A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries

Wei Dong, Ke Yi
Hong Kong University of Science and Technology
Hong Kong, China
{wdongac,yike}@cse.ust.hk

ABSTRACT

Releasing the result size of conjunctive queries and graph pattern queries under differential privacy (DP) has received considerable attention in the literature, but existing solutions do not offer any optimality guarantees. We provide the first DP mechanism for this problem with a fairly strong notion of optimality, which can be considered as a natural relaxation of instance-optimality to a constant.

CCS CONCEPTS

• **Information systems** → **Database query processing**; • **Security and privacy** → **Database and storage security**; • **Theory of computation** → **Theory of database privacy and security**.

KEYWORDS

Differential privacy; Neighborhood optimal; Conjunctive queries

ACM Reference Format:
Wei Dong, Ke Yi. 2022. A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3517804.3524143

1 INTRODUCTION

The bulk of work on *differential privacy (DP)* has been devoted to counting queries [17, 35]. For a query q, let $q(\mathbf{I})$ be the results of evaluating q on database instance $\mathbf{I}$, and let $|q(\mathbf{I})|$ be the cardinality of $q(\mathbf{I})$. A DP mechanism $\mathcal{M}_q(\mathbf{I})$ for a counting query q aims to release $|q(\mathbf{I})|$ masked with noise so as to satisfy the DP requirement. Formally, $\mathcal{M}_q(\cdot)$ is *ε-differentially private* if

$$\Pr[\mathcal{M}_q(\mathbf{I}) = y] \le e^{\varepsilon} \Pr[\mathcal{M}_q(\mathbf{I}') = y] \tag{1}$$

for any y and any pair of neighboring instances $\mathbf{I}, \mathbf{I}'$, i.e., $d(\mathbf{I}, \mathbf{I}') = 1$. Here, $d(\mathbf{I}, \mathbf{I}')$ denotes the minimum number of changes to turn $\mathbf{I}$ into $\mathbf{I}'$. DP policies may vary depending on what constitutes a "change". In strict DP [22, 26], a change can be inserting a tuple, deleting a tuple, or substituting a tuple by another, while in a relaxed version, only substitutions are allowed [6, 25]. The latter is more relaxed since there are less neighboring instances that must satisfy (1). In particular, neighboring instances must have the same size, so a DP mechanism may use the instance size $N = |\mathbf{I}|$ directly; on the other hand, N must be protected in strict DP. All the positive results in this paper hold for strict DP, while negative results hold even for the relaxed version. The privacy parameter ε is usually taken as a constant, which in practice ranges from 0.1 to 10.

Any (nontrivial) DP mechanism must be probabilistic by definition. Thus, the most common measure of the utility of $\mathcal{M}_q(\cdot)$ is its expected ℓ_2-error:

$$\begin{aligned}\mathrm{Err}(\mathcal{M}_q, \mathbf{I}) &= \sqrt{\mathsf{E}\left[(\mathcal{M}_q(\mathbf{I}) - |q(\mathbf{I})|)^2\right]} \\ &= \sqrt{(\mathsf{E}[\mathcal{M}_q(\mathbf{I})] - |q(\mathbf{I})|)^2 + \mathsf{Var}[\mathcal{M}_q(\mathbf{I})]},\end{aligned}$$

which consists of a bias term and a variance term. Between two mechanisms with the same $\mathrm{Err}(\mathcal{M}_q, \mathbf{I})$, the unbiased one is often preferred. All our DP mechanisms will be unbiased, in which case $\mathrm{Err}(\mathcal{M}_q, \mathbf{I}) = \sqrt{\mathsf{Var}[\mathcal{M}_q(\mathbf{I})]}$, while our lower bounds hold even for biased mechanisms.

The problem is now well understood for selection queries with various types of selection conditions [7, 10, 20, 31, 32, 36, 38]. However, the case when q is a conjunctive query (CQ) is still largely open. Existing solutions [22, 28, 30] are heuristics without any notion of optimality. Notably, there have been extensive studies on graph pattern counting queries [8, 9, 11, 23, 24, 33, 37], which are a special case of CQs equipped with inequalities. But none of them has any theoretical guarantee on the utility, either.

1.1 Sensitivity-based DP Mechanisms

Let $\mathbf{I}$ be a database instance and q a counting query. A widely used framework for designing an $\mathcal{M}_q$ is to first compute some measure of *sensitivity* $S_q(\mathbf{I})$, and then add noise drawn from a certain zero-mean distribution calibrated to $S_q(\mathbf{I})$. From now on, we may omit the subscript q from $\mathcal{M}_q$ and S_q when the context is clear. Note that all sensitivity-based mechanisms are unbiased. Various noise distributions (Section 2 gives more details) have been studied and they can all achieve $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = \Theta(S(\mathbf{I}))$, where the hidden constant depends on ε and the particular distribution. Thus, the problem essentially boils down to finding the smallest $S(\mathbf{I})$ that satisfies the DP requirement. When we say that a particular sensitivity measure $S(\cdot)$ is optimal, this actually means that the DP mechanism $\mathcal{M}(\mathbf{I})$ that adds noise drawn from a certain distribution calibrated to $S(\mathbf{I})$ achieves the optimal (in whatever sense) $\mathrm{Err}(\mathcal{M}, \mathbf{I})$.

The main reason for the lack of a good theoretical analysis for CQs under DP is that standard notions of optimality are either useless or unattainable. It is well known that the *global sensitivity GS*, which is defined as the maximum difference between the query

PODS '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9260-0/22/06…$15.00
https://doi.org/10.1145/3517804.3524143

answers on *any* two neighboring instances, is worst-case optimal. Note that GS is an instance-independent sensitivity measure. However, for any nontrivial CQ involving two or more relations, $GS = \infty$ in strict DP, as changing one tuple may affect an unbounded number of query results. In relaxed DP, where it is safe to use N in designing the mechanism, GS can be bounded, but can still be as high as $O(N^{n-1})$, where n is the number of relations in q.

For queries where GS is unbounded or too high, a natural idea is to use an instance-specific sensitivity measure [23, 29, 37]. The *local sensitivity* (definition given in Section 2.3) $LS(\mathbf{I})$ is the most natural choice, and it can be much smaller than GS on most real-world instances. However, Nissim et al. [29] point out that using $LS(\mathbf{I})$ to calibrate noise violates DP; instead, they propose *smooth sensitivity* $SS(\mathbf{I})$, which is always higher than $LS(\mathbf{I})$ but lower than GS, and show that it satisfies DP.

How do we quantify the utility of a DP mechanism that adds instance-specific noise? Ideally, we would like it to be *instance-optimal* [18]. More formally, a DP mechanism $\mathcal{M}(\cdot)$ is said to be *c-instance-optimal* if

$$\mathrm{Err}(\mathcal{M}, \mathbf{I}) \le c \cdot \mathrm{Err}(\mathcal{M}', \mathbf{I}),$$

for every instance $\mathbf{I}$ and every DP mechanism $\mathcal{M}'(\cdot)$.

Unfortunately, as pointed out by Asi and Duchi [4], instance-optimal DP mechanisms do not exist, unless the query is trivial (i.e., it returns the same count on all instances), because for an $\mathbf{I}$, one can always design a trivial DP mechanism $\mathcal{M}'(\cdot) \equiv |q(\mathbf{I})|$. It works perfectly on $\mathbf{I}$ but fails miserably on other instances. Yet, such a trivial $\mathcal{M}'$ rules out instance-optimal mechanisms.

1.2 Neighborhood-optimal DP Mechanisms

To eliminate such a trivial $\mathcal{M}'$, Asi and Duchi [4] propose the following natural relaxation of instance-optimality, which requires $\mathcal{M}'$ to not just work well for $\mathbf{I}$, but also in its neighborhood.

Definition 1.1 ((r, c)-neighborhood optimal DP mechanisms). An ε-DP mechanism $\mathcal{M}(\cdot)$ is said to be *(r, c)-neighborhood optimal* if for any instance $\mathbf{I}$ and any ε-DP mechanism $\mathcal{M}'(\cdot)$, there exists an instance $\mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') \le r$ such that

$$\mathrm{Err}(\mathcal{M}, \mathbf{I}) \le c \cdot \mathrm{Err}(\mathcal{M}', \mathbf{I}'). \tag{2}$$

Note that neighborhood optimality smoothly interpolates between instance optimality ($r = 0$) and worst-case optimality ($r = N$). It is also called *local minimax optimality* in [4], as it minimizes (up to a factor of c) the maximum error in the local neighborhood of $\mathbf{I}$.

We adopt the convention of *data complexity*, i.e., all asymptotic notations suppress dependencies on the query size. We also take ε to be a constant, as with most work on differential privacy. Thus, when we say that an ε-DP mechanism is $(O(1), O(1))$-neighborhood optimal, this means that r and c may depend on q and ε, but are independent of N. As we are most interested in constant-factor approximations, we often use "r-neighborhood optimal" as a shorthand of "$(r, O(1))$-neighborhood optimal".

While more relaxed than instance optimality, neighborhood optimality is still not easy to achieve (for small r). For example, we can show that the Median query (i.e., returning the median of N elements in $[0, 1]$, for odd N) does not have any (r, c)-neighborhood optimal mechanisms for $r \le \lfloor N/2 \rfloor$ and any c. To see this, consider $\mathbf{I} = (0, \ldots, 0)$ and $\mathcal{M}'(\cdot) \equiv 0$. Note that all instances in the r-neighborhood of $\mathbf{I}$ have output 0, so $\mathrm{Err}(\mathcal{M}', \mathbf{I}') = 0$ for all $\mathbf{I}'$ with $d(\mathbf{I}, \mathbf{I}') \le r$. Thus, we must set $\mathcal{M}(\mathbf{I}) = 0$ to satisfy (2). We can apply the same argument on $\mathbf{I}'' = (1, \ldots, 1)$ and conclude that $\mathcal{M}(\mathbf{I}'') = 1$. We have thus found two instances on which $\mathcal{M}(\cdot)$ returns different deterministic values, thus $\mathcal{M}$ cannot be DP by a standard argument [17].

Acute readers would realize that the negative result for Median relies on the fact that there are certain instances (like $(0, \ldots, 0)$) with a "flat" neighborhood, i.e., the query output is the same within the neighborhood. Fortunately, for full CQs, which are the focus of this paper, these flat neighborhoods do not exist. This is because in any instance $\mathbf{I}$, one can always add/remove a constant number (depending on q) of tuples to change $|q(\mathbf{I})|$. However, this is merely a necessary condition for a query to admit neighborhood-optimal DP mechanisms. To actually design such a mechanism $\mathcal{M}(\cdot)$, we need an upper bound on $\mathrm{Err}(\mathcal{M}, \mathbf{I})$, as well as a neighborhood lower bound on $\min_{\mathcal{M}'} \max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \le r} \mathrm{Err}(\mathcal{M}', \mathbf{I}')$. Note that both the upper bound and the lower bound are instance specific (i.e., functions of $\mathbf{I}$), and the worst-case (over all $\mathbf{I}$) gap between the upper and lower bounds would become the optimality ratio c.

