Empirical evaluation shows that our mechanism can return more accurate answer than existing algorithms for subgraph counting, while achieving the same or even stronger privacy guarantee.

9. REFERENCES

[1] C. Dwork and J. Lei. Differential privacy and robust statistics. In *STOC*, pages 371–380, 2009.

[2] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[3] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

[4] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[5] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.

[6] T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[7] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.

[8] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD Conference*, pages 193–204, 2011.

[9] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD Conference*, pages 19–30, 2009.

[10] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007. Full paper: http://www.cse.psu.edu/ asmith/pubs/NRS07.

[11] C. Palamidessi and M. Stronati. Differential privacy for relational algebra: improving the sensitivity bounds via constraint systems. In *QAPL*, 2012.

[12] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, pages 107–116, 2009.

[13] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.

[14] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations*, 10(2):12–22, 2008.

APPENDIX

A. PROOFS

LEMMA 1. $GS_{\ln \Delta} \le \beta$.

PROOF SKETCH. For all neighboring $(P_1, M_1) \preceq (P_2, M_2)$, let $i = \arg\min_i \{e^{i\beta}\theta : G_{|P_1|-i}(P_1, M_1) \le e^{i\beta}\theta\}$ and $j = \arg\min_j \{e^{j\beta}\theta : G_{|P_2|-j}(P_2, M_2) \le e^{j\beta}\theta\}$, we show that $i \le j \le i+1$. Note that G is a recursive sequence. Because $e^{(i-1)\beta}\theta < G_{|P_1|-(i-1)}(P_1, M_1) \le G_{|P_2|-(i-1)}(P_2, M_2)$, we have $j \ge i$. Similarly, because $e^{(i+1)\beta}\theta \ge G_{|P_1|-i}(P_1, M_1) \ge G_{|P_2|-(i+1)}(P_2, M_2)$, we have $j \le i+1$. $\square$

LEMMA 2. $\Delta \le \max\{\theta, e^{\beta} G_{|P|}\}$.

PROOF SKETCH. Suppose $\Delta = e^{i\beta}\theta$. If $i = 0$, then $\Delta = \theta$, otherwise, $\Delta = e^{\beta} e^{(i-1)\beta}\theta < e^{\beta} G_{|P|-i} \le e^{\beta} G_{|P|}$. $\square$

LEMMA 3. $G_{|P|-\ln(\frac{\Delta}{\theta})/\beta} \le \Delta$.

PROOF SKETCH. Suppose $\Delta = e^{i\beta}\theta$. Then the lemma is true because $\ln(\frac{\Delta}{\theta})/\beta = i$. $\square$

LEMMA 5. $\Pr[\widehat{\Delta} > e^{\mu+c}\Delta] \le \frac{1}{2} e^{-c\epsilon_1/\beta}$ *for any* $c > 0$.

PROOF SKETCH.

$$\Pr[\widehat{\Delta} > e^{\mu+c}\Delta] = \Pr_{Y \sim \text{Lap}(\beta/\epsilon_1)}[Y > c] \qquad (21)$$

$$= \Pr_{Y \sim \text{Lap}(1)}[Y > c\epsilon_1/\beta] \qquad (22)$$

$$= \frac{1}{2} e^{-c\epsilon_1/\beta} \qquad (23)$$

$\square$

LEMMA 6. $\Pr[\widehat{\Delta} < \Delta] \le \frac{1}{2} e^{-\mu\epsilon_1/\beta}$.

PROOF SKETCH. The same as the previous lemma. $\square$

LEMMA 7. *For any fixed* $\widehat{\Delta} \ge 0$, $GS_X \le \widehat{\Delta}$.

PROOF SKETCH. For all neighboring $(P_1, M_1) \preceq (P_2, M_2)$, let $i = \arg\min_i H_i(P_1, M_1) + (|P_1| - i)\widehat{\Delta}$ and $j = \arg\min_j H_j(P_2, M_2) + (|P_2| - j)\widehat{\Delta}$. Then, we have

$$X(P_1, M_1) = H_i(P_1, M_1) + (|P_1| - i)\widehat{\Delta} \qquad (24)$$

$$\le H_{j-1}(P_1, M_1) + (|P_1| - (j-1))\widehat{\Delta} \qquad (25)$$

$$\le H_j(P_2, M_2) + (|P_2| - j)\widehat{\Delta} \qquad (26)$$

$$= X(P_2, M_2) \qquad (27)$$

$$X(P_2, M_2) = H_j(P_2, M_2) + (|P_2| - j)\widehat{\Delta} \tag{28}$$

$$\leq H_i(P_2, M_2) + (|P_2| - i)\widehat{\Delta} \tag{29}$$

$$\leq H_i(P_1, M_1) + (|P_1| - i + 1)\widehat{\Delta} \tag{30}$$

$$= X(P_1, M_1) + \widehat{\Delta} \tag{31}$$

□

LEMMA 8. *If $\widehat{\Delta} \geq \Delta$, then $H_{|P| - g \ln(\frac{\Delta}{\theta})/\beta} \leq X \leq H_{|P|}$.*

