Computing Inconsistency Measures Under Differential Privacy

Anonymous Author(s)*

Abstract

Assessing data quality is crucial to knowing whether and how to use the data for different purposes. Specifically, given a collection of integrity constraints, various ways have been proposed to quantify the inconsistency of a database. Inconsistency measures are particularly important when we wish to assess the quality of private data without revealing sensitive information. In this work, we study the estimation of inconsistency measures for a database protected under Differential Privacy (DP). Such estimation is nontrivial since some measures intrinsically query sensitive information, and the computation of others involves functions on underlying sensitive data. Among five inconsistency measures that have been proposed in recent work, we identify that two are intractable in the DP setting. The major challenge for the other three is high sensitivity: adding or removing one tuple from the dataset may significantly affect the outcome. To mitigate that, we model the dataset using a conflict graph and investigate private graph statistics to estimate these measures. The proposed machinery includes adapting graph-projection techniques with parameter selection optimizations on the conflict graph and a DP variant of approximate vertex cover size. We experimentally show that we can effectively compute DP estimates of the three measures on five real-world datasets with denial constraints, where the density of the conflict graphs highly varies.

Keywords

Differential privacy, Inconsistency measures, Integrity constraints

ACM Reference Format:

Anonymous Author(s). 2018. Computing Inconsistency Measures Under Differential Privacy. In *Proceedings of SIGMOD International Conference on Management of Data (SIGMOD'25).* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Differential Privacy (DP) [17] has become the de facto standard for querying sensitive databases and has been adopted by various industry and government bodies [1, 13, 19]. DP offers high utility for aggregate data releases while ensuring strong guarantees on the sensitive data of individuals. The laudable progress in DP study, as demonstrated by multiple recent works [28, 38, 61–64], has made it approachable and useful in many common scenarios. A standard DP mechanism involves adding noise to the query output, constrained by a privacy budget that quantifies the permitted privacy leakage. Once the privacy budget is exhausted, no more queries can be answered directly using the database. However, while DP ensures data privacy, it limits the ability of users to directly observe or assess data quality, leaving them to rely on the data without direct validation.

Data quality has been thoroughly studied for databases without privacy concerns, as databases that lack quality and consistency cannot be used to derive trustworthy conclusions and train reliable models. A significant portion of previous work has been devoted to various integrity constraints that can capture quality issues, such as functional dependencies, conditional functional dependencies [8], and denial constraints [9], as well as various algorithms for repairing data quality issues by employing these constraints [11, 24, 44, 55]. These works propose several models for changing the database to repair it, including tuple deletion [24, 44] which is the most basic model for data repairing, and cell value updates [11, 55] which is a more granular manner of changing the data. Regardless of the employed model, the goal is for the changed dataset to comply with the given constraints. While these approaches effectively repair data to meet quality standards, they do not address cases where DP must be maintained. This leaves a gap in scenarios where users need to measure inconsistency without directly accessing or modifying sensitive data.

In this work, we aim to bridge this gap by considering the problem of assessing the quality of databases protected by DP with respect to integrity constraints. Such quality assessment will allow users to decide whether they can rely on conclusions drawn from the data, or even if the suggested data is suitable for them. To solve this problem, we must tackle several challenges. First, since the database is protected by DP, users are only allowed to observe noisy aggregate statistics which can be difficult to summarize into a quality score. Second, if the number of constraints is large (e.g., if they were generated with an automatic system [6, 42, 53]), translating each constraint to an SQL `COUNT` query and evaluating it over the database with a DP mechanism may lead to low utility since the number of queries is large, allowing for only a small portion of the privacy budget to be allocated to each query.

Hence, our proposed solution employs *inconsistency measures* [45, 51, 58] that quantify data quality with a single number for all constraints, essentially yielding *a data quality score.* This approach aligns well with DP, as such measures give a single aggregated numerical value representing data quality, regardless of the given number of constraints. Prior work [45] described various measures, such as the *problematic measure*, counting the number of constraint violations, the *minimal repair measure*, counting the minimal tuple deletions needed to achieve consistency. These measures often exhibit a broad range of outputs, making them more suitable for estimating inconsistency under DP. Still, the manner of computing these measures while satisfying DP is not straightforward.

As mentioned, one approach is to treat each inconsistency measure as an SQL query and answer these queries using DP SQL mechanisms [14, 31, 38, 57]. Queries expressing these measures

SIGMOD'25, June 22-27, 2025, Berlin, Germany

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

DP Algorithms	Adult [3]	Flight [49]	Stock [50]
R2T [14]	0.17 ± 0.01	0.12 ± 0.03	123.19 ± 276.73
This work	0.10 ± 0.05	0.10 ± 0.20	0.07 ± 0.08

Table 1: Relative errors for a SQL approach vs our approach to compute the minimal inconsistency measure at $\varepsilon = 1$

require self-joins and include complex `OR` conditions. However, the state-of-the-art DP mechanism for self-join queries, R2T [14], only supports measures that can be expressed as the SPJA queries. Measures that require specific operators, like `DISTINCT` or `GROUP BY` cannot be handled by R2T. Moreover, even for measures compatible with R2T, the accuracy of the results varies widely over different datasets, as shown by Example 1 in the sequel.

Conversely, our approach models the violations of the integrity constraints in the dataset as a *conflict graph*, where nodes are tuple identifiers and there is an edge between a pair of tuples if this pair violates a constraint. Then, each inconsistency measure can be mapped to a specific graph statistic. Using this view of the problem allows us to leverage prior work on releasing graph statistics with DP [12, 29, 36] and develop tailored mechanisms for computing inconsistency measures with DP.

We formally define the problem of computing inconsistency measures while satisfying DP and choose the measures that are suitable for computation with DP from those proposed in prior work [45] by an analysis of their sensitivity and computational cost. We then devise novel DP mechanisms to effectively estimate these measures. Our approach harnesses graph projection techniques from the state-of-the-art DP algorithms [12] that truncate the graph in order to achieve DP. While these algorithms have proven effective in prior studies on social network graphs, they may encounter challenges with conflict graphs arising from their unique properties. To overcome this, we devise a novel optimization for choosing the truncation threshold. We further provide a DP mechanism for the minimal repair measure that augments the classic 2-approximation of the vertex cover algorithm [59] to restrict its sensitivity and allow for effective DP guarantees with high utility. Our experimental study shows that our novel algorithms prove effective for different datasets with various conflict graph sizes and sparsity levels.

Example 1. *In Table 1 we show the results of evaluating R2T [14] on three datasets with the same privacy budget of* 1 *for the minimal inconsistency measure. Though R2T performed well for the Adult and Flight datasets, it reports more than 120% relative errors for the Stock dataset that had very few violations. On the other hand, our approach demonstrates strong performance across all three datasets.*

The main contributions of this paper are as follows:

- We formulate the novel problem of computing inconsistency measurements with DP for private datasets and discuss the associated challenges, including a thorough analysis of the sensitivity of each measure.
- We devise several algorithms that leverage the conflict graph and algorithms for releasing graph statistics under DP to estimate the measures that we have determined are suitable. Specifically, we propose a new optimization for choosing graph truncation threshold that is tailored to conflict graphs and augment the classic vertex cover approximation algorithms to bound its sensitivity to 2 in order to obtain accurate estimates of the measures.
- We present experiments on five real-world datasets with varying sizes and densities to show that the proposed DP algorithms are efficient in practice. Our average error across these datasets is 1.3%-67.9% compared to the non-private measure.

2 Preliminaries

We begin with some background that we need for describing the concept of inconsistency measures for private databases.

2.1 Database and Constraints

We consider a single-relation schema $\mathcal{A} = (A_1, \ldots, A_m)$, which is a vector of distinct attribute names A_i, each associated with a domain $\mathtt{dom}(A_i)$ of values. A database D over $\mathcal{A}$ is a associated with a set $tids(D)$ of *tuple identifiers*, and it maps every identifier $i \in tids(D)$ to a tuple $D[i] = (a_1, \ldots, a_m)$ in $A_1 \times \cdots \times A_m$. A database D' is a *subset* of D, denoted $D' \subseteq D$, if D' is obtained from D by deleting zero or more tuples, that is, $tids(D') \subseteq tids(D)$ and $D'[i] = D[i]$ for all $i \in tids(D')$.

Following previous work on related topics [21, 42], we focus on Denial Constraints (DCs) on pairs of tuples. Using the formalism of Tuple Relational Calculus, such a DC is of the form $\forall t, t' \neg(\varphi_1 \wedge \cdots \wedge \varphi_k)$ where each φ_j is a comparison $\sigma_1 \circ \sigma_2$ so that: (a) each of σ_1 and σ_2 is either $t[A_i]$, or $t'[A_i]$, or a, where A_i is some attribute and a is a constant value, and (b) the operator $\circ$ belongs to set $\{<, >, \leq, \geq, =, \neq\}$ of comparisons. This DC states that there cannot be two tuples t and t' such that all comparisons φ_j hold true (i.e., at least one φ_j should be violated).

Note that the class of DCs of the form that we consider generalizes the class of Functional Dependencies (FDs). An FD has the form $X \rightarrow Y$ where $X, Y \subseteq \{A_1, \ldots, A_m\}$, and it states that every two tuples that agree on (i.e., have the same value in each attribute of) X must also agree on Y.

In the remainder of the paper, we denote by Σ the given set of DCs. A database D *satisfies* Σ, denoted $D \models \Sigma$, if D satisfies every DC in Σ; otherwise, D *violates* Σ, denoted $D \not\models \Sigma$.

A common way of capturing the violations of Σ in D is through the *conflict graph* $\mathcal{G}_\Sigma^D$, which is the graph (V, E), where $V = tids(D)$ an edge $e = \{i, j\} \in E$ occurs whenever the tuples $D[i]$ and jointly $D[j]$ violate Σ. To simplify the notation, we may write simply $\mathcal{G}$ instead of $\mathcal{G}_\Sigma^D$ when there is no risk of ambiguity.

Example 2. *Consider a dataset that stores information about capital and country as shown in Figure 1. Assume an FD constraint σ : Capital $\rightarrow$ Country between attributes capital and country that says that the country of two tuples have to be the same if their capital is the same. Assume the dataset has 3 rows (white color) and a neighboring dataset has an extra row (grey color) with the typo in its country attribute. As shown in the right side of Figure 1, the dataset with 4 rows can be converted to a conflict graph with the nodes corresponding to each tuple and edges referring to conflicts between them. The $\mathcal{I}_{\mathsf{MI}}$ measure computes the size of the set of all minimally inconsistent subsets $|MI_\Sigma(D)|$ (the number of edges in the graph) for this dataset.*

ID	Capital	Country
1	Ottawa	Canada
2	Ottawa	Canada
3	Ottawa	Canada
4	Ottawa	Kanada

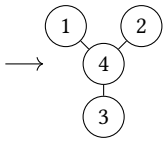


Figure 1: Toy example dataset to show a worst-case analysis. An additional row may violate all other rows in the dataset (left). Easier analysis can be done by instead converting the dataset into its corresponding conflict graph (right).