1.3 DP Mechanisms for Full CQs

We relate both the upper and the lower bounds to the smooth sensitivity $SS(\cdot)$. On the lower bound side, in Section 4, we show that $SS(\cdot)$ is an $O(1)$-neighborhood lower bound. Thus, the smooth sensitivity based mechanism [29] is $O(1)$-neighborhood optimal. However, $SS(\cdot)$ in general requires exponential time to compute. In Section 3, we extend the *residual sensitivity* $RS(\cdot)$ [13] to arbitrary full CQs, and show that it (1) satisfies DP, (2) is a constant-factor upper bound of $SS(\cdot)$, and (3) can be computed in polynomial time. Together with the lower bound, this yields our first main result:

Theorem 1.2. *For any full CQ q, there is an ε-DP mechanism $\mathcal{M}(\cdot)$ that is $O(1)$-neighborhood optimal. For any instance $\mathbf{I}$, $\mathcal{M}(\mathbf{I})$ can be computed in* $\mathrm{poly}(N)$ *time.*

1.4 CQs with Predicates

Next, we consider CQs with predicates. A *predicate* $P(\mathbf{y})$ is a computable function $P : \mathbf{dom}(\mathbf{y}) \to \{\mathsf{True}, \mathsf{False}\}$ for a set of variables $\mathbf{y}$. A valuation of $\mathbf{y}$ is a valid query result only if $P(\mathbf{y})$ evaluates to True.

Predicates are important for expressing graph pattern counting queries. Suppose we would like to count the number of length-3 paths (no repeated vertices) in a directed graph whose edges are stored in a relation Edge. However, the CQ $\mathrm{Edge}(x_1, x_2) \bowtie \mathrm{Edge}(x_2, x_3) \bowtie \mathrm{Edge}(x_3, x_4)$ would not just count the number of length-3 paths, but also length-1 paths, length-2 paths, and triangles. We have to equip the CQ with inequalities, i.e., $x_i \ne x_j$ for all $i \ne j$, to exclude these false positives. Another type of predicates are comparisons, i.e., $x_i \le x_j$ or $x_i < x_j$, which are common in queries over spatiotemporal databases.

In Section 5 we show how to modify $RS(\cdot)$ to take these predicates into consideration, while preserving its neighborhood optimality. The idea is conceptually easy: we just materialize each predicate $P(\mathbf{y})$ into a relation $\{t \in \mathbf{dom}(\mathbf{y}) \mid P(t)\}$, and then apply our DP mechanism in Theorem 1.2. However, this poses a computational issue, since this relation can be infinite (assuming infinite

domains). To address this issue, we show that it is actually possible to compute $RS(\cdot)$ without materializing the $P(\mathbf{y})$'s.

THEOREM 1.3. *For any full CQ q with predicates $P_1(\mathbf{y}_1), \ldots, P_\kappa(\mathbf{y}_\kappa)$, there is an $O(1)$-neighborhood optimal ε-DP mechanism $\mathcal{M}(\cdot)$, if for any $\mathbf{z} \subseteq var(q)$, the satisfiability of $\varphi_1 \wedge \cdots \wedge \varphi_S$ is decidable, where each φ_i is $P_j(\mathbf{u}_i)$ for any j and $\mathbf{u}_i$ is $\mathbf{y}_j$ after replacing all variables not in $\mathbf{z}$ by any constants. Furthermore, $\mathcal{M}(\mathbf{I})$ can be computed in* $\text{poly}(N)$ *time if all predicates are inequalities or comparisons.*

1.5 Non-full CQs

To complete the picture, we finally study non-full CQs in Section 6. Prior work on non-full CQs simply ignores the projection, and uses the sensitivity of the full CQ to calibrate noise. We show how to extend $RS(\cdot)$ to handle the projection more effectively so as to reduce the noise. Unfortunately, our lower bound no longer holds for non-full CQs, hence losing neighborhood optimality. However, we show that this is unavoidable. In particular, even for the simple non-full query $\pi_{x_2}(R_1(x_1) \bowtie R_2(x_1, x_2))$, we show that it does not admit any $o(\sqrt{N})$-neighborhood optimal DP mechanisms.

1.6 Related Work

The notion of neighborhood optimality has been recently proposed in the machine learning community [4]. However, their focus is on high-dimensional queries whose output domain must be continuous. In particular, their lower bound does not hold for any query that returns discrete outputs, in particular, counting queries. Our lower bound (Section 4.1) holds for arbitrary 1-dimensional queries (with either continuous or discrete outputs), and is also better than their lower bound. On the upper bound side, their algorithm in general requires exponential time to compute, although they show how it can be made to run in polynomial time for some machine learning problems like mean-estimation and PCA.

Our work builds upon smooth sensitivity $SS(\cdot)$ [29] and residual sensitivity $RS(\cdot)$ [13]. Although $SS(\cdot)$ has been shown to preserve DP in [29], two issues remain: whether it is optimal (in any sense) and whether it can be computed in polynomial time. We provide positive answers to both questions in the context of CQs. While our optimality of $SS(\cdot)$ is the first of its kind, the computational issue has been addressed for some specific queries, such as MEDIAN (but $SS(\cdot)$ is not neighborhood-optimal for MEDIAN!), triangle counting [29], and t-star counting [23], the latter two being special cases of CQs (with inequalities). However, it is still an open problem whether $SS(\cdot)$ can be computed in polynomial time for an arbitrary CQ in terms of data complexity. To dodge the computational difficulty, elastic sensitivity $ES(\cdot)$ [22] was introduced as a replacement. However, in Appendix C we show that $ES(\cdot)$ is not even worst-case optimal. Residual sensitivity $RS(\cdot)$ was proposed in [13], and it has been shown to be a constant-factor approximation of $SS(\cdot)$ while being polynomially computable. However, $RS(\cdot)$ as defined in [13] does not allow self-joins, which are challenging under DP, since changing one tuple corresponds to changing multiple logical relations when there are self-joins. This can be tricky for both the privacy guarantee and the mechanism's optimality. Second, no lower bound was provided in [13], so the optimality of $RS(\cdot)$ is not known until this paper. Thirdly, we extend $RS(\cdot)$ to CQs with predicates and non-full CQs.

The DP policy considered in this paper treats a change as the insertion, deletion, or substitution of a tuple, hence also called *tuple-DP*. In *user-DP* [26, 34], there is a designated *primary private relation* $R_P \in \mathbf{R}$ that contains all the users, while tuples in other relations with a foreign key (FK) referencing the primary key (PK) of a tuple (user) $t_P \in \mathbf{I}(R_P)$, directly or indirectly, are considered as data belonging to t_P. Two instances $\mathbf{I}$ and $\mathbf{I}'$ are considered neighbors if $\mathbf{I}'$ can be obtained from $\mathbf{I}$ by adding/deleting/substituting a set of tuples, all of which reference the same user $t_P \in \mathbf{I}(R_P)$. Applying user-DP and tuple-DP on the graph schema $\mathbf{R} = \{\texttt{Node(ID)}, \texttt{Edge(src, dst)}\}$, where `src` and `dst` are both FKs referencing `ID`, yields the two well-known DP policies for graph data: node-DP [24] and edge-DP [23]. By designating `Node` as the primary private relation, user-DP on $\mathbf{R}$ becomes node-DP; by setting $\mathbf{R}_P = \{\texttt{Edge}\}$, tuple-DP becomes edge-DP (note that in tuple-DP, we ignore the FK constraints, which effectively means that `Node` is irrelevant). User-DP is more general than tuple-DP, hence potentially incurring a higher privacy cost. Very recently, a logarithmic-neighborhood optimal mechanism has been proposed for CQs under user-DP [12]. Meanwhile, it has also been shown that $O(1)$-neighborhood optimality is not achievable under user-DP [14] even for the simple query $R_1(x_1) \bowtie R_2(x_1, x_2)$ where R_1 is the primary private relation and $R_2.x_1$ is an FK referencing $R_1.x_1$. Therefore, combined with the result obtained in this paper, we now have a separation for CQs under tuple-DP and user-DP.

2 PRELIMINARIES

2.1 Conjunctive Queries

Let $\mathbf{R}$ be a database schema. A *full conjunctive query (CQ)* has the form

$$q := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n),$$

where $R_1, \ldots, R_n$ are relation names in $\mathbf{R}$, and each $\mathbf{x}_i$ is a set of $arity(R_i)$ variables/attributes[1]. We call each $R_i(\mathbf{x}_i)$ an *atom*. We use $[n]$ to denote $\{1, \ldots, n\}$, and $[i, j] = \{i, \ldots, j\}$. For any $E \subseteq [n]$, $\bar{E} = [n] - E$. For a variable x, we use $\mathbf{dom}(x)$ to denote the domain of x. For $\mathbf{x} = (x_1, \ldots, x_k)$, let $\mathbf{dom}(\mathbf{x}) = \mathbf{dom}(x_1) \times \cdots \times \mathbf{dom}(x_k)$. Let $var(q)$ denote the set of variables in q.

When considering self-joins, there can be repeats, i.e., $R_i = R_j$. In this case, we assume $\mathbf{x}_i \neq \mathbf{x}_j$; otherwise one of the two atoms is redundant. Let $\mathbf{I}$ be a database instance over $\mathbf{R}$. For a relation name $R \in \mathbf{R}$, let $\mathbf{I}(R)$ denote the relation instance of R. We use I_i as a shorthand for $\mathbf{I}(R_i)$. $\mathbf{I}$ and the I_i's are called *physical instances*. On the other hand, we use $I_i(\mathbf{x}_i)$ to denote I_i after renaming its attributes to $\mathbf{x}_i$. The $I_i(\mathbf{x}_i)$'s are called the *logical instances*. Note that if R_i and R_j are the same relation name, then $I_i = I_j$, but $I_i(\mathbf{x}_i) \neq I_j(\mathbf{x}_j)$, as $I_i(\mathbf{x}_i)$ and $I_j(\mathbf{x}_j)$ have different attributes. For a self-join-free query, we may without loss of generality assume that $\mathbf{x}_i = sort(R_i)$ for all $i \in [n]$ so the logical instances are the same as the physical instances, but for queries with self-joins, one physical relation instance may correspond to multiple logical relation instances. This distinction makes the problem more difficult under DP, as the distance between two logical instances may be larger than between the physical instances.

[1] If $\mathbf{x}_i$ has constants, we can preprocess $R_i(\mathbf{x}_i)$ in linear time so that only tuples that match these constants remain.

By rearranging the atoms, we may assume that all appearances of the same relation name are consecutive. Suppose m distinct relation names are mentioned in q, and for $i = 1, \dots, m$, $R_{l_i}, \dots, R_{l_{i+1}-1}$ are the same relation name (set $l_{m+1} = n+1$). Let $D_i = [l_i, l_{i+1} - 1]$ and $n_i = l_{i+1} - l_i$, which is the number of copies of R_{l_i} mentioned in q.

2.2 Differential Privacy in Relational Databases

Differential privacy is already defined in (1). This notion can be applied to any problem by properly defining the distance function $d(\cdot, \cdot)$. As a database may contain both public and private relations, we use a more refined definition of $d(\cdot, \cdot)$. For two relation instances over the same relation name I, I', we use $d(I, I')$ to denote the distance between I and I', which is the minimum number of steps to change I into I'. Note that $d(I_j, I'_j)$ is the same for all $j \in D_i$, for any $i \in [m]$, as they are the same physical relation instance.