PROOF SKETCH. The second inequality is obvious, so we show the first inequality. Let $i = \arg\min_i H_i + (|P| - i)\widehat{\Delta}$ and suppose $\Delta = e^{j\beta}\theta$, then

$$H_{|P| - g \ln(\frac{\Delta}{\theta})/\beta} = H_{|P| - gj} \tag{32}$$

$$\text{(by the property of } g\text{-bounding sequence)}$$

$$\leq H_i + (|P| - i)G_{|P| - j} \tag{33}$$

$$\leq H_i + (|P| - i)\Delta \tag{34}$$

$$\leq H_i + (|P| - i)\widehat{\Delta} \tag{35}$$

$$= X \tag{36}$$

□

THEOREM 1. *For parameters $\epsilon_1 > 0$, $\epsilon_2 > 0$, $\beta > 0$, $\theta > 0$ and $\mu > 0$, recursive mechanism, as described above, satisfies $(\epsilon_1 + \epsilon_2)$-differential privacy, and is $(e^{2\mu}\Delta^* c/\epsilon_2 + g\lceil \ln(\frac{\Delta^*}{\theta})/\beta \rceil G_{|P|}, e^{-\mu\epsilon_1/\beta} + e^{-c})$-accurate for any $c > 0$, where $\Delta^* = \max\{\theta, e^\beta G_{|P|}\}$. If $\epsilon_1 = \Theta(\epsilon)$, $\epsilon_2 = \Theta(\epsilon)$, $\beta = \epsilon_1/k$, and θ and μ are constants, then the mechanism is $(O(k \ln(G_{|P|})G_{|P|}/\epsilon), 2e^{-k\mu})$-accurate as $\epsilon \to 0, k \to \infty$ and $G_{|P|} \to \infty$.*

PROOF SKETCH. The privacy guarantee is obvious since both the computation of $\widehat{\Delta}$ and $\widehat{X}$ satisfy differential privacy. The utility guarantee is also true because

1) with probability at least $1 - e^{-\mu\epsilon_1/\beta}$, we have $\Delta < \widehat{\Delta} < e^{2\mu}\Delta$;

2) with probability at least $1 - e^{-c}$, we have $|\widehat{X} - X| \leq \widehat{\Delta}c/\epsilon_2$;

3) if $\widehat{\Delta} \geq \Delta$, we have $|X - H_{|P|}| \leq (g \ln(\frac{\Delta}{\theta})/\beta)G_{|P|}$ □

THEOREM 2. *The sequence H is a recursive sequence, and the sequence G is a bounding sequence of H.*

PROOF SKETCH. For any neighboring $(P_1, M_1) \preceq (P_2, M_2)$, $P_1 \cup \{p\} = P_2$, and for any $0 \leq i \leq |P_1|$, $y \in \{1, 2\}$, let $P_y^i = \arg\min_{P' \subseteq P_y, |P'| = i} q(M_y(P'))$. Then, because $H_0 = 0$ and

$$H_i(P_2, M_2) = q(M_2(P_2^i)) \leq q(M_2(P_1^i)) \tag{37}$$

$$= q(M_1(P_1^i)) \tag{38}$$

$$= H_i(P_1, M_1) \tag{39}$$

$$H_i(P_1, M_1) = q(M_1(P_1^i)) \leq q(M_1(P_2^{i+1} - \{p\})) \tag{40}$$

$$\leq q(M_2(P_2^{i+1})) \tag{41}$$

$$= H_{i+1}(P_2, M_2) \tag{42}$$

H is a recursive sequence. The same reasoning also applies to G being a recursive sequence.

Now we show that G is a bounding sequence of H. For any $0 \leq i \leq j \leq |P|$, let $A = \arg\min_{P' \subseteq P, |P'| = i} q(M(P'))$, and let $B = \arg\min_{P' \subseteq P, |P'| = i} \widetilde{GS}_q(P', M)$. Then

$$H_j \leq q(M(B)) \tag{43}$$

$$\leq q(M(A \cap B)) + |B - A|\widetilde{GS}_q(B, M) \tag{44}$$

$$\leq q(M(A)) + (|P| - i)\widetilde{GS}_q(B, M) \tag{45}$$

$$= H_i + (|P| - i)G_j \tag{46}$$

□

For the following proofs, we define $f \cup g, f \cap g : P \to [0, 1]$ by $(f \cup g)(p) = \max(f(p), g(p))$ and $(f \cap g)(p) = \min(f(p), g(p))$. We also define $f_p : P \to [0, 1]$ as an indicator function that has $f_p(p) = 1$ and $f_p(p') = 0$ for all $p' \neq p$, and let $f_{P'} = \sum_{p \in P'} f_p$.