2.2 Inconsistency Measures

Inconsistency measures have been studied in previous work [4, 25, 26, 41, 43] as a means of measuring database quality for a set of DCs. We adopt the measures and notation of Livshits et al. [45]. Specifically, they consider five inconsistency measures that capture different aspects of the dataset quality. To define these concepts, we need some notation. Given a database D and a set Σ of anti-monotonic integrity constraints, we denote by $\mathsf{MI}_\Sigma(D)$ the set of all *minimally inconsistent subsets*, that is, the sets $E \subseteq D$ such that $E \not\models \Sigma$ but $E' \models \Sigma$ for all $E' \subsetneq E$. We also denote by $\mathsf{MC}_\Sigma(D)$ the set of all *maximal consistent subsets* of D; that is, the sets $E \subseteq D$ such that $E \models \Sigma$ and $E' \not\models \Sigma$ whenever $E \subsetneq E' \subseteq D$.

Definition 1 (Inconsistency measures [45]). *Given a database D and a set of DCs Σ, the inconsistency measures are defined as follows:*

- *Drastic measure: $\mathcal{I}_\mathsf{D}(D,\Sigma) = 1$ if $D \models \Sigma$ and 0 otherwise.*
- *Minimal inconsistency measure: $\mathcal{I}_\mathsf{MI}(D,\Sigma) = |\mathsf{MI}_\Sigma(D)|$.*
- *Problematic measure: $\mathcal{I}_\mathsf{P}(D,\Sigma) = |\cup\, \mathsf{MI}_\Sigma(D)|$.*
- *Maximal consistency measure: $\mathcal{I}_\mathsf{MC}(D,\Sigma) = |\mathsf{MC}_\Sigma(D)|$ [1].*
- *Optimal repair measure: $\mathcal{I}_\mathsf{R}(D,\Sigma) = |D| - |D_R|$, where $|D_R|$ is the largest subset $D_R \subseteq D$ such that $D_R \models \Sigma$.*

Observe that inconsistency measures also have a graphical interpretation with respect to the conflict graph $\mathcal{G}_\Sigma^D$. For instance, the drastic measure $\mathcal{I}_\mathsf{D}(D,\Sigma)$ corresponds to a binary indicator for whether there exists an edge in $\mathcal{G}_\Sigma^D$. We summarize the graph interpretation of these inconsistency measures in Table 2.

2.3 Differential Privacy

Differential privacy (DP) [17] aims to protect private information in the data. In this work, we consider the unbounded DP setting where we define two neighboring datasets, D and D' (denoted by $D \approx D'$) if D' can be transformed from D by adding or removing one tuple in D.

Definition 2 (Differential Privacy [17]). *An algorithm $\mathcal{M}$ is said to satisfy ε-DP if for all $S \subseteq Range(\mathcal{M})$ and for all $D \approx D'$,*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S]\,.$$

The privacy cost is measured by the parameters ε, often referred to as the *privacy budget*. The smaller ε is, the stronger the privacy is. Complex DP algorithms can be built from the basic algorithms following two essential properties of differential privacy:

Proposition 1 (DP Properties [15, 16]). *The following hold.*

(1) **(Sequential composition)** *If $\mathcal{M}_i$ satisfies ε_i-DP, then the sequential application of $\mathcal{M}_1, \mathcal{M}_2, \cdots$, satisfies $(\sum_i \varepsilon_i)$-DP.*

(2) **(Parallel composition)** *If each $\mathcal{M}_i$ accesses disjoint sets of tuples, then they together satisfy $(\max_i \varepsilon_i)$-DP.*

(3) **(Post-processing)** *Any function applied to the output of an ε-DP mechanism $\mathcal{M}$ also satisfies ε-DP.*

Many applications in DP require measuring the change in a particular function's result over two neighboring databases. The supremum over all pairs of neighboring databases is called the *sensitivity* of the function.

Definition 3 (Global sensitivity [18]). *Given a function $f : \mathcal{D} \to \mathbb{R}$, the sensitivity of f is*

$$\Delta_f = \max_{D' \approx D} |f(D) - f(D')|. \tag{1}$$

Laplace mechanism. The Laplace mechanism [18] is a common building block in DP mechanisms and is used to get a noisy estimate for queries with numeric answers. The noise injected is calibrated to the global sensitivity of the query.

Definition 4 (Laplace Mechanism [18]). *Given a database D, a function $f : \mathcal{D} \to \mathbb{R}$, and a privacy budget ε, the Laplace mechanism $\mathcal{M}_L$ returns $f(D) + v_q$, where $v_q \sim Lap(\Delta_f/\varepsilon)$.*

The Laplace mechanism can answer many numerical queries, but in many natural situations that require a non-numerical output, the exponential mechanism can be used.

Exponential mechanism. The exponential mechanism [47] expands the application of DP by allowing a non-numerical output.

Definition 5 (Exponential Mechanism [47]). *Given a dataset D, a privacy budget ε, a set Θ of output candidates, a quality function $q(D,\theta_i) \in \mathbb{R}$, the exponential mechanism $\mathcal{M}_{EM}$ outputs a candidate $\theta_i \in \Theta$ with probability proportional to $\exp\left(\frac{\varepsilon q(D,\theta_i)}{2\Delta_q}\right)$, where Δ_q is the sensitivity of the quality function q.*

DP for graphs. When the dataset is a graph $\mathcal{G} = (V, E)$, the standard definition can be translated to two variants of DP [29]. The first is *edge-DP* where two graphs are neighboring if they differ on one edge, and the second is *node-DP*, when two graphs are neighboring if one is obtained from the other by removing a node (and its incident edges). The two definitions offer different kinds of privacy protection. In our work, as we deal with databases and their corresponding conflict graphs, the addition or removal of a tuple of the dataset translates to node-DP. The corresponding definition of neighboring datasets changes to neighboring graphs where two graphs $\mathcal{G}$ and $\mathcal{G}'$ are called neighboring $\mathcal{G} \approx \mathcal{G}'$ if $\mathcal{G}'$ can be transformed from $\mathcal{G}$ by adding or removing one node along with all its edges in $\mathcal{G}$. Node-DP provides a stronger privacy guarantee than edge-DP since it protects an individual's privacy and all its connections, whereas edge-DP concerns only one such connection.

Definition 6 (Node sensitivity). *Given a function f over a graph $\mathcal{G}$, the sensitivity of f is $\Delta_f = \max_{\mathcal{G}' \approx \mathcal{G}} |f(\mathcal{G}) - f(\mathcal{G}')|$.*

The building blocks of DP, such as the Laplace and Exponential mechanisms, also work on graphs by simply substituting the input to a graph and the sensitivity to the corresponding node sensitivity.

Graph projection. Graph projection algorithms refer to a family of algorithms that help reduce the node sensitivity of a graph by

[1] We drop "-1" from the original definition [45] for simplicity.

	Non-private analysis [45]		DP analysis (this work)		
Inconsistency Measures for D	**Graph Interpretations in $\mathcal{G}_\Sigma^D(V,E)$**	**Computation cost**	**Sensitivity**	**Computation cost**	**Utility**
Drastic measure $\mathcal{I}_\mathsf{D}$	if exists an edge	$O(\lvert\Sigma\rvert n^2)$	1	N.A.	N.A.
Minimal inconsistency measure $\mathcal{I}_\mathsf{MI}$	the no. of edges	$O(\lvert\Sigma\rvert n^2)$	n	$O(\lvert\Sigma\rvert n^2 + \lvert\Theta\rvert m)$	$-\tilde{q}_{\text{opt}}(D,\varepsilon_2) + O(\frac{\theta_{\max}\ln\lvert\Theta\rvert}{\varepsilon_1})$
Problematic measure $\mathcal{I}_\mathsf{P}$	the no. of nodes with positive degrees	$O(\lvert\Sigma\rvert n^2)$	n	$O(\lvert\Sigma\rvert n^2 + \lvert\Theta\rvert m)$	$-\tilde{q}_{\text{opt}}(D,\varepsilon_2) + O(\frac{\theta_{\max}\ln\lvert\Theta\rvert}{\varepsilon_1})$
Maximal consistency measure $\mathcal{I}_\mathsf{MC}$	the no. of maximal independent sets	#P-complete	$O(3^n)$	N.A.	N.A.
Optimal repair measure $\mathcal{I}_\mathsf{R}$	the minimum vertex cover size	NP-hard	1	$O(\lvert\Sigma\rvert n^2 + m)$	$\mathcal{I}_R(D,\Sigma) + O(1/\varepsilon)$

Table 2: Summary of Inconsistency Measures, $n = |D| = |\mathcal{G}_\Sigma^D.V|$, $m = |\mathcal{G}_\Sigma^D.E|$, Θ is the candidate set.

truncating the edges and hence bounding the maximum degree of the graph. Several graph projection algorithms exist [7, 36], among which the *edge addition* algorithm [12] stands out for its effectiveness in preserving most of the underlying graph structure. The edge addition algorithm denoted by π_θ^Λ, takes as input the graph $\mathcal{G} = \mathcal{G}_\Sigma^D = (V,E)$, a bound on the maximum degree of each vertex (θ), and a stable ordering of the edges (Λ) to output a projected θ-bounded graph denoted by $\mathcal{G}_\theta = \pi_\theta^\Lambda(\mathcal{G})$.

Definition 7 (Stable ordering [12]). *A graph edge ordering Λ is stable if and only if given two neighboring graphs $\mathcal{G} = (V,E)$ and $\mathcal{G}' = (V',E')$ that differ by only a node, $\Lambda(\mathcal{G})$ and $\Lambda(\mathcal{G}')$ are consistent in the sense that if two edges appear both in $\mathcal{G}$ and $\mathcal{G}'$, their relative ordering are the same in $\Lambda(\mathcal{G})$ and $\Lambda(\mathcal{G}')$.*

The stable ordering of edges, $\Lambda(\mathcal{G})$, can be any deterministic ordering of all the edges E in the $\mathcal{G}$. Such stabling edge ordering can be easily obtained in practice. For example, it could be an ordering (e.g. alphabetical ordering) based on the node IDs of the graph such that in the neighboring dataset $\mathcal{G}'$, the edges occur in the same ordering as $\mathcal{G}$. The edge addition algorithm starts with an empty set of edges and operates by adding edges in the same order as Λ so that each node has a maximum degree of θ. To simplify the notation, in the remainder of the paper, we drop Λ and denote the edge addition algorithm $\pi_\theta^\Lambda(\mathcal{G})$ as $\pi_\theta(\mathcal{G})$.