We use $P^m \subseteq [m]$ to denote the set of private physical relations, while $P^n = \cup_{i \in P^m} D_i$ is the set of private logical relations. Let $m_P = |P^m|$, $n_P = |P^n|$. Two database instances can only differ in the private relations, i.e., $d(I_j, I'_j) = 0$ for every $j \in \bar{P}^n$. In the DP definition (1), we must use the distance between the physical database instances, i.e., $d(\mathbf{I}, \mathbf{I}') = \sum_{i \in [m]} d(I_{l_i}, I'_{l_i})$. Note that the distance between the logical instances, namely $\sum_{i \in [n]} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i))$, can be larger than $d(\mathbf{I}, \mathbf{I}')$ when self-joins are present.

A simple but important observation is that a query with self-joins on instance $\{I_i\}_i$ can be considered as a query without self-joins on instance $\{I_i(\mathbf{x}_i)\}_i$. This allows us to reuse some of the technical results from [13] on self-join-free queries. However, the critical difference is that this conversion enlarges the distance, while the DP guarantee must hold with respect to the distance on the original, physical instance.

2.3 Sensitivity-based DP Mechanisms

As mentioned in Section 1, the most common technique for designing DP mechanisms is to first compute some measure of sensitivity $S(\cdot)$ of the query, and then add noise drawn from a certain distribution calibrated to $S(\cdot)$.

The *local sensitivity* of q at instance $\mathbf{I}$ is how much $|q(\mathbf{I})|$ can change if we change one tuple in $\mathbf{I}$, i.e.,

$$LS(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I}, \mathbf{I}')=1} \big||q(\mathbf{I})| - |q(\mathbf{I}')|\big|. \tag{3}$$

However, using $LS(\cdot)$ directly breaches privacy [29], because two neighboring instances may have very different $LS(\cdot)$. On the other hand, it is safe to use the *global sensitivity*, which is the maximum $LS(\cdot)$ over all instances:

$$GS = \max_{\mathbf{I}} LS(\mathbf{I}).$$

It is well known that one can achieve ε-DP with $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = O(GS)$ by drawing noise from the Laplace distribution calibrated to GS/ε [16]. However, this loses the utility: the GS for CQs can be as large as $O(N^{n_P - 1})$ under relaxed DP, and ∞ under strict DP.

To address this issue, Nissim et al. [29] propose the *smooth sensitivity* $SS_\beta(\cdot)$. To define $SS_\beta(\cdot)$, we first define the *local sensitivity at distance* k:

$$LS^{(k)}(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I}, \mathbf{I}') \le k} LS(\mathbf{I}'). \tag{4}$$

Note that for CQs, (4) can be rewritten as

$$LS^{(k)}(\mathbf{I}) = \max_{\mathbf{I}', d(\mathbf{I}, \mathbf{I}')=k} LS(\mathbf{I}'), \tag{5}$$

because if $d(\mathbf{I}, \mathbf{I}') < k$, we can always insert dummy tuples so that $d(\mathbf{I}, \mathbf{I}') = k$. Then for a parameter β, the smooth sensitivity is

$$SS_\beta(\mathbf{I}) = \max_{k \ge 0} e^{-\beta k} LS^{(k)}(\mathbf{I}). \tag{6}$$

It is clear that $SS_\beta(\mathbf{I}) \le GS$. More importantly, $SS_\beta(\cdot)$ is "smooth", in the sense that $SS_\beta(\mathbf{I}) \le e^\beta SS_\beta(\mathbf{I}')$ for any two neighboring instances $\mathbf{I}$ and $\mathbf{I}'$. Due to its smoothness, it has been shown that setting $\beta = \varepsilon/10$ and drawing noise calibrated to $SS_\beta(\mathbf{I})/\beta$ from a general Cauchy distribution, which has pdf $h(z) \propto \frac{1}{1+|z|^4}$, achieves ε-DP with $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = \frac{SS_\beta(\mathbf{I})}{\beta} = O(SS_\beta(\mathbf{I}))$. The choice of the constant 10 is arbitrary, which affects the tail properties of the noise distribution, but not the variance (asymptotically). In the rest of the paper, we omit the subscript β from $SS_\beta(\cdot)$ for brevity.

However, computing $SS(\mathbf{I})$ is very costly. The definition (6) does not yield an efficient (or even computable) algorithm. To address this issue, Nissim et al. [29] show that any smooth upper bound of $SS(\cdot)$ can be used. Specifically, let

$$\hat{SS}(\mathbf{I}) = \max_{k \ge 0} e^{-\beta k} \hat{LS}^{(k)}(\mathbf{I}). \tag{7}$$

It has been shown that as long as $\hat{LS}^{(k)}(\cdot)$ is an upper bound of $LS^{(k)}(\cdot)$ and satisfies the smoothness property, i.e., for any neighbors $\mathbf{I}$ and $\mathbf{I}'$,

$$\hat{LS}^{(k)}(\mathbf{I}) \le \hat{LS}^{(k+1)}(\mathbf{I}'), \tag{8}$$

then one can use $\hat{SS}(\cdot)$ in place of $SS(\cdot)$ to calibrate noise while preserving ε-DP. The error becomes $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = \frac{10\hat{SS}(\mathbf{I})}{\varepsilon}$ accordingly.

3 RESIDUAL SENSITIVITY FOR FULL CQs

3.1 Residual Queries

The residual sensitivity $RS(\cdot)$ can be considered as an instantiation of $\hat{SS}(\cdot)$ for CQs, i.e., it is defined as in (7) with a particular choice of $\hat{LS}^{(k)}(\cdot)$ that has the smoothness property (8). Our $\hat{LS}^{(k)}(\cdot)$ is based on the *residual queries* of a given CQ q.

A *residual query* of q on a subset $E \subseteq [n]$ of relations is $q_E := \bowtie_{i \in E} R_i(\mathbf{x}_i)$. Its *boundary*, denoted ∂q_E, is the set of attributes that belong to atoms both in and out of E, i.e., $\partial q_E = \{x \mid x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\}$. For a *residual* query q_E on database instance $\mathbf{I}$, its *maximum multiplicity over the boundary* is defined as

$$T_E(\mathbf{I}) = \max_{t \in \mathbf{dom}(\partial q_E)} |q_E(\mathbf{I}) \bowtie t|.$$

A *witness tuple* of the maximum multiplicity over the boundary of $q_E(\mathbf{I})$ is

$$t_E(\mathbf{I}) = \arg\max_{t \in \mathbf{dom}(\partial q_E)} |q_E(\mathbf{I}) \bowtie t|. \tag{9}$$

Per convention, when $E = \emptyset$, we set $q_E(\mathbf{I}) = \{\langle\rangle\}$, where $\langle\rangle$ denotes the empty tuple, so $T_\emptyset(\mathbf{I}) = 1$.

We note that $T_E(\mathbf{I})$ is exactly an AJAR/FAQ query [3, 21], where the annotations of all tuples are 1 with two semiring aggregations + and max. The + aggregation is done group-by ∂q, followed by a max over all the + aggregates. Such a query can be computed in

$O(N^w)$ time, where w is its AJAR/FAQ width, a constant depending on the query only.

To develop $RS(\cdot)$, we need some properties of $T_E(\cdot)$. We start with a simple one:

Lemma 3.1. *Given any CQ q, $E \subseteq [n]$, and two database instances $\mathbf{I}, \mathbf{I}'$, if $I_i(\mathbf{x}_i) \subseteq I'_i(\mathbf{x}_i)$ for all $i \in E$, then $T_E(\mathbf{I}) \le T_E(\mathbf{I}')$. In particular, if $I_i(\mathbf{x}_i) = I'_i(\mathbf{x}_i)$ for all $i \in E$, then $T_E(\mathbf{I}) = T_E(\mathbf{I}')$.*

As will become clear, the sensitivity of q depends on the sensitivity of $T_E(\cdot)$. The following technical lemma provides such an upper bound (proofs of all lemmas and theorems are provided in [15].):

Lemma 3.2. *For any CQ q, $E \subseteq [n]$, and two instances $\mathbf{I}, \mathbf{I}'$,*

$$|T_E(\mathbf{I}) - T_E(\mathbf{I}')| \le \sum_{E' \subseteq E, E' \neq \emptyset} \left(T_{E-E'}(\mathbf{I}) \prod_{i \in E'} d(I_i(\mathbf{x}_i), I'_i(\mathbf{x}_i)) \right).$$

3.2 Local Sensitivity of CQs

We first consider the local sensitivity $LS(\cdot)$. For a CQ without self-joins, its local sensitivity is precisely characterized by the $T_E(\cdot)$'s.

Lemma 3.3 ([13]). *For a CQ without self-joins,*

$$LS(\mathbf{I}) = \max_{i \in P^n} T_{\overline{\{i\}}}(\mathbf{I}).$$

To extend this result to CQs with self-joins, we need to bound how much $|q(\mathbf{I})|$ can change when multiple relations change simultaneously, as a change in one physical relation instance may correspond to changes in multiple logical relations when self-joins are present. We first consider the case for self-join-free queries.

Lemma 3.4. *Let q be a CQ q without self-joins, $B \subseteq [n]$, $B \neq \emptyset$, and let $\mathbf{I}, \mathbf{I}'$ be instances such that $d(I_j, I'_j) = 1$ for all $j \in B$ while $d(I_j, I'_j) = 0$ otherwise. Then*

$$\big||q(\mathbf{I})| - |q(\mathbf{I}')|\big| \le \sum_{E \subseteq B, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}).$$

Based on this and (3), we can derive an upper bound on $LS(\mathbf{I})$ for CQs with self-joins.

Theorem 3.5. *For a CQ q,*

$$LS(\mathbf{I}) \le \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}).$$

Remark. Note that when self-joins are present, we can no longer obtain an exact formula for $LS(\mathbf{I})$ like for self-join-free queries in Lemma 3.3. This is precisely due to the fact that self-joins induce changes in multiple logical relations that may interact in complex manners.

3.3 Global Sensitivity of CQs

Because $GS = \max_{\mathbf{I}} LS(\mathbf{I})$, a by-product of Theorem 3.5 is an upper bound on GS under relaxed DP where the instance size N is public. This upper bound can be much smaller than the trivial upper bound $O(N^{n_P - 1})$ mentioned in Section 2.3 for many CQs.

By Theorem 3.5, we have

$$GS \le \max_{\mathbf{I}} \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} T_{\bar{E}}(\mathbf{I}) \le \max_{i \in P^m} \sum_{E \subseteq D_i, E \neq \emptyset} \max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}). \quad (10)$$

Observe that $\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I})$ is upper bounded by the maximum join size of the residual query $q_{\bar{E}}$, when the logical relations of the same physical relation are allowed to be instantiated differently and the domain size of each variable in $\partial q_E = \partial q_{\bar{E}}$ is set to 1, which is equivalent to removing these variables. We can bound the maximum join size using the *AGM bound* [5]. Together with (10) this yields an upper bound on GS.