THEOREM 3. *The sequence H is a recursive sequence, and $H_{|P|}(P, R) = q(\text{supp}(R))$.*

PROOF SKETCH. $H_{|P|}(P, R) = q(\text{supp}(R))$ is obvious due to correctness of ϕ. We will show that H is a recursive sequence.

For any neighboring $(P_1, R_1) \preceq (P_2, R_2)$, $P_1 \cup \{p\} = P_2$, and for any $0 \leq i \leq |P_1|$, $y \in \{1, 2\}$, let $f_y^i = \arg\min_{f \in [0,1]^{P_y}, |f| = i} \sum_t q(t)\phi_{R_y(t)}(f)$. Then, we have $H_0 = 0$ and

$$H_i(P_2, R_2) = \sum_t q(t)\phi_{R_2(t)}(f_2^i) \tag{47}$$

$$\leq \sum_t q(t)\phi_{R_2(t)}(f_1^i) \tag{48}$$

$$\text{(naturalness of } \phi)$$

$$= \sum_t q(t)\phi_{R_2(t)|_{p \to \text{False}}}(f_1^i) \tag{49}$$

$$(R_1(t) \text{ and } R_2(t)|_{p \to \text{False}} \text{ are equivalent})$$

$$= \sum_t q(t)\phi_{R_1(t)}(f_1^i) \tag{50}$$

$$= H_i(P_1, R_1) \tag{51}$$

Due to $|f_2^{i+1} \cap (1 - f_p)| \geq i$ and monotonicity of ϕ, we have

$$H_i(P_1, R_1) = \sum_t q(t)\phi_{R_1(t)}(f_1^i) \tag{52}$$

$$\leq \sum_t q(t)\phi_{R_1(t)}(f_2^{i+1} \cap (1 - f_p)) \tag{53}$$

$$= \sum_t q(t)\phi_{R_2(t)}(f_2^{i+1} \cap (1 - f_p)) \tag{54}$$

$$\leq \sum_t q(t)\phi_{R_2(t)}(f_2^{i+1}) \tag{55}$$

$$= H_{i+1}(P_2, R_2) \tag{56}$$

□

THEOREM 4. *The sequence G is a 2-bounding sequence of H.*

PROOF SKETCH. The proof for G being a recursive sequence is the same as the proof for H. Now we show that for any $0 \leq i \leq j \leq |P|$, we have $H_j \leq H_i + (|P| - i)G_k$, where $k = |P| - \lfloor (|P| - j)/2 \rfloor$.

Let $h = \arg\min_{h \in [0,1]^P, |h| = i} \sum_t q(t)\phi_{R(t)}(h)$, $g = \arg\min_{g \in [0,1]^P, |g| = k} 2 \max_p \sum_t q(t)\phi_{R(t)}(g)S_{R(t),p}$, and $f(p) = \max(0, 1 - 2(1 - g(p)))$ for all $p \in P$. We first observe that, due to truncated linearity of ϕ, if $\phi_{R(t)}(f) > 0$, then $\phi_{R(t)}(g) > 0.5$. Thus,

$$H_j \leq \sum_t q(t)\phi_{R(t)}(f) \quad \text{(note that } |f| \geq j) \tag{57}$$

$$\leq \sum_t q(t)\phi_{R(t)}(h \cap f) +$$

12

$$|f - h \cap f| \max_p \sum_{t:\phi_{R(t)}(f)>0} q(t)S_{R(t),p} \qquad (58)$$

$$(\phi_{R(t)}(f) > 0 \Rightarrow \phi_{R(t)}(g) > 0.5) \qquad (59)$$

$$\leq \sum_t q(t)\phi_{R(t)}(h \cap f)+ $$

$$|f - h \cap f| \max_p \sum_t 2q(t)\phi_{R(t)}(g)S_{R(t),p} \qquad (60)$$

$$\leq \sum_t q(t)\phi_{R(t)}(h) + (|P| - i)G_k \qquad (61)$$

$$= H_i + (|P| - i)G_k \qquad (62)$$

□

THEOREM 5. *The mapping ϕ, defined above, have the desired properties of correctness, naturalness, monotonicity, convexity, and truncated linearity.*

PROOF SKETCH. These properties can be easily proved by induction, so we omit the details. □

LEMMA 10. *(Convexity of H) $H_{i+1} - H_i \leq H_{i+2} - H_{i+1}$ for all $0 \leq i \leq |P| - 2$.*

PROOF SKETCH. First note that the function $h(f) = \sum_t q(t)\phi_{R(t)}(f)$ is convex, due to the convexity of ϕ. Then, let $f^i = \arg\min_{f \in [0,1]^P, |f|=i} h(f)$. We have

$$H_{i+1} = h(f^{i+1}) \leq h((f^i + f^{i+2})/2) \qquad (63)$$

$$\text{(convexity of } h)$$

$$\leq (h(f^i) + h(f^{i+2}))/2 \qquad (64)$$

$$= (H_i + H_{i+2})/2 \qquad (65)$$

□