Algorithm 1: Edge addition algorithm [12]

Data: Graph $\mathcal{G}(V,E)$, Bound θ, Stable ordering Λ
Result: θ-bounded graph $\pi_\theta(\mathcal{G})$

$E^\theta \leftarrow \emptyset; d(v) \leftarrow 0$ for each $v \in V$;
for $e = (u,v) \in \Lambda$ **do**
  if $d(u) < \theta \& d(v) < \theta$ **then**
    $E^\theta \leftarrow E^\theta \cup \{e\}$ $d(u) \leftarrow d(u)+1, d(v) \leftarrow d(v)+1$;
return $G^\theta = (V, E^\theta)$;

3 Inconsistency Measures under DP

Problem Setup. Consider a private dataset D, a set of DCs Σ, and a privacy budget ε. For an inconsistency measure $\mathcal{I}$ from the set $\{\mathcal{I}_\mathsf{D}, \mathcal{I}_\mathsf{MI}, \mathcal{I}_\mathsf{P}, \mathcal{I}_\mathsf{MC}, \mathcal{I}_\mathsf{R}\}$ (Definition 1), we would like to design an ε-DP algorithm $\mathcal{M}(D,\Sigma,\varepsilon)$ such that with high probability, $|\mathcal{M}(D,\Sigma,\varepsilon) - \mathcal{I}(D,\Sigma)|$ is bounded with a small error.

Sensitivity Analysis. We first analyze the sensitivity of the five inconsistency measures and discuss the challenges to achieving DP.

Proposition 2. *Given a database D and a set of DCs Σ, where $|D| = n$, the following holds:*

(1) The global sensitivity of $\mathcal{I}_\mathsf{D}$ is 1.
(2) The global sensitivity of $\mathcal{I}_\mathsf{MI}$ is n.
(3) The global sensitivity of $\mathcal{I}_\mathsf{P}$ is n.
(4) The global sensitivity of $\mathcal{I}_\mathsf{MC}$ is exponential in n.
(5) The global sensitivity of $\mathcal{I}_\mathsf{R}$ is 1.

The proof can be found in Appendix A.1.

Inadequacy of $\mathcal{I}_\mathsf{D}$ and $\mathcal{I}_\mathsf{MC}$. We note that two inconsistency measures are less suitable for DP. First, the drastic measure $\mathcal{I}_\mathsf{D}$ is a binary measure that outputs 1 if at least one conflict exists in the dataset and 0 otherwise. Due to its binary nature, the measure's sensitivity is 1, meaning adding or removing a single row can significantly alter the result. Adding DP noise to such a binary measure can render it meaningless.

One way to compute the $\mathcal{I}_\mathsf{D}$ measure could be to consider a proxy of $\mathcal{I}_\mathsf{D}$ by employing a threshold-based approach that relies on $\mathcal{I}_\mathsf{P}$ or $\mathcal{I}_\mathsf{MI}$. For example, if these measures are below a certain given number, we return 0 and, otherwise, return 1. A recent work [52] addresses similar problems for synthetic data generation by employing the exponential mechanism. However, since we focus on measures that are directly computable in the DP setting, we leave this intriguing subject for future work.

Additionally, prior work [41] showed that computing $\mathcal{I}_\mathsf{MC}$ is #P-complete, restricting its practicality. Even in special cases where $\mathcal{I}_\mathsf{MC}$ can be polynomially computed (when $\mathcal{G}_\Sigma^D$ is P_4-free [37]), its global sensitivity is exponential in the number of nodes of $\mathcal{G}_\Sigma^D$, diminishing the utility of its DP estimate. Therefore, we defer the study of $\mathcal{I}_\mathsf{D}$ and $\mathcal{I}_\mathsf{MC}$ to future work.

Challenges for $\mathcal{I}_\mathsf{R}$, $\mathcal{I}_\mathsf{MI}$, and $\mathcal{I}_\mathsf{P}$. Although the $\mathcal{I}_\mathsf{R}$ measure has a low sensitivity of 1 for its output range $[0,n]$, it is an NP-hard problem, and the common non-private solution is to solve a linear approximation that requires solving a linear program [41]. However, in the worst case, this linear program again has sensitivity equal to n (number of rows in the dataset) and may have up to $\binom{n}{2}$ number of constraints (all rows violating each other). Existing state-of-the-art DP linear solvers [30] are slow and fail for such a challenging task. Our preliminary experiments to solve such a linear program timed out after 24 hours with $n = 1000$. For $\mathcal{I}_\mathsf{MI}$ and $\mathcal{I}_\mathsf{P}$ they have polynomial computation costs and reasonable output ranges. However, they still have high sensitivity n. In the upcoming sections 4 and 5, we show that these problems can be alleviated by pre-processing the input dataset as a conflict graph and computing these inconsistency measures as private graph statistics.

4 DP Graph Projection for $\mathcal{I}_\mathsf{MI}$ and $\mathcal{I}_\mathsf{P}$

Computing graph statistics such as edge count and degree distribution while preserving node-differential privacy (node-DP) is a well-explored area [7, 12, 36]. Hence, in this section, we leverage

Algorithm 2: Graph projection approach for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

Data: Dataset D, constraint set Σ, candidate set Θ, privacy budgets ε_1 and ε_2
Result: DP inconsistency measure for $\mathcal{I}_{\mathsf{MI}}$ or $\mathcal{I}_{\mathsf{P}}$
1 Construct the conflict graph $\mathcal{G}_\Sigma^D$
2 Sample θ^* from Θ with a ε_1-DP mechanism // Basic EM (Algorithm 3); Optimized EM (Algorithm 4)
3 Compute θ^*-bounded graph $\mathcal{G}_{\theta^*} \leftarrow \pi_{\theta^*}(\mathcal{G}_\Sigma^D)$ // Edge addition algorithm [12]
4 **Return** $f(\mathcal{G}_{\theta^*}) + \mathrm{Lap}(\frac{\theta^*}{\varepsilon_2})$ // $f(\cdot)$ returns edge count for $\mathcal{I}_{\mathsf{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_{\mathsf{P}}$

the state-of-the-art node-DP approach for graph statistics to analyze the inconsistency measures $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ represented as graph statistics on the conflict graph $\mathcal{G}_\Sigma^D$. However, the effectiveness of this approach hinges on carefully chosen parameters. We introduce two optimization techniques that consider the constraints imposed on the database D to optimize parameter selection and enhance the algorithm's utility.

4.1 Graph Projection Approach for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

A primary utility challenge in achieving node-DP for graph statistics is their high sensitivity. In the worst case, removing a single node from a graph of n nodes can result in removing $(n-1)$ edges. To mitigate this issue, the state-of-the-art approach [12] first projects the graph $\mathcal{G}$ onto a θ-bounded graph $\mathcal{G}_\theta$, where the maximum degree is no more than θ. Subsequently, the edge count of the transformed graph is perturbed by the Laplace mechanism with a sensitivity value of less than n. However, the choice of θ is critical for accurate estimation. A small θ reduces Laplace noise due to lower sensitivity, but results in significant edge loss during projection. Conversely, a θ close to n preserves more edges but increases the Laplace noise. Prior work addresses this balance using the exponential mechanism (EM) to prefer a θ that minimizes the combined errors arising from graph projection and the Laplace noise.

We outline this general approach in Algorithm 2. This algorithm takes in the dataset D, the constraint set Σ, a candidate set Θ for degree bounds, and privacy budgets ε_1 and ε_2. These privacy budgets are later composed to get a final guarantee of ε-DP. We start by constructing the conflict graph $\mathcal{G}_\Sigma^D$ generated from the input dataset D and constraint set Σ (line 1), as defined in Section 2.1. Next, we sample in a DP manner a value of θ^* from the candidate set Θ with the privacy budget ε_1 (line 2). A baseline choice is an exponential mechanism detailed in Algorithm 3 to output a degree that minimizes the edge loss in a graph and the Laplace noise. In line 3, we compute a bounded graph $\mathcal{G}_{\theta^*}$ using the edge addition algorithm [12], we compute a θ^*-bounded graph $\mathcal{G}_{\theta^*}$ (detailed in Section 2). Finally, we perturb the true measure (either the number of edges for $\mathcal{I}_{\mathsf{MI}}$ or the number of positive degree nodes for $\mathcal{I}_{\mathsf{P}}$) on the projected graph, denoted by $f(\mathcal{G}_{\theta^*})$, by adding Laplace noise using the other privacy budget ε_2 (line 4).

The returned noisy measure at the last step has two sources of errors: (i) the bias incurred in the projected graph, i.e., $f(\mathcal{G}) - f(\mathcal{G}_{\theta^*})$, and (ii) the noise from the Laplace mechanism with an expected square root error $\sqrt{2}\theta^*/\varepsilon_2$. Both errors depend on the selected parameter θ^*, and it is important to select an optimal

Algorithm 3: EM-based first try for parameter selection

Data: Graph $\mathcal{G}$, candidate set Θ, quality function q, privacy budget $\varepsilon_1, \varepsilon_2$
Result: Candidate θ^*
1 Find the maximum value in Θ as $\theta_{\max}$
2 For each $\theta_i \in \Theta$, compute $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ // See Equation (2)
3 Sample θ^* with prob $\propto \exp(\frac{\varepsilon_1 q_{\varepsilon_2}(\mathcal{G},\theta_i)}{2\theta_{\max}})$
4 **Return** θ^*

θ	e_{bias}	$\sqrt{2}\theta/\varepsilon_2$	q
1	*2*	$\sqrt{2}$	$-2-\sqrt{2}$
2	*1*	$2\sqrt{2}$	$-1-2\sqrt{2}$
3	*0*	$3\sqrt{2}$	$-3\sqrt{2}$

Table 3: Quality function computation for $\mathcal{I}_{\mathsf{MI}}$ for the conflict graph in Figure 1 when $\varepsilon_2 = 1$

θ^* that minimizes the combined errors. Next, we describe a DP mechanism that helps select this parameter.

EM-based first try for parameter selection. The EM (Definition 5) specifies a quality function $q(\cdot,\cdot)$ that maps a pair of a database D and a candidate degree θ to a numerical value. The optimal θ value for a given database D should have the largest possible quality value and, hence, the highest probability of being sampled. We also denote $\theta_{\max}$ the largest degree candidate in Θ and use it as part of the quality function to limit its sensitivity.