Example 3.6. We illustrate how this is done on the triangle query $q = \mathtt{Edge}(x_1, x_2) \bowtie \mathtt{Edge}(x_2, x_3) \bowtie \mathtt{Edge}(x_1, x_3)$ on a single physical relation Edge. For $E = \{3\}$, i.e., $q_{\bar{E}} = \mathtt{Edge}(x_1, x_2) \bowtie \mathtt{Edge}(x_2, x_3)$ and $\partial q_E = \{x_1, x_3\}$, we have

$$\begin{aligned}
\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}) &= \max_{\mathbf{I}} \max_{t \in \mathbf{dom}(x_1, x_3)} |\mathtt{Edge}(x_1, x_2) \bowtie \mathtt{Edge}(x_2, x_3)| \\
&\le \max_{\mathbf{I}} \max_{t \in \mathbf{dom}(x_1, x_3)} |\mathtt{Edge}_1(x_1, x_2) \bowtie \mathtt{Edge}_2(x_2, x_3)| \\
&= \max_{\mathbf{I}} (\mathtt{Edge}_1(x_2) \bowtie \mathtt{Edge}_2(x_2)). \\
&= \mathsf{AGM}(\mathtt{Edge}_1(x_2) \bowtie \mathtt{Edge}_2(x_2)).
\end{aligned}$$

We can similarly derive a bound for other E's. Note that when E consists of 2 relations, $\max_{\mathbf{I}} T_{\bar{E}}(\mathbf{I}) \le 1$. Thus,

$$\begin{aligned}
GS \le& \mathsf{AGM}(\mathtt{Edge}_1(x_2) \bowtie \mathtt{Edge}_2(x_2)) \\
&+ \mathsf{AGM}(\mathtt{Edge}_2(x_3) \bowtie \mathtt{Edge}_3(x_3)) \\
&+ \cdots \\
=& O(N).
\end{aligned}$$

Another example using the path-4 query is given in Appendix A. Furthermore, any join size upper bound can be plugged into (10). For example, when degree information or functional dependencies are publicly available, tighter upper bounds can be derived [2, 19]. Although DP mechanisms based on GS are not as accurate as our $RS(\cdot)$-based mechanisms to be presented next, they can be computed in $O(1)$ time (excluding the time for computing $|q(\mathbf{I})|$).

3.4 Deriving $\hat{LS}^{(k)}(\cdot)$

Theorem 3.5 has provided an upper bound for $LS(\mathbf{I})$. The next step is to derive $\hat{LS}^{(k)}(\mathbf{I})$, an upper bound for $LS^{(k)}(\mathbf{I})$.

For any two instances $\mathbf{I}, \mathbf{I}'$, define their *distance vector* as

$$\mathbf{s} = (d(I_1, I'_1), d(I_2, I'_2), \ldots, d(I_n, I'_n)).$$

For any $\mathbf{s} = (s_1, \ldots, s_n)$, let $\mathcal{I}^{\mathbf{s}} = \{\mathbf{I}' : d(I_j, I'_j) = s_j, j \in [n]\}$ be the set of instances whose distance vectors are $\mathbf{s}$ from $\mathbf{I}$. Note that when self-joins are present, not any $\mathbf{s} \in \mathbb{N}^n$ is a valid distance vector. We must ensure $s_{l_i} = s_{l_i+1} = \cdots = s_{l_i+n_i-1}$, for any $i \in [m]$. Let $\mathbf{S}^k$ be the set of valid distance vectors such that the total distance of all private relations is k, i.e.,

$$\mathbf{S}^k = \left\{ \mathbf{s} : \sum_{i \in P^m} s_{l_i} = k; s_j = 0, j \in \bar{P}^n; \forall i \in [m], j \in D_i, s_j = s_{l_i} \right\}.$$

Denote the set of instances at k distance from $\mathbf{I}$ as $\mathcal{I}^k = \{\mathbf{I}' : d(\mathbf{I}, \mathbf{I}') = k\}$, i.e.,

$$\mathcal{I}^k = \cup_{\mathbf{s} \in \mathbf{S}^k} \mathcal{I}^{\mathbf{s}}.$$

We now derive an upper bound of $LS^{(k)}(\cdot)$ in terms of $T_E(\cdot)$.

Lemma 3.7.

$$LS^{(k)}(\mathbf{I}) \le \max_{\mathbf{s}\in\mathbb{S}^k}\max_{i\in P^m}\sum_{E\subseteq D_i, E\neq\emptyset}\max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} T_{\bar{E}}(\mathbf{I}').$$

Let $\hat{T}_{E,\mathbf{s}}(\mathbf{I})$ be an upper bound of $\max_{\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}} T_E(\mathbf{I}')$. Then

$$\hat{LS}^{(k)}(\mathbf{I}) := \max_{\mathbf{s}\in\mathbb{S}^k}\max_{i\in P^m}\sum_{E\subseteq D_i, E\neq\emptyset}\hat{T}_{\bar{E},\mathbf{s}}(\mathbf{I}'), \tag{11}$$

is clearly an upper bound of $LS^{(k)}(\mathbf{I})$.

Now, it remains to find a valid $\hat{T}_{E,\mathbf{s}}(\mathbf{I})$. By Lemma 3.2, we have for any $E\subseteq[n]$ and any $\mathbf{I}'\in\mathcal{I}^{\mathbf{s}}$,

$$T_E(\mathbf{I}') \le T_E(\mathbf{I}) + \sum_{E'\subseteq\bar{E}, E'\neq\emptyset}\left(T_{E-E'}(\mathbf{I})\prod_{j\in E'}s_j\right).$$

So we set $\hat{T}_{E,\mathbf{s}}(\mathbf{I})$ as

$$\hat{T}_{E,\mathbf{s}}(\mathbf{I}) := \sum_{E'\subseteq E}\left(T_{E-E'}(\mathbf{I})\prod_{j\in E'}s_j\right), \tag{12}$$

where we define $\prod_{j\in\emptyset}s_j = 1$.

Finally, the residual sensitivity is defined as in (7) by setting $\hat{LS}^{(k)}(\mathbf{I})$ as in (11):

$$RS(\mathbf{I}) = \max_{k\ge 0} e^{-\beta k}\hat{LS}^{(k)}(\mathbf{I}). \tag{13}$$

The following theorem shows that $\hat{LS}^{(k)}(\cdot)$ is smooth, so $RS(\cdot)$ satisfies ε-DP.

Theorem 3.8. *For any CQ and any $\mathbf{I},\mathbf{I}'$ such that $d(\mathbf{I},\mathbf{I}')=1$, $\hat{LS}^{(k)}(\mathbf{I}) \le \hat{LS}^{(k+1)}(\mathbf{I}')$ for any $k\ge 0$.*

3.5 Computing $RS(\cdot)$

Recall that for any given k, $\hat{LS}^{(k)}(\mathbf{I})$ can be computed in polynomial time since each $T_{\bar{E}}(\mathbf{I})$ is an AJAR/FAQ query [3, 21]. The last missing piece is to bound the range of k that one has to consider when computing $RS(\mathbf{I})$ using (13). The following lemma implies that we only need to consider $k = 0,\dots,\hat{k} = O(1)$ when computing $RS(\mathbf{I})$.

Lemma 3.9. *For any $k \ge \hat{k} = \frac{m_p}{1-\exp(-\beta/\max_{i\in[m]} n_i)}$,*

$$e^{-\beta k}\hat{LS}^{(k)}(\mathbf{I}) \le e^{-\beta(k-1)}\hat{LS}^{(k-1)}(\mathbf{I}).$$

Remark. In actual implementation, we first compute $T_{\bar{E}}(\mathbf{I})$ for all $E\subseteq D_i, E\neq\emptyset$. After that, it only takes $O(1)$ time to compute $RS(\mathbf{I})$ using formulas (11), (12), and (13). Thus, the concrete computational complexity of $RS(\cdot)$ for a CQ q is $O(N^{w_{\max}})$, where $w_{\max}$ is the maximum AJAR/FAQ width [3, 21] of the residual queries of q.

4 NEIGHBORHOOD OPTIMALITY

In this section we prove Theorem 1.2. This is done in three steps: We first derive a sufficient condition for $SS(\cdot)$ to be an r-neighborhood lower bound. Next, we show that this condition holds for full CQs with $r = O(1)$. Finally, we show that $RS(\cdot)$ is at most a constant factor larger than $SS(\cdot)$,

4.1 General Neighborhood Lower Bounds

We first develop two general neighborhood lower bounds that hold for arbitrary queries (not necessarily CQs), one based on $LS^{(k)}(\cdot)$ while the other based on $SS(\cdot)$. These lower bounds hold for an arbitrary query q with vectored outputs. We start with an observation from [35]:

Lemma 4.1 ([35]). *For any ε-DP mechanism $\mathcal{M}'(\cdot)$ and any instance $\mathbf{I}$, there exists an $\mathbf{I}'$ with $d(\mathbf{I},\mathbf{I}')\le 1$ such that*

$$\Pr\left[\left|\mathcal{M}'(\mathbf{I}') - q(\mathbf{I}')\right| \ge \frac{LS(\mathbf{I})}{2}\right] \ge \frac{1}{1+e^{\varepsilon}}.$$

This implies that $LS(\cdot)$ is an 1-neighborhood lower bound, i.e.,

$$\max_{\mathbf{I}':d(\mathbf{I},\mathbf{I}')\le 1}\mathrm{Err}(\mathcal{M}',\mathbf{I}') \ge \frac{1}{2\sqrt{1+e^{\varepsilon}}}\cdot LS(\mathbf{I}), \tag{14}$$

for any $\mathbf{I}$ and any $\mathcal{M}'$. We generalize this result, showing that $LS^{(r-1)}(\cdot)$ is an r-neighborhood lower bound:

Lemma 4.2. *For any $\mathbf{I}$, any ε-DP mechanism $\mathcal{M}'$, and any $r\ge 1$,*

$$\max_{\mathbf{I}':d(\mathbf{I},\mathbf{I}')\le r}\mathrm{Err}(\mathcal{M}',\mathbf{I}') \ge \frac{1}{2\sqrt{1+e^{\varepsilon}}}\cdot LS^{(r-1)}(\mathbf{I}). \tag{15}$$

Note that (14) is the special case of (15) with $r=1$.

Previously, Asi and Duchi [4] also derive a neighborhood lower bound. In Appendix B, we show that our lower bound is always no worse than theirs for $\varepsilon = O(1)$, while can be polynomially better for certain queries. Furthermore, their lower bound requires a technical condition on q while our lower bound holds for an arbitrary q.

To show that $SS(\cdot)$ is an r-neighborhood lower bound, we need a condition on $LS^{(k)}(\cdot)$, that they do not grow more than exponentially quickly when $k\ge r$.

Lemma 4.3. *Given any $\varepsilon,\beta>0$ and any instance $\mathbf{I}$, if for some $r\ge 1$ (possibly depending on β and $\mathbf{I}$),*

$$LS^{(k)}(\mathbf{I}) \le e^{\beta k}LS^{(r-1)}(\mathbf{I}), \tag{16}$$

for any $k\ge r$, for any ε-DP mechanism $\mathcal{M}'$,

$$\max_{\mathbf{I}':d(\mathbf{I},\mathbf{I}')\le r}\mathrm{Err}(\mathcal{M}',\mathbf{I}') \ge \frac{1}{2\sqrt{1+e^{\varepsilon}}}\cdot SS(\mathbf{I}).$$

Remark 1. Recall from Section 2.3 that β and ε are just a constant-factor apart, so β is also a constant if ε is considered a constant.

Remark 2. The restriction of the growth rate is very mild, except that it also forbids $LS^{(k)}(\cdot)$ to go from zero to nonzero. This is why we impose this restriction only for $k\ge r$. For certain problems like Median, this requires a large r, which is actually unavoidable since a large flat neighborhood will rule out r-neighborhood optimal mechanisms for small r anyway, as we argued in Section 1.

Before considering CQs, as a warm-up, we apply Lemma 4.3 to the triangle counting problem. Here, the instance $\mathbf{I}$ is a simple graph (i.e., no self-loops and multi-edges), and the query q returns the number of triangles in $\mathbf{I}$.

Lemma 4.4. *For the triangle counting problem, the condition in Lemma 4.3 holds with $r = \max\left\{3, \left\lceil 4\frac{\ln(1/\beta)}{\beta}\right\rceil\right\}$.*

Note that the r needed in the lemma above is independent of $\mathbf{I}$. Thus, $SS(\cdot)$ is an $O(1)$-neighborhood lower bound for the triangle counting problem, i.e., the previous SS-based DP-mechanism for triangle counting [29] is $O(1)$-neighborhood optimal. This is the first optimality guarantee for triangle counting under DP, while our main optimality result is a vast generalization of this.

4.2 Neighborhood Lower Bound for CQs

To show that $SS(\cdot)$ is an $O(1)$-neighborhood lower bound for CQs, we need to show that the condition in Lemma 4.3 holds with some constant r. This requires an upper bound on $LS^{(k)}(\cdot)$, as well as a lower bound on $LS^{(r-1)}(\cdot)$. For the upper bound on $LS^{(k)}(\cdot)$, we use the $\hat{LS}^{(k)}(\cdot)$ developed in Section 3.4. For the lower bound, we first consider the case $r = n_P$. Recall that $n_P = |P^n|$ is the number of private logical relations.

Lemma 4.5. *For any CQ, any instance $\mathbf{I}$, and any $E \subseteq P^n$, $E \neq \emptyset$, we have $LS^{(n_P-1)}(\mathbf{I}) \geq T_{\bar{E}}(\mathbf{I})$.*

To prove this lemma, we need to make no more than $n_P - 1$ changes to $\mathbf{I}$ to obtain an $\mathbf{I}'$ that is highly sensitive, i.e., there is one tuple in $\mathbf{I}'$ that is involved in at least $T_{\bar{E}}(\mathbf{I})$ join results, or one tuple not in $\mathbf{I}'$ that would produce at least $T_{\bar{E}}(\mathbf{I})$ join results if inserted. The technical construction of such an $\mathbf{I}'$ is given in [15].

Next, recall from equations (11) and (12) that $\hat{LS}^{(k)}(\mathbf{I})$ is also defined in terms of the $T_{\bar{E}}(\mathbf{I})$'s. Together with Lemma 4.5, this allows us to build a connection between $\hat{LS}^{(k)}(\mathbf{I})$ and $LS^{(n_P-1)}(\mathbf{I})$:

Lemma 4.6. *For any CQ, any instance $\mathbf{I}$, and any $k \geq 1$, we have*

$$\hat{LS}^{(k)}(\mathbf{I}) \leq (4k)^{n_P-1} LS^{(n_P-1)}(\mathbf{I}).$$

Lemma 4.6 almost meets the condition of Lemma 4.3, except that $(4k)^{n_P-1}$ is not necessarily smaller than $e^{\beta k}$. But as the former is a polynomial while the latter is exponential, this is not an issue as long as k is larger than a constant.

Theorem 4.7. *For any CQ q, any $\varepsilon, \beta > 0$, there exist a constant $r > 0$ (depending on q, ε, β) such that for any $\mathbf{I}$ and any ε-DP mechanism $\mathcal{M}'$,*

$$\max_{\mathbf{I}': d(\mathbf{I},\mathbf{I}') \leq r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \geq \frac{1}{2\sqrt{1+e^{\varepsilon}}} \cdot SS(\mathbf{I}).$$

4.3 Optimality of $RS(\cdot)$

To complete the proof of Theorem 1.2, we show that $RS(\cdot)$ is at most a constant-factor larger than $SS(\cdot)$.

Lemma 4.8. *For any CQ and any $\mathbf{I}$, $RS(\mathbf{I}) \leq \left(\frac{4(n_P-1)}{\beta e^{1-\beta}}\right)^{n_P-1} SS(\mathbf{I})$.*

In Appendix C, we give an example on which the elastic sensitivity $ES(\cdot)$ [22], which is the only known DP mechanism for CQs with self-joins, is asymptotically larger than GS. This means that $ES(\cdot)$ is not even worst-case optimal.

5 CQs WITH PREDICATES

A CQ with predicates (CQP) has the form

$$q := \sigma_{P_1(\mathbf{y}_1) \wedge \cdots \wedge P_\kappa(\mathbf{y}_\kappa)}(R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n)),$$

where each $P_j : \mathbf{dom}(\mathbf{y}_j) \to \{\text{True}, \text{False}\}$ is a computable function for some $\mathbf{y}_j \subseteq var(q) = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$. By a slight abuse of notation, we also use $P(\mathbf{y})$ to denote the (possibly infinite) relation $\{t \in \mathbf{dom}(\mathbf{y}) \mid P(t)\}$. This way, a CQP can be written as a normal CQ:

$$q := R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n) \bowtie P_1(\mathbf{y}_1) \bowtie \cdots \bowtie P_\kappa(\mathbf{y}_\kappa). \quad (17)$$

Note that the $P_j(\mathbf{y}_j)$'s are all public, since they only depend on the query and the domain, not on the instance.

The current approach to dealing with a CQP under DP [13, 22, 26] is to evaluate the CQP as given, but compute the sensitivity without considering the predicates. This yields a valid DP mechanism, but loses optimality. To see this, just consider an extreme case where a predicate always returns False. Then the query becomes a trivial query and the optimal (under any notion of optimality) mechanism is $\mathcal{M}(\cdot) \equiv 0$, i.e., $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = 0$ for all $\mathbf{I}$, but the sensitivity of the query without the predicate must be nonzero.

In this section, we show how to extend $RS(\cdot)$ to CQPs while preserving its $O(1)$-neighborhood optimality. The idea is simple, we just consider a CQP as a CQ as defined in (17), so optimality immediately follows from Theorem 1.2. The issue, however, is how to compute $RS(\cdot)$ when some relations are infinite. In Section 5.1 we first give a general algorithm, which may take exponential time, to compute $RS(\cdot)$ for arbitrary predicates under the technical condition of Theorem 1.3; in Section 5.2 we give a polynomial-time algorithm for the case where all the predicates are inequalities or comparisons, which proves the second part of Theorem 1.3.

5.1 General Predicates

The first observation is that, when the $P(\mathbf{y}_j)$'s are arbitrary, no optimal (under any notion of optimality) DP mechanisms for CQPs exist (see Appendix D for a proof). However, the situation is not hopeless. Below we show how to compute $RS(\mathbf{I})$ for any CQP if for any $\mathbf{z} \subseteq var(q)$, the satisfiability of $\varphi_1 \wedge \cdots \wedge \varphi_S$ is decidable, where each φ_i is $P_j(\mathbf{u}_i)$ for any j and $\mathbf{u}_i$ is $\mathbf{y}_j$ after replacing all variables not in $\mathbf{z}$ by any constants. This is a very mild condition; in fact, the entire literature on *constraint satisfaction problems (CSPs)* is devoted to designing efficient algorithms for determining the satisfiability of $\varphi_1 \wedge \cdots \wedge \varphi_S$ when the φ_i's take certain forms, and finding a satisfying valuation for $\mathbf{z}$, if one exists.

It suffices to show how to compute $T_{\bar{E}}(\mathbf{I})$ for any $E \subseteq P^n$. Since all the predicates correspond to public relations, the residual query has the form

$$q_{\bar{E}} = (\bowtie_{i \in \bar{E}} R_i(\mathbf{x}_i)) \bowtie (\bowtie_{j \in [\kappa]} P_j(\mathbf{y}_j)).$$

We split the boundary variables as $\partial q_{\bar{E}} = \partial q^1_{\bar{E}} \cup \partial q^2_{\bar{E}}$, where

$$\partial q^1_{\bar{E}} = \{x \mid x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\},$$

and

$$\partial q^2_{\bar{E}} = \{x \mid x \in \mathbf{x}_i \cap \mathbf{y}_j, i \in E, j \in [\kappa]\} - \partial q^1_{\bar{E}}.$$

Let

$$q^\circ_{\bar{E}} = (\bowtie_{i \in \bar{E}} R_i(\mathbf{x}_i))$$

be the CQ part of $q_{\bar{E}}$.

Example 5.1. Figure 1 illustrates these concepts with the query

$$q = R_1(x_1, x_2, x_3) \bowtie R_2(x_3, x_4, x_5) \bowtie R_3(x_5, x_6, x_7) \bowtie R_4(x_1, x_7, x_8) \bowtie P_1(x_2, x_4) \bowtie P_2(x_2, x_8) \bowtie P_3(x_3, x_7) \bowtie P_4(x_4, x_6),$$

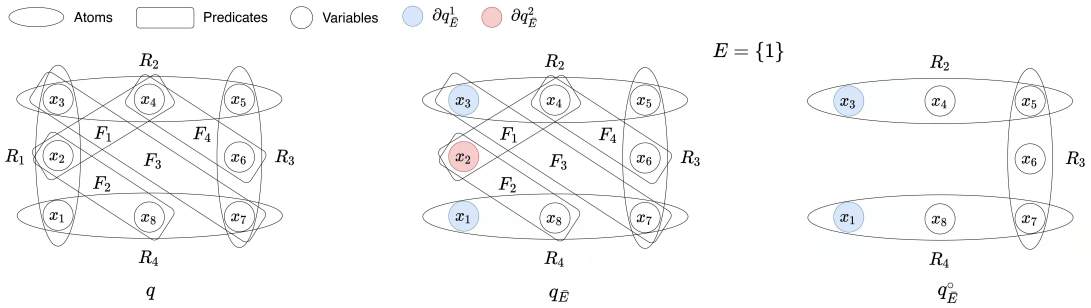

Figure 1: $q_{\bar{E}}$, $q^\circ_{\bar{E}}$, $\partial q^1_{\bar{E}}$ **and** $\partial q^2_{\bar{E}}$.

where we set $E = \{1\}$.

The following observations about the boundary variables are straightforward.

Lemma 5.2. *For any CQP q and any $E \subseteq [n]$,*
(1) $\partial q^2_{\bar{E}} \subseteq \mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa \subseteq \partial q^2_{\bar{E}} \cup var(q^\circ_{\bar{E}})$;
(2) $\partial q^1_{\bar{E}} \subseteq var(q^\circ_{\bar{E}})$;
(3) $var(q^\circ_{\bar{E}}) \cap \partial q^2_{\bar{E}} = \emptyset$.