The quality function we choose to compute the inconsistency measures includes two terms: for each $\theta \in \Theta$,

$$q_{\varepsilon_2}(\mathcal{G}, \theta) = -e_{\mathrm{bias}}(\mathcal{G}, \theta) - \sqrt{2}\theta/\varepsilon_2 \tag{2}$$

where the first term e_{bias} captures the bias in the projected graph, and the second term $\sqrt{2}\theta/\varepsilon_2$ captures the error from the Laplace noise at budget ε_2. For the minimum inconsistency measure $\mathcal{I}_{\mathsf{MI}}$, we define the bias term as

$$e_{\mathrm{bias}}(\mathcal{G}, \theta) = |\mathcal{G}_{\theta_{\max}}.E| - |\mathcal{G}_\theta.E| \tag{3}$$

i.e., the number of edges truncated at degree θ as compared to that at degree $\theta_{\max}$. For the problematic measure $\mathcal{I}_{\mathsf{P}}$, we have

$$e_{\mathrm{bias}}(\mathcal{G}, \theta) = |\mathcal{G}_{\theta_{\max}}.V_{>0}| - |\mathcal{G}_\theta.V_{>0}| \tag{4}$$

where $\mathcal{G}_\theta.V_{>0}$ denote the nodes with positive degrees.

Example 3. *Consider the same graph as Example 2 and a candidate set $\Theta = [1, 2, 3]$ to compute the $\mathcal{I}_{\mathsf{MI}}$ measure (number of edges) with $\varepsilon_2 = 1$. For the first candidate $\theta = 1$, as node 4 has degree 3, the edge addition algorithm would truncate 2 edges, for $\theta = 2$, 1 edge would be truncated and for $\theta = 3$, no edges would be truncated. We can, therefore, compute each term of the quality function for each θ given in Table 3. For this example, we see that $\theta = 1$ has the best quality even if it truncates the most number of edges as the error from Laplace noise overwhelms the bias error.*

We summarize the basic EM for the selection of the bounded degree in Algorithm 3. This algorithm has a complexity of $O(|\Theta|m)$, where m is the edge size of the graph, as computing the quality function for each θ candidate requires running the edge addition algorithm once. The overall Algorithm 2 has a complexity of

$O(|\Sigma|n^2 + |\Theta|m)$, where the first term is due to the construction of the graph.

Privacy analysis. The privacy guarantee of Algorithm 2 depends on the budget spent for the exponential mechanism and the Laplace mechanism, as summarized below.

Theorem 1. *Algorithm 2 satisfies $(\varepsilon_1 + \varepsilon_2)$-node DP for $\mathcal{G}_\Sigma^D$ and $(\varepsilon_1 + \varepsilon_2)$-DP for the input database D.*

The analysis is based on the sequential composition of two DP mechanisms. As stated below, we just need to analyze the sensitivity of the quality function in the exponential mechanism and the sensitivity of the measure over the projected graph.

Lemma 1. *The sensitivity of the quality function $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ in Algorithm 3 defined in Equation* (2) *is $\theta_{\max} = \max(\Theta)$.*

Lemma 2. *The sensitivity of $f \circ \pi_\theta(\cdot)$ in Algorithm 2 is θ, where π_θ is the edge addition algorithm with the user input θ, and $f(\cdot)$ return returns edge count for $\mathcal{I}_{\text{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_{\text{P}}$.*

Proofs for Theorem 1, Lemma 1, and Lemma 2 can be found in Appendix A.2.

Utility analysis. The utility of Algorithm 2 is directly encoded by the quality function of the exponential mechanism in Algorithm 3. We first define the best possible quality function value for a given database and its respective graph as

$$q_{\text{opt}}(D, \varepsilon_2) = \max_{\theta \in \Theta} q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) \tag{5}$$

and the set of degree values that obtain the optimal quality value as

$$\Theta_{\text{opt}} = \{\theta \in \Theta : q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) = q_{\text{opt}}(D, \varepsilon_2)\}. \tag{6}$$

However, we define e_{bias} as the difference in the number of edges or nodes in the projected graph $\mathcal{G}_\theta$ compared to that of $\mathcal{G}_{\theta_{\max}}$, instead of $\mathcal{G}$. This is to limit the sensitivity of the quality function. To compute the utility, we slightly modify the quality function without affecting the output of the exponential mechanism.

$$\tilde{q}_{\varepsilon_2}(\mathcal{G}, \theta) = q_{\varepsilon_2}(\mathcal{G}, \theta) + f(\mathcal{G}_{\theta_{\max}}) - f(\mathcal{G}_\Sigma^D), \tag{7}$$

where $f(\cdot)$ returns edge count for $\mathcal{I}_{\text{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_{\text{P}}$. This modified quality function should give the same set of degrees Θ_{opt} with optimal values equal to

$$\tilde{q}_{\text{opt}}(D, \varepsilon_2) = \max_{\theta \in \Theta} q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta) + f(\mathcal{G}_{\Sigma\ \theta_{\max}}^D) - f(\mathcal{G}_\Sigma^D). \tag{8}$$

Then, we derive the utility bound for Algorithm 2 based on the property of the exponential mechanism as follows.

Theorem 2. *On any database instance D and its respective conflict graph $\mathcal{G}_\Sigma^D$, let o be the output of Algorithm 2 with Algorithm 3 over D. Then, with a probability of at least $1 - \beta$, we have*

$$|o - a| \leq -\tilde{q}_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}| \cdot \beta}) \tag{9}$$

where a is the true inconsistency measure over D and $\beta \leq \frac{1}{e^{\sqrt{2}}}$.

The proof can be found in Appendix A.2.6.

This theorem indicates that the error incurred by Algorithm 2 with Algorithm 3 is directly proportional to the log of the candidate size $|\Theta|$ and the sensitivity of the quality function. Without prior knowledge about the graph, $\theta_{\max}$ is usually set as the number of nodes n, and Θ includes all possible degree values up to n, resulting in poor utility. Fortunately, for our use case, the edges in the graph arise from the DCs that are available to us. In the next section, we show how we can leverage these constraints to improve the utility of our algorithm by truncating candidates in the set Θ.

4.2 Optimized Parameter Selection

Our developed strategy to improve the parameter selection includes two optimization techniques. The overarching idea behind these optimizations is to gradually truncate large candidates from the candidate set Θ based on the density of the graph. For example, we observe that the Stock dataset [50] has a sparse conflict graph, and its optimum degree for graph projection is in the range of $10^0 - 10^1$. In contrast, the graph for the Adult dataset sample [3] is extraordinarily dense and has an optimum degree θ greater than 10^3, close to the sampled data size. Removing unneeded large candidates, especially those greater than the true maximum degree of the graph, can help the high sensitivity issue of the quality function and improve our chances of choosing a better bound.

Our first optimization estimates an upper bound for the true maximum degree of the conflict graph and removes candidates larger than this upper bound from the initial candidate set. The second optimization is a hierarchical exponential mechanism that utilizes two steps of exponential mechanisms. The first output, θ^1, is used to truncate Θ further by removing candidates larger than θ^1 from the set, and the second output is chosen as the final candidate θ^*. In the rest of this section, we dive deeper into the details of these optimizations and discuss their privacy analysis.

Estimating the degree upper bound using FDs. Given a conflict graph $\mathcal{G}(V, E)$, we use $d(\mathcal{G}, v)$ to denote the degree of the node $v \in V$ in $\mathcal{G}$ and $d_{\max}(\mathcal{G}) = \max_{v \in V} d(\mathcal{G}, v)$ to denote the maximum degree in $\mathcal{G}$. We estimate $d_{\max}$ by leveraging how conflicts were formed for its corresponding dataset D under Σ.

The degree for each vertex in $\mathcal{G}$ can be found by going through each tuple t in the underlying dataset D and checking for how many tuples jointly with t violate the constraint set Σ. However, computing this value for each tuple is computationally expensive and highly sensitive, making it impossible to learn directly with differential privacy. We observe that the conflicts that arise due to functionality dependencies (FDs) depend on the values of the left attributes in the FD.

Example 4. *Consider the same setup as Example 2 and an FD σ : Capital $\rightarrow$ Country. We can see that the number of violations added due to the erroneous grey row is 3. This number is also one smaller than the maximum frequency of values occurring in the Capital attribute, and the most frequent value is "Ottawa".*

Based on this observation, we can derive an upper bound for the maximum degree of a conflict graph if it involves only FDs, and this upper bound has a lower sensitivity. We show the upper bound in Lemma 3 for one FD first and later extend for multiple FDs.

Lemma 3. *Given a database D and a FD $\sigma : X \rightarrow Y$ as the single constraint, where $X = \{A_1, \ldots, A_k\}$ and Y is a single attribute. For its respective conflict graph $\mathcal{G}_{\Sigma=\{\sigma\}}^D$, simplified as $\mathcal{G}_\sigma^D$, we have the*

maximum degree of the graph $d_{\max}(\mathcal{G}_\sigma^D)$ upper bounded by

$$d_{\text{bound}}(D, X) = \max_{\vec{a_X} \in \text{dom}(A_1) \times \ldots \times \text{dom}(A_k)} \text{freq}(D, \vec{a_X}) - 1, \quad (10)$$

where $\text{freq}(D, \vec{a_X})$ *is the frequency of values* $\vec{a_X}$ *occurring for the attributes* X *in the database* D*. The sensitivity for* $d_{\text{bound}}(D, X)$ *is 1.*

Proof. An FD violation can only happen to a tuple t with other tuples t' that share the same values for the attributes X. Let $\vec{a_X}^*$ be the most frequent value for X in D, i.e.,

$$\vec{a_X}^* = \text{argmax}_{\vec{a_X} \in \text{dom}(A_1) \times \ldots \times \text{dom}(A_k)} \text{freq}(D, \vec{a_X}).$$

In the worst case, a tuple t has the most frequent value $\vec{a_X}^*$ for X but has a different value in Y with all the other tuples with $X = \vec{a_X}^*$. Then the number of violations involved by t is $\text{freq}(D, \vec{a_X}^*) - 1$.

Adding a tuple or removing a tuple to a database will change, at most, one of the frequency values by 1. Hence, the sensitivity of the maximum frequency values is 1. □

Now, we will extend the analysis to multiple FDs.

Theorem 3. *Given a database D and a set of FDs $\Sigma = \{\sigma_1, \ldots, \sigma_l\}$, for its respective conflict graph $\mathcal{G}_\Sigma^D$, we have the maximum degree of the graph $d_{\max}(\mathcal{G}_\Sigma^D)$ upper bounded by*

$$d_{\text{bound}}(D, \Sigma) = \sum_{(\sigma: X \to Y) \in \Sigma} d_{\text{bound}}(D, X) \quad (11)$$

Proof. By Lemma 3, for each FD $\sigma : X \to Y$, a tuple may violate at most $d_{\text{bound}}(D, X)$ number of tuples. In the worst case, the same tuple may violate all FDs. □

We will spend some privacy budget ε_0 to perturb the upper bound $d_{\text{bound}}(D, X)$ for all FDs with LM and add them together. Each FD is assigned with $\varepsilon_0 / |\Sigma_{\text{FD}}|$, where Σ_{FD} is the set of FDs in Σ. We denote this perturbed upper bound as $\tilde{d}_{\text{bound}}$ and add it to the candidate set Θ if absent.