Now, we look at how to compute $T_{\bar{E}}(\mathbf{I})$:

Lemma 5.3.

$$T_{\bar{E}}(\mathbf{I}) = \max_{\substack{t_1 \in \pi_{\partial q^1_{\bar{E}}} q^\circ_{\bar{E}}(\mathbf{I}) \\ t_2 \in \mathbf{dom}(\partial q^2_{\bar{E}})}} \left| q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2 \right|. \quad (18)$$

Since $|q^\circ_{\bar{E}}(\mathbf{I})|$ is bounded by $O(N^n)$, the choices of t_1 are limited. The difficulty is that $t_2 \in \mathbf{dom}(\partial q^2_{\bar{E}})$ has infinitely many choices. The idea is to flip the problem around. For any $B \subseteq q^\circ_{\bar{E}}(\mathbf{I})$, we check if there exist t_1, t_2 such that

$$|B \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2| = |B|. \quad (19)$$

This is equivalent to checking if $t_B \bowtie t_1 \bowtie t_2$ can pass all predicates for every $t_B \in B$. Since $\partial q^1_{\bar{E}} \subseteq var(q^\circ_{\bar{E}})$, for each $t_B \in B$, t_1 must be $\pi_{\partial q^1_{\bar{E}}} t_B$. Thus the problem boils down to deciding if

$$\bigwedge_{t_B \in B, j\in[\kappa]} P_j(\mathbf{y}_j(t_B)) \quad (20)$$

is satisfiable, where $\mathbf{y}_j(t_B)$ denotes $\mathbf{y}_j$ after replacing its variables by the corresponding constants if they appear in t_B. Note that free variables in (20) are $\mathbf{z} = \partial q^2_{\bar{E}}$. This is precisely the technical condition we impose on the predicates. Finally, we enumerate all B, and return the maximum $|B|$ for which (20) is satisfiable. This proves the first part of Theorem 1.3. However, this algorithm runs in exponential time since there are $2^{|q^\circ_{\bar{E}}(\mathbf{I})|} = 2^{\text{poly}(N)}$ B's that need to be considered.

5.2 Comparison and Inequality Predicates

For CQPs where the predicates are inequalities or comparisons, we may without loss of generality assume that the domain of all attributes in $\mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa$ is $\mathbb{Z}$. We show in this subsection how to reduce the running time of the algorithm to poly(N) in this case. Let $\rho = |\partial q^2_{\bar{E}}|$. Then t_2 takes values from $\mathbb{Z}^\rho$. The key to an efficient algorithm is thus to reduce this domain, and then apply the algorithm in [3, 21].

To reduce the domain of t_2, one natural idea is to only consider the *active domain* [1]. Let $\mathbb{Z}^*(\mathbf{I})$ be the set of integers appearing in $\mathbf{I}$ on attributes $\mathbf{y}_1 \cup \cdots \cup \mathbf{y}_\kappa$, and let $\mathbb{Z}^*(q)$ be the set of integers appearing in the predicates of q. Then the active domain is $\mathbb{Z}^*(q, \mathbf{I}) = \mathbb{Z}^*(q) \cup \mathbb{Z}^*(\mathbf{I}) \cup \{-\infty, \infty\}$. However, only considering $t_2 \in (\mathbb{Z}^*(q, \mathbf{I}))^\rho$ is not enough; Appendix E gives an example showing that $T_{\bar{E}}(\mathbf{I})$ may attain its maximum at some value between two consecutive values in the active domain. Thus, we augment the active domain to $\mathbb{Z}^+(q, \mathbf{I})$, as follows. Let $\mathbb{Z}^*(q, \mathbf{I}, i)$ be the ith elements in $\mathbb{Z}^*(q, \mathbf{I})$ in order. $\mathbb{Z}^+(q, \mathbf{I})$ includes all elements in $\mathbb{Z}^*(q, \mathbf{I}, i)$, plus 2κ arbitrary distinct elements between $\mathbb{Z}^*(q, \mathbf{I}, i)$ and $\mathbb{Z}^*(q, \mathbf{I}, i+1)$ for all $i \in [|\mathbb{Z}^*(q, \mathbf{I})| - 1]$. If there are less than 2κ elements between $\mathbb{Z}^*(q, \mathbf{I}, i)$ and $\mathbb{Z}^*(q, \mathbf{I}, i+1)$, all elements in between are included.

We show that it suffices to use $\mathbb{Z}^+(q, \mathbf{I})$ as the domain of t_2.

Lemma 5.4. *When all the predicates are inequalities and comparisons,*

$$T_{\bar{E}}(\mathbf{I}) = \max_{\substack{t_1 \in \pi_{\partial q^1_{\bar{E}}} q^\circ_{\bar{E}}(\mathbf{I}) \\ t_2 \in (\mathbb{Z}^+(q,\mathbf{I}))^\rho}} \left| q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa]} P_j(\mathbf{y}_j)) \bowtie t_1 \bowtie t_2 \right|. \quad (21)$$

Since $\mathbb{Z}^+(q, \mathbf{I}) = O(N + \kappa) = O(N)$, we can simply materialize each $P_j(\mathbf{y}_j)$ into $\{t \in (\mathbb{Z}^+(q, \mathbf{I}))^2 \mid P_j(t)\}$, which has size $O(N^2)$. Thus, evaluating (21) using the algorithm in [3, 21] also takes polynomial time, and we have concluded the proof of the second part of Theorem 1.3.

As a practical improvement, observe that if a variable $y \in \partial q^2_{\bar{E}}$ is only involved in inequality predicates, then it can always take a value such that all these inequalities hold. Thus, there is no need to materialize these predicates. In particular, we arrive at a simpler formula for computing $T_{\bar{E}}(\mathbf{I})$ when all predicates are inequalities, e.g., graph pattern counting queries.

Corollary 5.5. *For a CQP where all predicates are inequalities,*

$$T_{\bar{E}}(\mathbf{I}) = \max_{t_1 \in \mathbf{dom}(\partial q^1_{\bar{E}})} \left| q^\circ_{\bar{E}}(\mathbf{I}) \bowtie (\bowtie_{j\in[\kappa], \mathbf{y}_j \subseteq var(q^\circ_{\bar{E}})} P_j(\mathbf{y}_j)) \bowtie t_1 \right|.$$

To compute $T_{\bar{E}}(\mathbf{I})$, we compute $q^\circ_{\bar{E}}(\mathbf{I})$, apply all predicates $P_j(\mathbf{y}_j))$ for $j \in [\kappa], \mathbf{y}_j \subseteq var(q^\circ_{\bar{E}}))$, do a count group-by $\partial q^1_{\bar{E}}$, and return the maximum count.

6 NON-FULL CQs

A non-full CQ has the form

$$q := \pi_{\mathbf{o}} \left(R_1(\mathbf{x}_1) \bowtie \cdots \bowtie R_n(\mathbf{x}_n) \right),$$

where $\mathbf{o} \subseteq \mathbf{x}$ denotes the set of output variables.

Similarly, the current approach [13, 22, 26] simply computes the noise ignoring the projection. This performs badly as the projection usually reduces the true count significantly, so the noise becomes relatively much larger. In this section, we show how to add projection into the residual sensitivity framework.

For any $E \subseteq [n]$, define

$$\mathbf{o}_E = \mathbf{o} \cap (\cup_{i \in E} \mathbf{x}_i).$$

Note that $\mathbf{o} = \mathbf{o}_{[n]}$.

Given $E \subseteq [n]$, the residual query with projection is

$$q_E := \pi_{\mathbf{o}_E} (\bowtie_{i \in E} R_i(\mathbf{x}_i)).$$

The boundary variables $\partial q_E = \{x | x \in \mathbf{x}_i \cap \mathbf{x}_j, i \in E, j \in \bar{E}\}$ remain unchanged, but the maximum boundary $T_E(\mathbf{I})$ and witness $t_E(\mathbf{I})$ are modified as

$$T_E(\mathbf{I}) = \max_{t \in \mathbf{dom}(\partial q_E)} |\pi_{\mathbf{o}_E} (\bowtie_{i \in E} I_i(\mathbf{x}_i) \bowtie t)|$$

and

$$t_E(\mathbf{I}) = \arg\max_{t \in \mathbf{dom}(\partial q_E)} |\pi_{\mathbf{o}_E} (\bowtie_{i \in E} I_i(\mathbf{x}_i) \bowtie t)|.$$

If $\mathbf{o}_E = \emptyset$, there is always a $t \in \mathbf{dom}(\partial q_E)$ such that $(\bowtie_{i \in E} I_i(\mathbf{x}_i)) \bowtie t \neq \emptyset$, which becomes $\{\langle\rangle\}$ after the projection, so $T_E(\mathbf{I}) = 1$.

Note that these definitions degenerate into the full-CQ case when $\mathbf{o} = var(q)$.

We as before compute $\hat{LS}^{(k)}(\mathbf{I})$ by (11), (12), and then $RS(\mathbf{I})$ by (13), but using the new definition of $T_E(\mathbf{I})$ with projection. Below we show that $RS(\cdot)$ is still a valid ε-DP mechanism and it can be computed efficiently. Recall that the validity of $RS(\cdot)$ is based on (1) $\hat{LS}^{(k)}(\cdot)$ is an upper bound of $LS^{(k)}(\cdot)$; and (2) $\hat{LS}^{(k)}(\cdot)$ satisfies the smoothness property (8). The first depends on Lemma 3.2 and Theorem 3.5 while the second depends on Lemma 3.2. One can verify that, as long as Lemma 3.2 and Theorem 3.5 hold for non-full CQs, the rest of the validity proof will go through. We thus focus on verifying Lemma 3.2 and Theorem 3.5 on non-full CQs.

Lemma 6.1. *For non-full CQs, Lemma 3.1, and 3.2 still hold.*

Theorem 6.2. *For non-full CQs, Theorem 3.5 still holds.*

In terms of computation, we observe that $T_E(\mathbf{I})$ with projection is still an AJAR/FAQ query, but now with 3 semiring aggregations (max, +, max), so it can still be computed by the algorithm in [3, 21] in polynomial time. Furthermore, one can verify that Lemma 3.9 still holds non-full queries, so it takes $O(1)$ time to compute $RS(\cdot)$ after all the $T_E(\mathbf{I})$'s have been computed.

Theorem 6.3. *For any non-full CQ q, $RS(\cdot)$ is an ε-DP mechanism that can be computed in* $\mathrm{poly}(N)$ *time.*

Non-full CQs with predicates can be handled by combining the methods described in this and Section 5. More precisely, for a non-full CQ with general predicates, we add $\pi_{\mathbf{o}_{\bar{E}}}$ on both sides of (19); if the predicates are inequalities and comparisons, we materialize each predicate and then apply the algorithm above. The resulting $RS(\cdot)$ is still ε-DP, and can be much smaller than that on the full CQ. However, the lower bound Theorem 4.7 no longer holds for non-full CQs, thus $RS(\cdot)$ is not $O(1)$-neighborhood optimal. We complement this with the following negative result.

Theorem 6.4. *For any $\varepsilon > 0$, any (r, c)-neighborhood optimal ε-DP mechanism $\mathcal{M}(\cdot)$ for the query $q := \pi_{x_1} (R_1(x_1, x_2) \bowtie R_2(x_2))$, where R_1 is the private relation, must have $cr^2 \geq N$.*

Thus, if one still desires $c = O(1)$, r must be at least $\Omega(\sqrt{N})$. We also remark that this negative result holds even under relaxed DP, since only substitutions are used when defining the neighborhoods in the proof.

7 PRACTICAL PERFORMANCE

Besides its theoretical optimality, our DP mechanism also enjoys excellent practical performance compared with prior work. Some preliminary experimental results are provided in Appendix F, which show that the accuracy of $RS(\cdot)$ is very close to that of $SS(\cdot)$ with order-of-magnitude reduction in computational cost. In fact, for most queries, $SS(\cdot)$ is not even known to be polynomially computable. For these queries, $ES(\cdot)$ is the only known DP mechanism, and $RS(\cdot)$ offers drastic improvement in terms of utility. Finally, it is worth pointing out that our mechanism just requires the evaluation of a number of residual queries, whose results are then combined using a certain formula. Hence, it can be implemented in a relational DBMS easily (e.g., using PL/SQL or a plug-in).