Extension to general DCs. The upper bound derived in Theorem 3 only works for FDs but fails for general DCs. General DCs have more complex operators, such as "greater/smaller than," in their formulas. Such inequalities require the computation of tuple-specific information, which is hard with DP. For example, consider the DC $\sigma : \neg(t_i[gain] > t_j[gain] \wedge t_i[loss] < t_j[loss])$ saying that if the gain for tuple t_i is greater than the gain for tuple t_j, then the loss for t_i should also be greater than t_j. We can observe that similar analyses for FDs do not work here as the frequency of a particular domain value in D does not bound the number of conflicts related to a tuple. Instead, we have to iterate each tuple t's gain value and find how many other tuples t's violate this gain value. In the worst case, such a computation may have a sensitivity equal to the data size. Therefore, estimation using DCs may result in much noise, especially when the dataset has fewer conflicts and the noise is added to correspond to the large sensitivity.

Our experimental study (Section 6) shows that datasets with general DCs have dense conflict graphs, which favors larger θs for graph projection. Hence, if we learn a small noisy upper bound $\tilde{d}_{\text{bound}}$ based on the FDs with LM, we will first prune all degree candidates smaller than $\tilde{d}_{\text{bound}}$, but then include $|V|$, which corresponds to the case when no edges are truncated, and Laplace mechanism is applied with the largest possible sensitivity $|V|$, i.e.,

$$\Theta' = \{\theta \in \Theta \mid \theta \leq \tilde{d}_{\text{bound}}\} \cup \{|V|\}. \quad (12)$$

Though the maximum value in Θ' is $|V|$, the sensitivity of the quality function over the candidate set Θ' remains $\tilde{d}_{\text{bound}}$. For the $|V|$ candidate, the quality function only depends on the Laplace error $\frac{\sqrt{2}|V|}{\varepsilon_2}$ and has no error from e_{bias} as no edges will be truncated.

In practice, one may skip this upper bound calculation process and skip directly to the two-step exponential mechanism if it is known that the graph is too dense or contains few FDs and more general DCs. We discuss this in detail in the experiments section.

θ	q	EM	2-EM ($\theta_1^* = \theta_3$)	2-EM ($\theta_1^* = \theta_2$)
1	−3.41	0.35	0.51	1
2	−3.82	0.33	0.49	-
3	−4.24	0.31	-	-

Table 4: Probabilities of choosing candidates with exponential mechanism (EM) vs two-step hierarchical exponential mechanism (2-EM). θ_1^* refers to the output of the first step of 2-EM.

Hierarchical EM. The upper bound d_{bound} may not be tight as it estimates the maximum degree in the worst case. The graph would be sparse with low degree values, and there is still room for pruning. To further prune candidate values in the set Θ, we use a hierarchical EM that first samples a degree value θ^* to prune values in Θ and then sample again another value θ^* from the remaining candidates as the final degree the graph projection. Our work uses a two-step hierarchical EM by splitting the privacy budget equally into halves. One may extend this EM to more steps at the cost of breaking their privacy budget more times, but in practice, we notice that a two-step is enough for a reasonable estimate.

Example 5. *Consider the same setup as Example 2. For this dataset, we start with $\Theta = [1, 2, 3]$ and the $\theta_{\max}$ for this setup is 3. Assume no values are pruned in the first optimization phase. We compare a single versus a two-step hierarchical EM for the second optimization step. From Table 4 in Example 3, we know that the θ_1 has the best quality. However, as the quality values are close, the probability of choosing the best candidate is similar, as shown in Table 4 with $\varepsilon = 1$. The exponential mechanism will likely choose a suboptimal candidate in such a scenario as the probabilities are close. But if a two-step exponential mechanism is used even with half budget $\varepsilon = 0.5$, the likelihood of choosing the best candidate θ_1 goes up to 0.51 if the first step chose θ_3 or 1 if the first step chosen θ_2.*

Incorporating the optimizations into the algorithm. Algorithm 4 outlines the two optimization techniques. First, we decide when to use the estimated upper bound for the maximum degrees, for example, when the constraint set Σ mainly consists of FDs. We will spend part of the budget ε_0 from ε_1 to perturb the upper bounds $d_{\text{bound}}(D, X)$ for all FDs with Laplace mechanism and add them together (lines 1-3). The noisy upper bound $\tilde{d}_{\text{bound}}$ is used to prune the candidate set (line 4). We also add $|V|$ to the candidate set if there are general DCs in Σ, and then set the sensitivity of the quality function $\theta_{\max}$ to be the minimum of the noisy upper bound or $|V|$ (line 5). Then we conduct the two-step hierarchical exponential mechanism for parameter selection (lines 6-10). Lines 7-8 work similarly to the previous exponential mechanism algorithm with

Algorithm 4: Optimized EM for parameter selection

Data: Graph $\mathcal{G}(V, E)$, candidate set $\Theta = \{1, \ldots, |V|\}$, quality function q, privacy budget $\varepsilon_1, \varepsilon_2$
Result: Candidate θ^*

1 **if** Σ *mainly consists of FDs* **then**
2 $\quad \varepsilon_0 \leftarrow \varepsilon_1/4$, $\varepsilon_1 \leftarrow \varepsilon_1 - \varepsilon_0$
3 $\quad$ Compute noisy upper bound $\tilde{d}_{\text{bound}} \leftarrow \sum_{\sigma: X \to Y} (d_{\text{bound}}(D, X) + \text{Lap}(|\Sigma_{\text{FD}}|/\varepsilon_0))$
4 Prune candidates $\Theta \leftarrow \{\theta \in \Theta \mid \theta \le \tilde{d}_{\text{bound}}\} \cup \{\tilde{d}_{\text{bound}}, |V|\}$
5 Set $\theta_{\max} \leftarrow \min(\tilde{d}_{\text{bound}}, |V|)$
6 **for** $s \in \{1, 2\}$ **do**
7 $\quad$ For each $\theta_i \in \Theta$, compute $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ // See Equation (2)
8 $\quad$ Sample θ^* with prob $\propto \exp(\frac{\frac{\varepsilon_1}{2} q_{\varepsilon_2}(\mathcal{G}, \theta_i)}{2\theta_{\max}})$
9 $\quad$ Prune candidates $\Theta \leftarrow \{\theta \in \Theta \mid \theta \le \theta^*\}$
10 $\quad$ Set $\theta_{\max} \leftarrow \theta^*$
11 **Return** θ^*

half of the remaining ε_1, where we choose a θ^* based on the quality function. However, instead of using it as the final candidate, we use it to prune values in Θ and improve the sensitivity $\theta_{\max}$ for the second exponential mechanism (lines 9-10). Then we repeat the exponential mechanism again, and output the sampled θ^* (line 11). Algorithm 4 has a similar complexity of $O(|\Theta|m)$ as Algorithm 3, where $|E|$ is the edge size of the graph. The overall Algorithm 2 has a complexity of $O(|\Sigma|n^2 + |\Theta|m)$.

Privacy and utility analysis. The privacy analysis of the optimizations with our algorithm depends on the analysis of three major steps: d_{bound} computation with Laplace mechanism, two-step exponential mechanism, and the final measure calculation with Laplace mechanism. By sequential composition, we have Theorem 4.

Theorem 4. *Algorithm 2 with the optimized EM in Algorithm 4 satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.*

The proof is similar as Theorem 1, except we have a tighter sensitivity analysis for the quality function in EM over the pruned candidate set. The sensitivity analysis is given by Lemma 4 and is used for $\theta_{\max}$ in line 8 of Algorithm 4.

Lemma 4. *The sensitivity of $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ in the 2-step EM (Algorithm 4) defined in Equation (2) is $\theta_{\max} = \min(\tilde{d}_{\text{bound}}, |V|)$ for 1st EM step and $\theta_{\max} = \theta^*$ for the 2nd EM step.*

The proofs for the theorem and lemma are at Appendix A.2.4 and Appendix A.2.5.

The utility analysis in Theorem 2 for Algorithm 2 with the basic EM (Algorithm 3) still applies to the optimized EM (Algorithm 4). The basic EM usually has $\theta_{\max} = |D| = |V|$ and the full budget ε_1, while the optimized EM has a much smaller $\theta_{\max}$ and slightly lower privacy budget when the graph is sparse. In practice (Section 6), we see significant utility improvements by the optimized EM for sparse graphs. When the graph is dense, we see the utility degrade slightly due to less budget used for each EM. However, the degradation is negligible with respect to the true inconsistency measure.

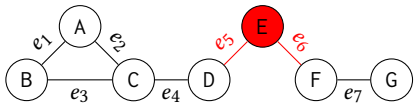


Figure 2: Toy graph example $\mathcal{G}$ with seven nodes (A to G) and seven edges. Consider a neighboring graph $\mathcal{G}'$ with the differing node E (red) and its two edges.

Algorithm 5: DP approximation of minimum vertex cover size for $\mathcal{I}_{\text{R}}$

Data: Graph $\mathcal{G}(V, E)$, stable global ordering Λ, privacy parameter ε
Result: DP minimum vertex cover size for $\mathcal{I}_{\text{R}}$

1 Initialize vertex cover set $C = \emptyset$ and size $c = 0$
2 Initialize edge list $E_0 = E$
3 **for** $i \in \{1 \ldots |\Lambda|\}$ **do**
4 $\quad$ pop edge $e_i = \{u, v\}$ in order from Λ
5 $\quad$ add u and v to C and $c = c + 2$
6 $\quad$ E_{i+1} = remove all edges incident to u or v from E_i
7 **Return** $c + \text{Lap}(\varepsilon/2)$

5 DP Minimum Vertex Cover for $\mathcal{I}_{\text{R}}$

This section details our approach for computing the optimal repair measure, $\mathcal{I}_{\text{R}}$, using the conflict graph. $\mathcal{I}_{\text{R}}$ is defined as the minimum number of vertices that have to be removed to eliminate all conflicts within the dataset. For the conflict graph $\mathcal{G}_{\Sigma}^{D}$, this corresponds to finding the minimum vertex cover – an NP-hard problem. To address this, we apply a well-known polynomial-time algorithm that provides a 2-approximation for vertex cover [59]. This randomized algorithm iterates through a random ordering of edges, adding both nodes of each edge to the vertex cover if they haven't been encountered, then removes all incident edges. The process repeats until the edge list is exhausted. In our setting, we aim to compute the minimum vertex cover size while satisfying DP. A straightforward approach would be to analyze the sensitivity of the 2-approximation algorithm and add the appropriate DP noise. However, determining the sensitivity of this naive approximation is challenging, as the algorithm's output can fluctuate significantly depending on the order of selected edges. This variability is illustrated in Example 6.

Example 6. *Let us consider a graph $\mathcal{G}$ with 7 vertices A to G and 7 edges e_1 to e_7 as shown in Figure 2. We can have a neighboring graph $\mathcal{G}'$ by considering the vertex E as the differing vertex and two of its edges e_5 and e_6 as the differing edges. This example shows that according to the vanilla 2-approximate algorithm, the output for the graphs $\mathcal{G}$ and $\mathcal{G}'$ may vary drastically. For $\mathcal{G}$, if e_2 is selected followed by e_7, then the vertex cover size is 4. However, for graph $\mathcal{G}'$, if e_1 or e_4 is selected first and subsequently after the other one e_6 is selected, then the output is 6. Moreover, this difference may get significantly large if the above graph is stacked multiple times and the corresponding vertex that creates this difference is chosen every time.*

To solve the sensitivity issue, we make a minor change in the algorithm by traversing the edges in a particular order (drawing on [12]). We use a similar stable ordering Λ defined in Section 2.3. The new algorithm is shown in Algorithm 5. We start by initializing

an empty vertex cover set C, its size c, and an edge list (lines 1–2). We then start an iteration over all edges in the same ordering as the stable ordering Λ (line 3). For each edge $e_i = \{u, v\} \in E$ that is part of the graph, we add both u and v to C and correspondingly increment the size c (lines 4–5). We also remove all other edges, including e_i, that are connected to u or v from E and continue the iteration (line 5). Finally, we return the noisy size of the vertex cover (line 6). The sensitivity of this algorithm is given by Proposition 3.

Proposition 3. *Algorithm 5 obtains a vertex cover, and its size has a sensitivity of 2.*

The proof can be found in Appendix A.3.2.

Example 7. *Let us consider our running example in Figure 2 as input to Algorithm 5 and use it to understand the proof. We have two graphs – $\mathcal{G}$ which has 6 vertices $V = [\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{F}, \mathsf{G}]$ and edges $E = [e_1, e_2, e_3, e_4, e_7]$ and $\mathcal{G}'$ has 7 vertices $V' = [\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}]$ and edges $E' = [e_1, e_2, \ldots, e_7]$. The total possible number of edges is $\binom{7}{2} = 21$, and we can have a global stable ordering of the edges Λ depending on the lexicographical ordering of the vertices as $e_1, e_2, e_3, \ldots, e_{21}$. When the algorithm starts, both vertex cover sizes are initialized to $c = 0, c' = 0$, and the algorithm's state is in Case 1 with $v^* = \mathsf{E}$. We delineate the next steps of the algorithm below:*

- *Iteration 1 (Subcase 1b) : $e_1(\mathsf{A}, \mathsf{B})$ is chosen. A and B are both in E_0 and E'_0. Hence $c = 2, c' = 2$.*
- *Iteration 2 and 3 (Subcase 1c) : $e_2(\mathsf{A}, \mathsf{C})$ and $e_3(\mathsf{B}, \mathsf{C})$ are chosen. Both are removed in iteration 1. Hence $c = 2, c' = 2$.*
- *Iteration 4 (Subcase 1b) : $e_4(\mathsf{C}, \mathsf{D})$ is chosen. C and D are both in E_3 and E'_3. Hence $c = 4, c' = 4$.*
- *Iteration 5 (Subcase 1c) : $e_5(\mathsf{D}, \mathsf{E})$ is chosen. It's removed from E'_4 in iteration 4 and was never present in E. Hence $c = 4, c' = 4$.*
- *Iteration 6 (Subcase 1a) : $e_6(\mathsf{E}, \mathsf{F})$ is chosen. It's in E'_5 but not in E_5. Hence, $c = 4, c' = 6$ and the new $v^* = \mathsf{E}$.*
- *Iteration 7 (Subcase 2a) : $e_7(\mathsf{F}, \mathsf{G})$ is chosen. It's It's in E_6 but removed from E_6 in Iteration 6. Hence, $c = 6, c' = 6$ and the algorithm is complete.*

Privacy and utility analysis. We now show the privacy and utility analysis of Algorithm 5 using Theorem 5 below.

Theorem 5. *Algorithm 5 satisfies ε-node DP and always outputs the size of a 2-approximate vertex cover of graph $\mathcal{G}$.*

The privacy analysis of Algorithm 5 is straightforward as we calculate the private vertex cover using the Laplace mechanism with sensitivity 2 according to Proposition 3. The utility analysis can be derived from the original 2-approximation algorithm. The stable ordering Λ in Algorithm 5 can be perceived as one particular random order of the edges and hence has the same utility as the original 2-approximation algorithm.

6 Experiments

In this section, we present our experiment results on computing the three measures outlined in Section 4 and Section 5. The questions that we ask through our experiments are as follows:

(1) How far are the private measures from the true measures?
(2) How do the different strategies for the degree truncation bound compare against each other?

Dataset	#Tuple	#Attrs	#DCs(#FDs)	Max Deg with 1% RNoise
Adult [3]	32561	15	3 (2)	9635
Flight [49]	500000	20	13 (13)	1520
Hospital [54]	114919	15	7 (7)	793
Stock [50]	122498	7	1 (1)	1
Tax [10]	1000000	15	9 (7)	373

Table 5: Description of datasets. The max deg column shows the maximum degree of any node in a $10k$ rows subset of the conflict graph of the dataset with 1% RNoise.

(3) How do our methods perform with different privacy budgets?

6.1 Experimental Setup

All our experiments are performed on a server with Intel Xeon Platinum 8358 CPUs (2.60GHz) and 1 TB RAM. Our code is in Python 3.11 and can be found in the artifact submission. All experiments are repeated for 10 runs, and the mean error value is reported.

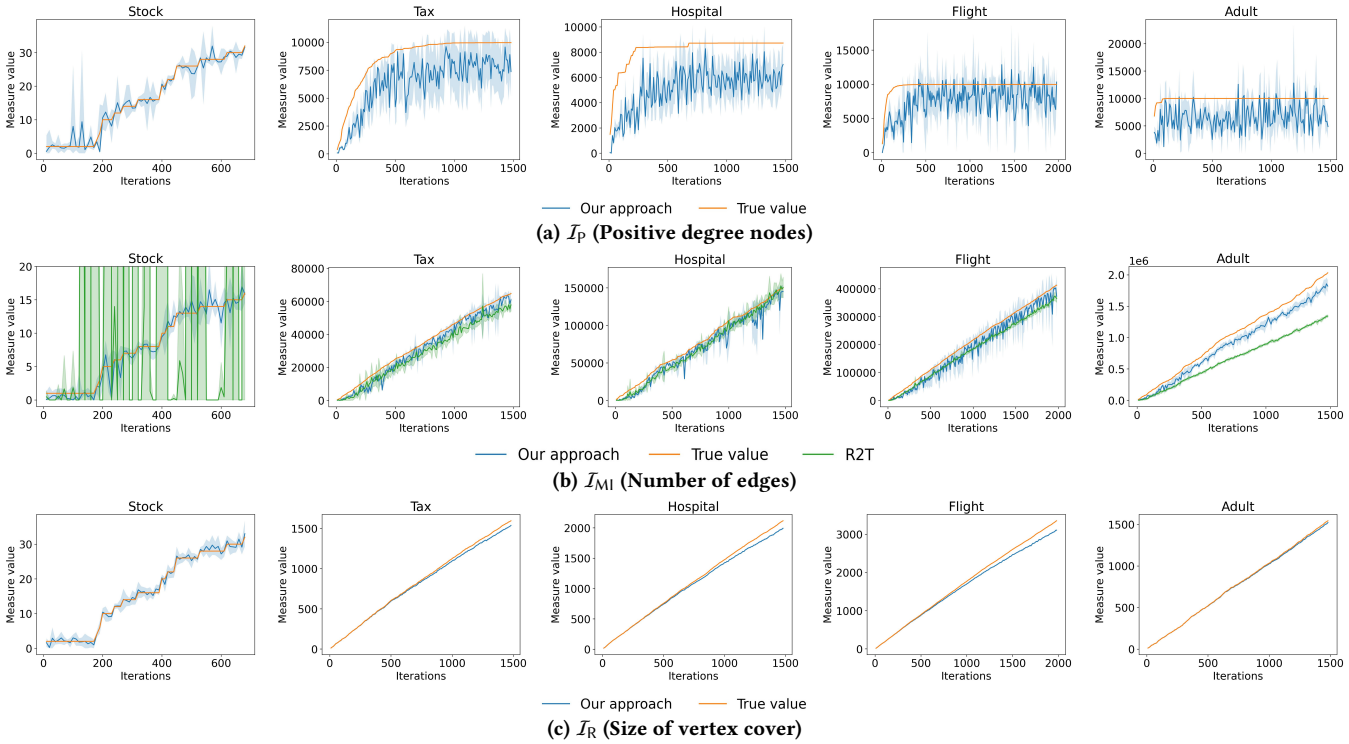
Datasets and violation generation. We conduct experiments on five real-life datasets and their corresponding DCs as described in Livshits et al. [41].

- Adult [3]: Annual income results from various factors.
- Flight [49]: Flight information across the US.
- Hospital [54]: Information about different hospitals across the US and their services.
- Stock [50]: Trading stock information on various dates
- Tax [10]: Personal tax infomation.

The datasets used in our as described in Table 5 are initially consistent with the constraints. We replicate the exact setup as Livshits et al. [41] for experimentation. All experiments are done on a subset of $10k$ rows, and violations are added in the same manner, namely CONoise (for Constraint-Oriented Noise) and RNoise (for Random Noise). CONoise introduces random violations of the constraints by running 200 iterations of the following procedure:

(1) Randomly select a constraint σ from the constraint set Σ.
(2) Randomly select two tuples t_i and t_j from the database.
(3) For every predicate $\phi = (t_i[a_1] \circ t_j[a_2])$ of σ:
   - If t_i and t_j jointly satisfy ϕ, continue to the next predicate.
   - If $\circ \in \{=, \leq, \geq\}$, change either $t_i[a_1]$ or $t_j[a_2]$ or vice versa (the choice is random).
   - If $\circ \in \{<, >, \neq\}$, change either $t_i[a_1]$ or $t_2[a_2]$ (the choice is again random) to another value from the active domain of the attribute such that ϕ is satisfied, if such a value exists, or a random value in the appropriate range otherwise.

The second algorithm, RNoise, is parameterized by the parameter α that controls the level of noise by modifying α of the values in the dataset. At each iteration of RNoise, we randomly select a database cell corresponding to an attribute that occurs in at least one constraint. Then, we either change its value to another value from the active domain of the corresponding attribute (with probability 0.5) or to a typo. The datasets vary immensely in the density of their conflict graphs as described in the max degree column of Table 5. For example, the Adult $10k$ nodes subset has a maximum degree of 9635, whereas the Stock dataset has a maximum of 1 with the same amount of conflict addition.