ACKNOWLEDGMENTS

This work has been supported by HKRGC under grants 16201318, 16201819, and 16205420. We would also like to thank the anonymous reviewers who have made valuable suggestions on improving the presentation of the paper.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of databases*. Vol. 8. Addison-Wesley Reading.

[2] Mahmoud Abo Khamis, Hung Ngo, and Dan Suciu. 2017. What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another?. In *Proc. ACM Symposium on Principles of Database Systems*.

[3] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. FAQ: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 13–28.

[4] Hilal Asi and John C Duchi. 2020. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in Neural Information Processing Systems* 33 (2020).

[5] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 739–748.

[6] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*. 6277–6287.

[7] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 273–282.

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

[9] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.

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

[11] Xiaofeng Ding, Xiaodong Zhang, Zhifeng Bao, and Hai Jin. 2018. Privacy-preserving triangle counting in large graphs. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1283–1292.
[12] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2T: Instance-optimal Truncation for Differentially PrivateQuery Evaluation with Foreign Keys. In *Proc. ACM SIGMOD International Conference on Management of Data*.
[13] Wei Dong and Ke Yi. 2021. Residual Sensitivity for Differentially Private Multi-Way Joins. In *Proc. ACM SIGMOD International Conference on Management of Data*.
[14] Wei Dong and Ke Yi. 2021. Universal Private Estimators. *arXiv preprint arXiv:2111.02598* (2021).
[15] Wei Dong and Ke Yi. 2022. A Nearly Instance-optimal Differentially Private Mechanism for Conjunctive Queries. (2022). https://www.cse.ust.hk/~yike/DP-CQ.pdf
[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[17] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
[18] R. Fagin, A. Lotem, and M. Noar. 2003. Optimal aggregation algorithms for middleware. *J. Comput. System Sci.* 66 (2003), 614–656.
[19] Georg Gottlob, Stephanietien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *J. ACM* 59, 3 (2012).
[20] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*. 2339–2347.
[21] Manas R Joglekar, Rohan Puttagunta, and Christopher Ré. 2016. Ajar: Aggregations and joins over annotated relations. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 91–106.
[22] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.
[23] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.
[24] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.
[25] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 193–204.
[26] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: a differentially private SQL query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.
[27] Jure Leskovec and Andrej Krevl. 2016. SNAP datasets: Stanford large network dataset collection (2014). *URL http://snap. stanford. edu/data* (2016), 49.
[28] Arjun Narayan and Andreas Haeberlen. 2012. DJoin: Differentially private join queries over distributed databases. In *USENIX Symposium on Operating Systems Design and Implementation*. 149–162.
[29] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.
[30] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating Data to Sensitivity in Private Data Analysis. *Proceedings of the VLDB Endowment* 7, 8 (2014).
[31] Wahbeh Qardaji, Weining Yang, and Ninghui Li. [n.d.]. Practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.
[32] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.
[33] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. 2009. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 107–116.
[34] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing Local Sensitivities of Counting Queries with Joins. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 479–494.
[35] Salil Vadhan. 2017. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer, 347–450.
[36] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering* 23, 8 (2010), 1200–1214.
[37] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 731–745.
[38] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 587–595.

A ANOTHER EXAMPLE ON COMPUTING GS

Example A.1. Consider the path-4 query

$$q = \mathsf{Edge}(x_1, x_2) \bowtie \mathsf{Edge}(x_2, x_3) \bowtie \mathsf{Edge}(x_3, x_4) \bowtie \mathsf{Edge}(x_4, x_5).$$

We have

$$\begin{aligned} GS \le & \mathsf{AGM}(\mathsf{Edge}_2(x_3) \bowtie \mathsf{Edge}_3(x_3, x_4) \bowtie \mathsf{Edge}_4(x_4, x_5)) \\ & + \mathsf{AGM}(\mathsf{Edge}_1(x_1) \bowtie \mathsf{Edge}_3(x_4) \bowtie \mathsf{Edge}_4(x_4, x_5)) \\ & + \mathsf{AGM}(\mathsf{Edge}_1(x_1, x_2) \bowtie \mathsf{Edge}_2(x_2) \bowtie \mathsf{Edge}_4(x_5)) \\ & + \cdots \\ = & O(N^2). \end{aligned}$$

B COMPARISON WITH THE NEIGHBORHOOD LOWER BOUND IN [4]

The r-neighborhood lower bound by Asi and Duchi (Lemma C.1 in [4]), when specialized to the 1-dimensional case, is as follows. Given a query q, for any k and any instance $\mathbf{I}$, define

$$\omega(\mathbf{I}, r) := \max_{\mathbf{I}', d(\mathbf{I}, \mathbf{I}') \le k} |q(\mathbf{I}) - q(\mathbf{I}')|.$$

If $\{q(\mathbf{I}') : d(\mathbf{I}, \mathbf{I}') \le k\}$ contains an interval of length $c \cdot \omega(\mathbf{I}, k)$ for some $c > 0$ and all $k \le r$, then

$$\max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \le r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \ge \frac{c}{16} \max_{k \le r} e^{-\varepsilon k} \omega(\mathbf{I}, k). \tag{22}$$

Our lower bound, which does not require any condition on q, is

$$\max_{\mathbf{I}': d(\mathbf{I}, \mathbf{I}') \le r} \mathrm{Err}(\mathcal{M}', \mathbf{I}') \ge \frac{1}{2\sqrt{1 + e^{\varepsilon}}} \cdot LS^{(r-1)}(\mathbf{I}). \tag{23}$$

Next we compare (22) and (23). By the definition of $LS^{(k)}(\mathbf{I})$ and $\omega(\mathbf{I}, k)$, we have

$$\omega(\mathbf{I}, k) \le \sum_{0 \le k' \le k-1} LS^{(k')}(\mathbf{I}) \le k \cdot LS^{(k-1)}(\mathbf{I}).$$

Then,

$$(22) \le \frac{c}{16} \max_{k \le r} (e^{-\varepsilon k} \cdot k \cdot LS^{(k-1)}(\mathbf{I})) \le \frac{c}{16} \cdot \max_{k \le r} (k e^{-\varepsilon k}) \cdot LS^{(r-1)}(\mathbf{I}),$$

which is asymptotically upper bounded by (23) as long as

$$k e^{-\varepsilon k} \le O\left(\frac{1}{\sqrt{1 + e^{\varepsilon}}}\right),$$

which is true when $\varepsilon = O(1)$.

On the other hand, the gap between (22) and (23) can be $\mathrm{poly}(N)$. Consider the Median query with a constant ε. Let $\mathbf{I}$ consist of $\log N$ copies of 0.5, while the remaining entries are half 0 and half 1. For any $r \ge \log N$, our lower bound (23) is $LS^{(r)}(\mathbf{I}) = 0.5$, while their lower bound (22) is

$$\max_{k \le r} e^{-\varepsilon k} \omega(\mathbf{I}, k) \le e^{-\varepsilon \log N} \cdot 0.5 = 1/N^{\Omega(1)}.$$

Nevertheless, Lemma C.1 in [4] yields better lower bounds for high-dimensional queries.

C NON-OPTIMALITY OF ELASTIC SENSITIVITY

Elastic sensitivity [22], denoted as $ES(\cdot)$, is the only other DP mechanism for CQs with self-joins. It is also a version of $\hat{SS}(\cdot)$, but defined using a different $\hat{LS}^{(k)}(\cdot)$. For $i \in [n], \mathbf{x} \subseteq \mathbf{x}_i$, let $mf(\mathbf{x}, I_i(\mathbf{x}_i))$ be the *maximum frequency* in $I_i(\mathbf{x}_i)$ on attributes $\mathbf{x}$, i.e., $mf(\mathbf{x}, I_i(\mathbf{x}_i)) = \max_{t \in \mathbf{dom}(\mathbf{x})} |I_i(\mathbf{x}_i) \ltimes t|$. For $ES(\cdot)$, $\hat{LS}^{(k)}(\cdot)$ is defined as a product of a number of maximum frequencies; please see [22] for the exact formula.

We give an example below showing that $ES(\mathbf{I})$ can be asymptotically larger than GS. This means that $ES(\cdot)$ is not even worst-case optimal (i.e., not N-neighborhood optimal).

Example C.1. Consider the path-4 query:

$$q = \mathsf{Edge}(x_1, x_2) \bowtie \mathsf{Edge}(x_2, x_3) \bowtie \mathsf{Edge}(x_3, x_4) \bowtie \mathsf{Edge}(x_4, x_5).$$

We showed that $GS = O(N^2)$ in Example A.1. Now consider the following instance $\mathbf{I}$ on Edge relation (assume the domain is $\mathbb{N}$):

$$\begin{aligned}\mathbf{I}(\mathsf{Edge}) =& \{(0,1),(0,2),\ldots,(0,\frac{N}{2}),\\ & (\frac{N}{2}+1, N+1),\ldots,(N,N+1)\}.\end{aligned}$$

Note that $mf(x_i, E(x_i, x_{i+1})) = mf(x_{i+1}, E(x_i, x_{i+1})) = \frac{N}{2}$ for $i = 1, 2, 3, 4$. By the formula in [22], we have $\hat{LS}^{(0)}(\mathbf{I}) = 4(\frac{N}{2})^3 = \frac{N^3}{2}$, thus

$$ES(\mathbf{I}) = \max_{k \geq 0} e^{-\beta k} \hat{LS}^{(k)}(\mathbf{I}) \geq \hat{LS}^{(0)}(\mathbf{I}) = \Omega(N^3).$$

D NONEXISTENCE OF OPTIMAL DP MECHANISMS FOR CQs WITH ARBITRARY PREDICATES

We show that, when the $P(\mathbf{y}_j)$'s are arbitrary (but still computable), it is undecidable to check if a given CQP is a trivial query. Recall that if the query is trivial, the optimal DP mechanism $\mathcal{M}$ is deterministic and achieves $\mathrm{Err}(\mathcal{M}, \mathbf{I}) = 0$ for all $\mathbf{I}$; otherwise, the mechanism must be probabilistic. Since one cannot distinguish between the two cases, optimal (under any notion of optimality) DP mechanisms do not exist.

For the undecidability result, just consider the simple CQP $q_M = R(x) \bowtie P_M(x)$, where $P_M(x) = \mathrm{True}$ iff the Turing machine M terminates in less than x steps. Note that $P_M(x)$ is decidable. However, it is easy to see that $|q_M(\cdot)| \equiv 0$ iff M does not halt.

E INCORRECTNESS OF USING ACTIVE DOMAIN TO COMPUTE $T_E(\cdot)$ FOR CQs WITH COMPARISON AND INEQUALITY PREDICATES

Example E.1. Following Example 5.1, suppose $P_1(x_2, x_4)$ is $x_2 > x_4$, $P_2(x_2, x_8)$ is $x_8 > x_2$, while ignoring P_3 and P_4. Consider the following instance $\mathbf{I}$:

$$\begin{aligned}R_1(x_1, x_2, x_3) &= \{(0,3,0),(0,5,0)\},\\ R_2(x_3, x_4, x_5) &= \{(0,1,0),(0,2,0),(0,3,0)\},\\ R_3(x_5, x_6, x_7) &= \{(0,0,0)\},\\ R_4(x_7, x_8, x_1) &= \{(0,5,0),(0,6,0),(0,7,0)\}.\end{aligned}$$

For $E = \{1\}$, $T_{\bar{E}}(\mathbf{I})$ attains its maximum at $x_2 = 4$, which is not included in $\mathbb{Z}^*(q, \mathbf{I})$.