(a) $\mathcal{I}_P$ (Positive degree nodes)

(b) $\mathcal{I}_{MI}$ (Number of edges)

(c) $\mathcal{I}_R$ (Size of vertex cover)

Figure 3: True vs Private estimates for all dataset with RNoise $\alpha = 0.01$ at $\varepsilon = 1$. The $\mathcal{I}_P$ measure (figure a) and $\mathcal{I}_{MI}$ measure (figure b) are computed using our graph projection approach, and $\mathcal{I}_R$ measure using our private vertex cover size approach.

Metrics. Following Livshits et al. [41], we randomly select a subset of 10k rows of each dataset, add violations to the subset, and compute the inconsistency measures on the dataset with violations. To measure performance, we utilize the normalized ℓ_1 distance error [17], $|\mathcal{I}(D,\Sigma) - \mathcal{M}(D,\Sigma,\varepsilon)|/\mathcal{I}(D,\Sigma)$, where $\mathcal{M}(D,\Sigma,\varepsilon)$ represents the estimated private value of the measure and $\mathcal{I}(D,\Sigma$ denotes the true value. For the $\mathcal{I}_R$ measure, we use the linear approximation algorithm from Livshits et al. [41] to estimate the non-private value.

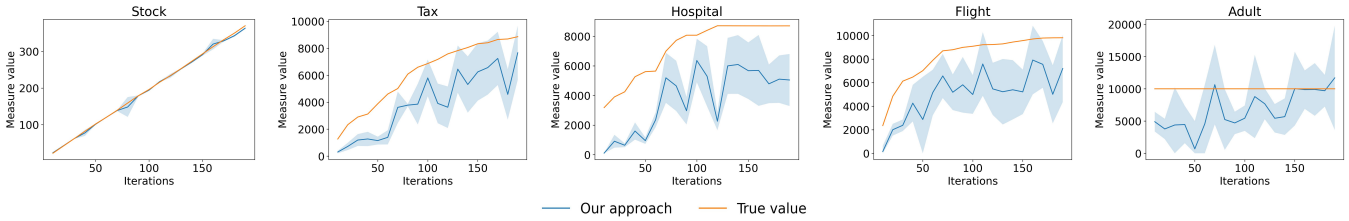
Algorithm variations. We experiment with multiple different variations of Algorithm 2 for $\mathcal{I}_{MI}$ and $\mathcal{I}_P$. The initial candidate set for the degree bound is $\Theta = [1, 5, 10, 100, 500, 1000, 2000, 3000, \ldots, 10000]$ with multiples of 1000 along with some small candidates.

- *Baseline 1*: naively sets the bound θ^* to the maximum possible degree $|V|$ in Algorithm 2 by skipping line 2 and the unused privacy budget ε_1 is used for the final noise addition step.
- *Baseline 2*: sets the bound θ^* to the actual maximum degree of the conflict graph $d_{\max}(\mathcal{G}_\Sigma^D)$. Note this is a non-private baseline that only acts as an upper bound as one of the best values that can be achieved without considering privacy constraints.
- *Exponential mechanism*: choose θ^* over the complete candidate set Θ using the basic EM in Algorithm 3.
- *Hierarchical exponential mechanism*: chooses θ^* using a two-step EM with an equal budget for each step in Algorithm 4, but skipping Lines 1-5 of the upper bound computation step.
- *Upper bound + hierarchical exponential mechanism (our approach)*: encompasses both the optimization strategies including the upper bound computation and the hierarchical exponential mechanism discussed in Section 4.2 (the full Algorithm 4).

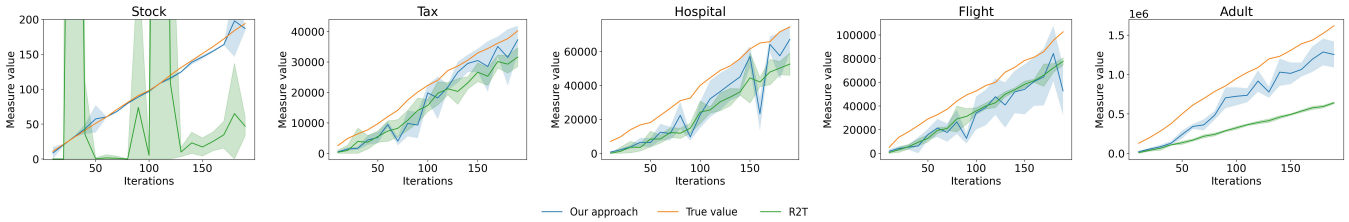
By default, we experiment with a total privacy budget of $\varepsilon = 1$ unless specified otherwise.

6.2 Results

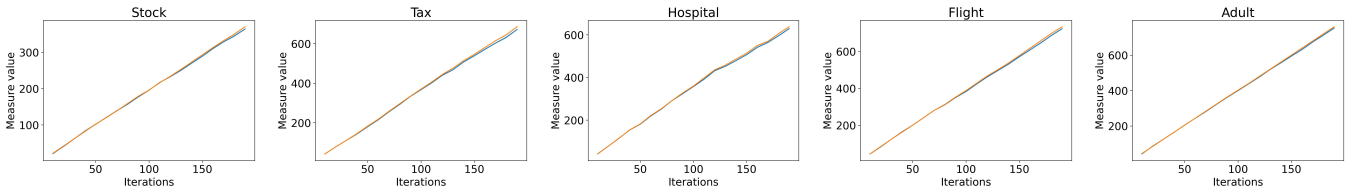
6.2.1 True vs private estimation. In Figures 3 and 4, we plot the true vs private estimates at $\varepsilon = 1$ for all datasets with RNoise ($\alpha = 0.01$) and CONoise (200 iterations) respectively. The datasets are ordered according to their densities from left to right. Each figure contains the measured value ($\mathcal{I}_{MI}$, $\mathcal{I}_P$, or $\mathcal{I}_R$) on the Y-axis and the number of iterations on the X-axis. For the CONoise, the number of iterations is set to 200 for every dataset, and for RNoise, the iterations correspond to the number of iterations required to reach 1% ($\alpha = 0.01$) number of random violations. The orange line corresponds to the true value of the measure, and the blue line corresponds to the private measure using our approach. For $\mathcal{I}_{MI}$ and $\mathcal{I}_P$ measures, the blue line represents the upper bound + hierarchical exponential mechanism strategy as described in Section 4.2 along with its standard deviation in shaded blue. For the $\mathcal{I}_R$ measure, it represents the private minimum vertex cover size algorithm. We also add a baseline approach using a state-of-the-art private SQL approach called R2T [14]. This baseline is different from the experiment in Table 1 as we write all the constraints into one query instead of naively splitting the privacy budget for each constraint. We add this baseline only for the $\mathcal{I}_{MI}$ measure as $\mathcal{I}_R$ cannot be written with SQL and $\mathcal{I}_P$ requires the DISTINCT/GROUP BY clause



(a) $\mathcal{I}_{\mathrm{P}}$ (Positive degree nodes)

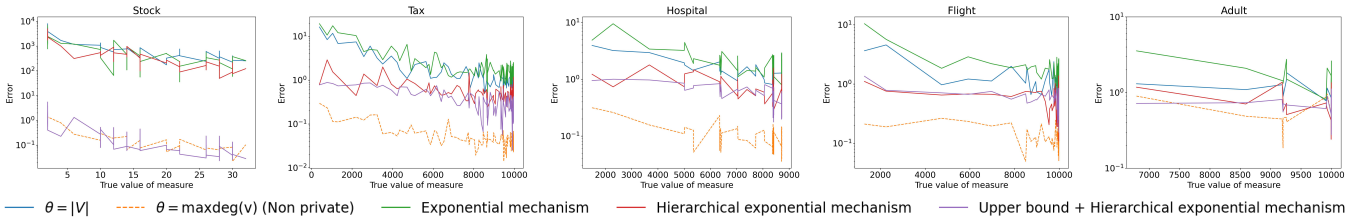


(b) $\mathcal{I}_{\mathrm{MI}}$ (Number of edges)

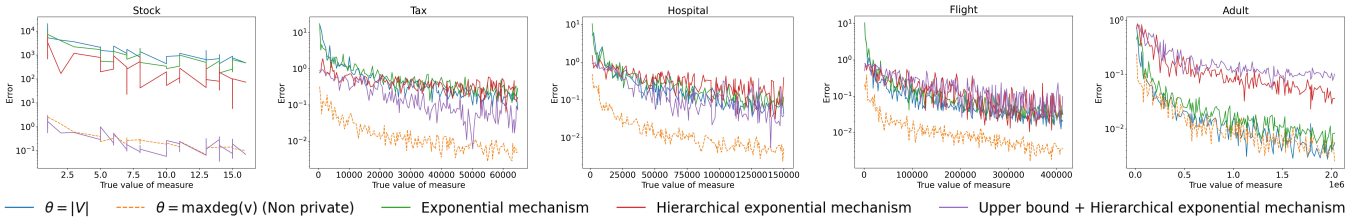


(c) $\mathcal{I}_{\mathrm{R}}$ (Size of vertex cover)

Figure 4: True vs Private estimates for all dataset with CONoise at $\varepsilon = 1$ for 200 iterations



(a) $\mathcal{I}_{\mathrm{P}}$ (Positive degree nodes)



(b) $\mathcal{I}_{\mathrm{MI}}$ (Number of edges)

Figure 5: Computing different strategies for choosing θ for all datasets with RNoise at $\alpha = 0.01$ and $\varepsilon = 1$. The datasets are arranged according to their densities from sparsest (left) to densest (right).

that R2T does not support. Based on the experiments, we draw three major observations.

First, compared to the SQL baseline (R2T), our approach has a better relative error in average across all datasets. However, R2T is slightly behind for moderate to high dense datasets such as Tax (0.207 vs 0.334), Hospital (0.209 vs 0.386), and Flight(0.202 vs 0.205) and Adult (0.187 vs 0.269) but fails miserably for sparse datasets such as Stock (0.492 vs 137.05). This is because the true value of the measure is small and R2T adds large amounts of noise.

Second, our approach for the $\mathcal{I}_{\mathrm{MI}}$ and $\mathcal{I}_{\mathrm{P}}$ fluctuates more and has a higher standard deviation compared to the $\mathcal{I}_{\mathrm{R}}$ measure. This is because of the privacy noise due to the relatively high sensitivity of our upper bound + hierarchical exponential mechanism approach. On the other hand, the vertex cover size approach for $\mathcal{I}_{\mathrm{R}}$ has a sensitivity equal to 2 and, therefore, does not show much fluctuation when the true measure value is large enough.