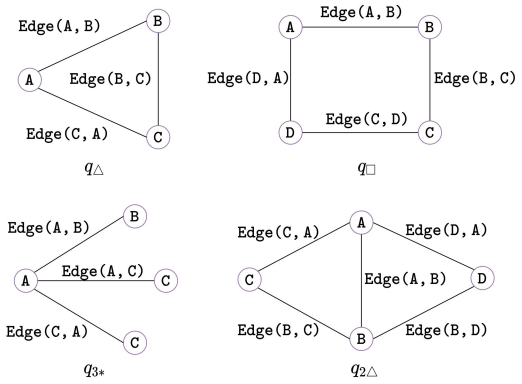

Figure 2: The join structure of queries.

F EXPERIMENTS

Our analysis in Section 4 shows that $RS(\cdot)$ is at most a constant-factor larger than $SS(\cdot)$, both of which are $O(1)$-neighborhood optimal. At the same time, $ES(\cdot)$ does not have any optimal guarantee. In this section, we conduct an experimental study on the actual values of these sensitivities on some sub-graph counting queries over real-world graph data. Note that since the subsequent noise generation process is the same for all three sensitivity measures, it suffices to only compare the sensitivities instead of the ℓ_2-errors.

F.1 Setup

Datasets. We use five graph network datasets: **CondMat**, **AstroPh**, **HepPh**, **HepTh**, and **GrQc**, which contain 23133, 18772, 12008, 9877, 5242 nodes and 186878, 396100, 236978, 51946, 28980 edges, respectively. These five datasets describe the collaboration between the authors on arXiv in Condensed Matter, Physics, High Energy Physics, High Energy Physics Theory and General Relativity categories. The datasets are obtained from SNAP [27]. The graphs are directed and we store all edges in a relation $\mathsf{Edge}(\mathsf{From}, \mathsf{To})$.

Queries. We experimented with 4 pattern counting queries as shown in Figure 2. We also added all inequalities between every two distinct variables. Since polynomial-time algorithms for computing $SS(\cdot)$ are known only for triangle counting [29] and t-star counting [23], the results on $SS(\cdot)$ are available only on $q_\triangle$ and q_{3*} Note that the count returned by the graph pattern counting CQ is actually 3 times (resp. 6 times) the number of triangles and 3-stars in the graph, so we need to scale down $SS(\cdot)$ and $ES(\cdot)$ accordingly.

F.2 Experimental Results

The experiments were conducted on a Linux server equipped with a 48-core 2.2GHz Intel Xeon CPU and 512GB of memory. For running

Dataset			CondMat	AstroPh	HepPh	HepTh	GrQc
$q_{\triangle}$	Query result		1,040,166	8,108,646	20,150,994	170,034	289,560
	Smooth sensitivity (SS)	Value	489	1,050	1,350	102	183
		Running Times(s)	895	615	261	171	50.7
	Residual Sensitivity (RS)	Value	493	1,054	1,354	205	222
		Running Times(s)	6.17	24	24.7	1.37	1.05
	Elastic Sensitivity (ES)	Value	234,361	763,561	724,717	12,871	19,927
		Running Times(s)	3.5	10.8	13	0.874	0.7
	RS vs SS	Value RS/SS	1.01×	1.00×	1.00×	2.01×	1.21×
		Running Times SS/RS	145×	25.6×	10.6×	124×	48.3×
	RS vs ES	Value ES/RS	475×	724×	535×	62.8×	89.8 ×
		Running Times RS/ES	1.76×	2.22×	1.9×	1.57×	1.4×
q_{3*}	Query result		222,690,360	3,274,065,312	7,661,801,994	12,590,010	14,896,428
	Smooth sensitivity (SS)	Value	232,686	760,536	721,770	12,480	19,440
		Running Times(s)	3.09	3.08	2.47	1.92	1.52
	Residual Sensitivity (RS)	Value	233,524	762,049	723,244	12,676	19,684
		Running Times(s)	0.463	0.4	0.462	0.374	0.316
	Elastic Sensitivity (ES)	Value	234,361	763,561	724,717	12,871	19,927
		Running Times(s)	0.314	0.272	0.348	0.234	0.197
	RS vs SS	Value RS/SS	1.00×	1.00×	1.00×	1.02×	1.01×
		Running Times SS/RS	6.67×	7.69×	5.34×	5.13×	4.83×
	RS vs ES	Value ES/RS	1.00×	1.00×	1.00×	1.02×	1.01×
		Running Times RS/ES	1.47×	1.47×	1.33×	1.6×	1.6×
$q_{\square}$	Query result		12,043,064	359,332,392	3,894,935,680	1,912,648	8,437,784
	Residual Sensitivity (RS)	Value	12,575	72,832	313,976	7,089	8,927
		Running Times(s)	41.3	296	120	6.49	2.17
	Elastic Sensitivity (ES)	Value	87,338,719	513,622,369	474,931,535	1,124,111	2,165,455
		Running Times(s)	2.62×	12.5×	11.5×	0.651×	0.388×
	RS vs ES	Value ES/RS	6,950×	7,050×	1,510×	159×	243×
		Running Times RS/ES	15.8×	23.8×	10.4×	9.97×	5.59×
$q_{2\triangle}$	Query result		9,398,600	289,422,860	3,747,561,340	1,716,052	8,165,996
	Residual Sensitivity (RS)	Value	308,937	361,551	515,616	279,488	285,394
		Running Times(s)	20.8	84.8	118	4.37	4.6
	Elastic Sensitivity (ES)	Value	30,514,062,601	323,903,424,601	291,786,363,781	92,041,951	220,614,031
		Running Times(s)	3.79	9.57	13.2	0.75	0.669
	RS vs ES	Value ES/RS	98,800×	896,000×	566,000×	329×	773×
		Running Times RS/ES	5.49×	8.86×	8.92×	5.77×	6.87×

Table 1: Comparison between smooth sensitivity, residual sensitivity and elastic sensitivity when $\beta = 0.1$.

time, we repeated each query 30 times and took the average. Table 1 gives the results for the setting $\beta = 0.1$, which corresponds to $\varepsilon = 1$.

Let's first compare $RS(\cdot)$ and $SS(\cdot)$. We see that $RS(\cdot)$ is only 2% larger than $SS(\cdot)$ in most cases, and the largest difference is 2 times for $q_{\triangle}$ on the HepTh dataset. This shows that the constant factor derived in Lemma 4.8 is actually quite loose, and the practical utility of $RS(\cdot)$ is close to that of $SS(\cdot)$. On the other hand, $SS(\cdot)$ is much more computational costly: the time for computing $SS(\cdot)$ is $4.83 \sim 145$ times that of $RS(\cdot)$.

For the comparison between $RS(\cdot)$ and $ES(\cdot)$, for $q_{\triangle}$, q_{3*} and $q_{2\triangle}$, $ES(\cdot)$ is much larger than $RS(\cdot)$. On q_{3*}, all sensitivity measures are very close. This is because, for this query, the query result solely depends on the degrees (namely, this is an easy query), and so do all three sensitivities. Actually, the formulation of $ES(\cdot)$ essentially only makes use of the degree information, which can be verified by the fact that its values on $q_{\triangle}$ and q_{3*} are equal. On the other hand, $RS(\cdot)$ and $ES(\cdot)$ exploits the actual structure of the graph.

We also tested with different values of β. The results in Figure 3 show this does not affect the sensitivity measures much, except for very small values of β (i.e., the high privacy regime).

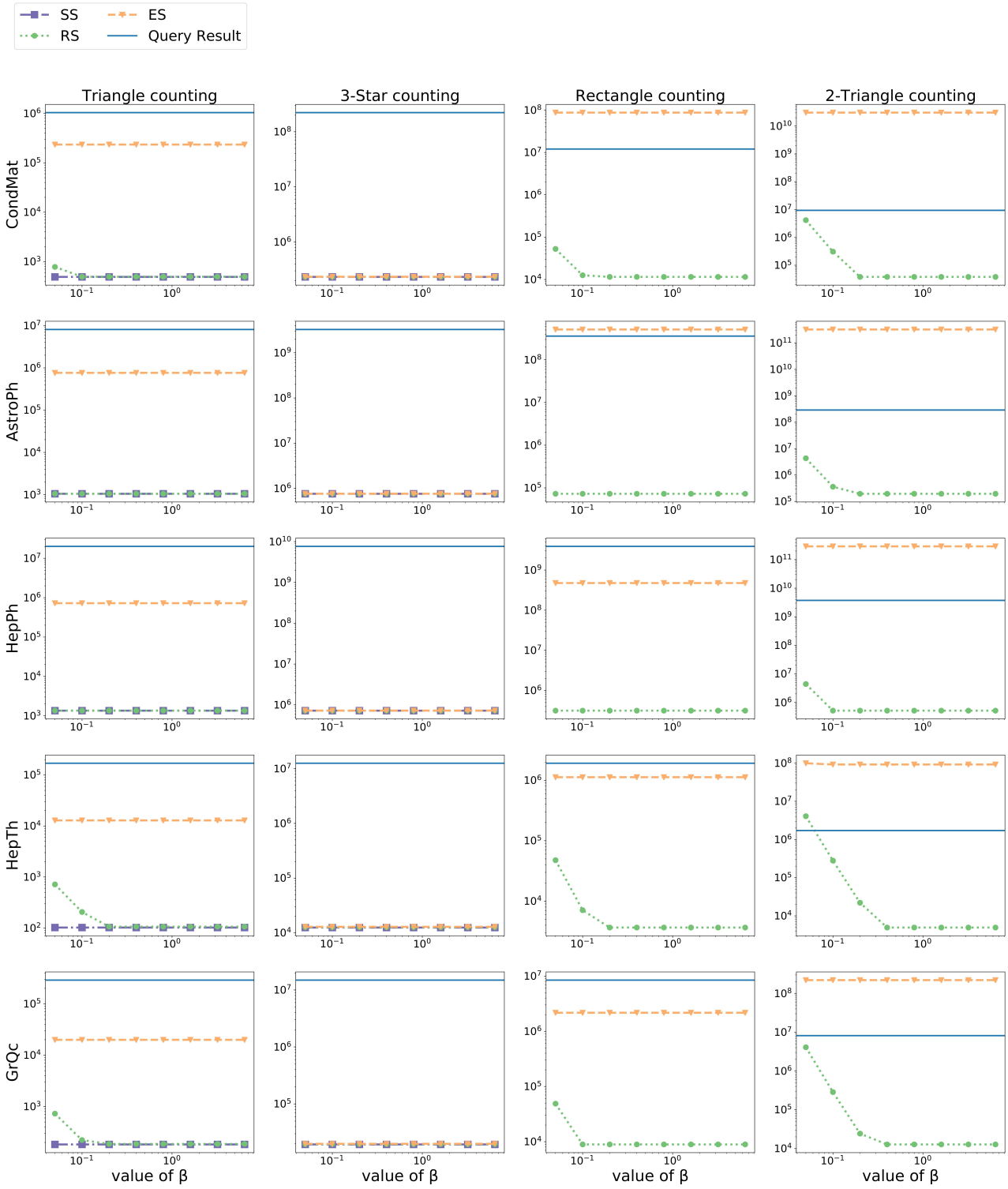

Figure 3: Smooth sensitivity, residual sensitivity and elastic sensitivity with different β under different data and queries.