Third, we observe that our approach generally performs well across all five datasets and all inconsistency measures. The $\mathcal{I}_{\mathrm{MI}}$

and $\mathcal{I}_{\mathsf{P}}$ measures have average errors of 0.25 and 0.46, respectively, across all datasets where Stock is the worst performing dataset for $\mathcal{I}_{\mathsf{MI}}$ and Adult is the worst performing for $\mathcal{I}_{\mathsf{P}}$. The $\mathcal{I}_{\mathsf{R}}$ performs the best with an average error of 0.08, with Stock as the worst-performing dataset. We investigate the performance of each dataset in detail in our next experiment and find out that the density of the graphs plays a significant role in the performance of our algorithms.

6.2.2 Comparing different strategies for choosing θ. In Figure **??**, we present the performance of different algorithm variations in computing the $\mathcal{I}_{\mathsf{P}}$ and $\mathcal{I}_{\mathsf{MI}}$ measures for all datasets using RNnoise at $\alpha = 0.01$ and $\varepsilon = 1$. The y-axis in each figure shows the logarithmic scaled error, while the x-axis displays the actual measure value, with different colors representing the strategies. The graphs are ordered from most sparse (Stock) to least sparse (Adult) to compare methods for choosing the θ value at $\varepsilon = 1$ ($\varepsilon_1 = 0.4$, $\varepsilon_2 = 0.6$). The methods include all variations described in the algorithm variation section.

We note all the strategies are private except baseline 2 (orange dash line) that sets θ as the true maximum degree of the conflict graph.

Our experimental results reveal several observations. First, we noticed that across all datasets, the initial error is higher at smaller iterations due to privacy noise dominating the true value. For the sparsest dataset (Stock), all strategies showed errors in the magnitude 3-4 larger, except for the non-private baseline (orange) and the upper bound + hierarchical exponential mechanism (purple). This is because the candidate set contains many large candidates, and pruning it with the upper bound strategy yields more meaningful results.

For moderately sparse graphs (Tax and Hospital), our upper bound + hierarchical mechanism (purple) consistently outperformed the other private methods, although the two-step hierarchical exponential mechanism (red) also demonstrated competitive performance. The red method, while underperforming for Stock, showed comparable results within a 1-magnitude error difference for these datasets, indicating that upper bound estimation is less crucial when the true max degree is not excessively low.

Finally, for dense graphs (Flight and Adult), the optimized exponential mechanisms (red and purple) outperformed private baselines (blue and green) in the $\mathcal{I}_{\mathsf{P}}$ measure but did not surpass even the naive baseline (blue) for the $\mathcal{I}_{\mathsf{MI}}$ measure. This is due to the optimal degree bound value being close to the largest possible value $|V|$, limiting the efficiency of the pruning strategy. Despite this, the relative errors remained small, and the noisy answers preserved the order of the true measures (shown in previous experiments in Figures 3 and 4).

6.2.3 Varying privacy budget. Figure 6 illustrates how our proposed DP algorithms perform at varying privacy budgets with $\varepsilon \in [0.1, 0.2, 0.5, 1.0, 2.0, 3.0, 5.0]$. The rightmost figure for the repair measure $\mathcal{I}_{\mathsf{R}}$ has a log scale on the y-axis for better readability. We experiment with three datasets of different density properties (sparsest Stock and densest Flight) and show that our algorithm gracefully scales with the ε privacy budget for all three inconsistency measures. We also observe that the algorithm has a more significant error variation at smaller epsilons (< 1) except for Stock, which has a larger variation across all epsilons. This happens when

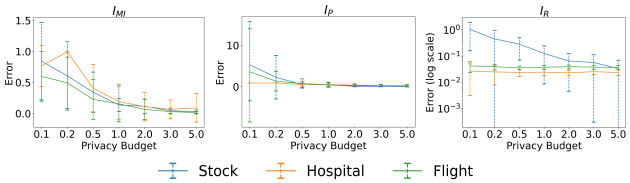


Figure 6: Computing inconsistency measures for different datasets with RNoise at $\alpha = 0.005$ and varying privacy budgets

the true value for this measure is small, and adding noise at a smaller budget ruins the estimate drastically. For the $\mathcal{I}_{\mathsf{R}}$ measure, the private value reaches with 0.05 error at $\varepsilon = 3$ for Stock and as early as $\varepsilon = 0.1$ for others.

7 Related Work

We survey relevant works in the fields of DP and data repairing.

Differential privacy. DP has been studied in multiple settings [31, 32, 38, 46, 57, 60], including systems that support complex SQL queries that, in particular, can express integrity constraints [14]. The utilization of graph databases under DP has also been thoroughly explored, with both edge privacy [29, 33–35, 56, 62] and node privacy [7, 12, 36]. Our approach draws on [12] to allow for efficient DP computation of the inconsistency measures over the conflict graph, whereas we have seen worse performance from alternative approaches for releasing graph statistics that tend to aggresively truncate edges or nodes. In the context of data quality, previous work [40] has proposed a framework for releasing a private version of a database relation for publishing, supporting specific repair operations, while more recent, work [22] provides a DP synthetic data generation mechanism that considers soft DCs [9].

Data repair. Various classes of constraints have been proposed by the literature, including FDs, conditional FDs [8], metric FDs [39], and DCs [9]. Our work focuses on DCs which are a general class of integrity constraints that subsumes all of the aforementioned constraints. While computing the minimal data repair in some cases has been shown to be polynomial time [44], computing the minimal repair in most of the general cases, corresponding to $\mathcal{I}_{\mathsf{R}}$, are NP-hard. Therefore, a prominent vein of research has been devoted to utilizing these constraints for data repairing [2, 5, 11, 20, 23, 24, 44, 55]. The repair model in these works varies between several options: tuple deletion, cell value updates, tuple addition, and combinations thereof. The process of data repairing through such algorithms can be time consuming due to the size of the data and the size and complexity of the constraint set. Hence, previous work [41, 45] proposed to keep track of the repairing process and ensure that it progresses correctly using inconsistency measures. In our work, we capitalize on the suitability of these measures to DP as they provide an aggregate form that summarizes the quality of the data for a given set of constraints.

8 Conclusions

We proposed a new problem of studying inconsistency measures for private datasets in the DP setting. We studied five measures and showed that two are intractable with DP, and the other three face a significant challenge of high sensitivity. To solve this challenge, we leveraged the dataset's corresponding conflict graph and used

graph projection and a novel approximate DP vertex cover size algorithm to accurately estimate the private inconsistency measures. To test our algorithm, we experimented with five real-world datasets with varying density properties and showed that our algorithm could accurately calculate these measures across all datasets. Our future work includes studying the problem of private inconsistency measures with different privacy notions, such as k-anonymity, to implement the other two inconsistency measures intractable with DP and use these private inconsistency measures in real-world data cleaning applications.

References

[1] John M Abowd. 2018. The US Census Bureau adopts differential privacy. In *SIGKDD*. 2867–2867.

[2] Foto N. Afrati and Phokion G. Kolaitis. 2009. Repair Checking in Inconsistent Databases: Algorithms and Complexity. In *ICDT*. 31–41.

[3] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[4] Leopoldo E. Bertossi. 2018. Measuring and Computing Database Inconsistency via Repairs. In *SUM (LNCS, Vol. 11142)*. Springer, 368–372.

[5] Leopoldo E. Bertossi, Solmaz Kolahi, and Laks V. S. Lakshmanan. 2013. Data Cleaning and Query Answering with Matching Dependencies and Matching Functions. *Theory Comput. Syst.* 52, 3 (2013), 441–482.

[6] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient Denial Constraint Discovery with Hydra. *Proc. VLDB Endow.* 11, 3 (2017), 311–323.

[7] 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.

[8] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *ICDE*.

[9] Jan Chomicki and Jerzy Marcinkowski. 2005. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.* 197, 1-2 (2005), 90–121.

[10] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1498–1509.

[11] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *ICDE*. 458–469.

[12] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *SIGMOD*. 123–138.

[13] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems* 30 (2017).

[14] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. 2022. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *SIGMOD*. 759–772.

[15] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Vol. 4052. 1–12.

[16] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, Serge Vaudenay (Ed.), Vol. 4004. Springer, 486–503.

[17] 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.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2016. Calibrating Noise to Sensitivity in Private Data Analysis. *J. Priv. Confidentiality* 7, 3 (2016), 17–51. https://doi.org/10.29012/jpc.v7i3.405

[19] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[20] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. 2015. Dichotomies in the Complexity of Preferred Repairs. In *PODS*. 3–15.

[21] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. 2021. Kamino: Constraint-Aware Differentially Private Data Synthesis. *Proc. VLDB Endow.* 14, 10 (2021), 1886–1899. https://doi.org/10.14778/3467861.3467876

[22] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. 2021. Kamino: Constraint-Aware Differentially Private Data Synthesis. *Proc. VLDB Endow.* 14, 10 (2021), 1886–1899.

[23] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC Data-Cleaning Framework. *Proc. VLDB Endow.* 6, 9 (2013), 625–636.

[24] Amir Gilad, Daniel Deutch, and Sudeepa Roy. 2020. On Multiple Semantics for Declarative Database Repairs. In *SIGMOD*. 817–831.

[25] John Grant and Anthony Hunter. 2013. Distance-Based Measures of Inconsistency. In *ECSQARU (LNCS, Vol. 7958)*. Springer, 230–241.

[26] John Grant and Anthony Hunter. 2023. Semantic inconsistency measures using 3-valued logics. *Int. J. Approx. Reason.* 156 (2023), 38–60.

[27] Jerrold R. Griggs, Charles M. Grinstead, and David R. Guichard. 1988. The number of maximal independent sets in a connected graph. *Discret. Math.* 68, 2-3 (1988), 211–220. https://doi.org/10.1016/0012-365X(88)90114-8

[28] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially private combinatorial optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1106–1125.

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

[30] Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan Ullman. 2014. Privately solving linear programs. In *ICALP*. Springer, 612–624.

[31] 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.

[32] Noah M Johnson, Joseph P Near, and Dawn Xiaodong Song. 2017. Practical differential privacy for SQL queries using elastic sensitivity. *CoRR, abs/1706.09479* (2017).

[33] 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.

[34] Vishesh Karwa and Aleksandra B Slavković. 2012. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2012, Palermo, Italy, September 26-28, 2012. Proceedings*. Springer, 273–285.

[35] Vishesh Karwa, Aleksandra B Slavković, and Pavel Krivitsky. 2014. Differentially private exponential random graphs. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2014, Ibiza, Spain, September 17-19, 2014. Proceedings*. Springer, 143–155.

[36] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2013. Analyzing Graphs with Node Differential Privacy. In *TCC*, Amit Sahai (Ed.), Vol. 7785. Springer, 457–476. https://doi.org/10.1007/978-3-642-36594-2_26

[37] Benny Kimelfeld, Ester Livshits, and Liat Peterfreund. 2020. Counting and enumerating preferred database repairs. *Theor. Comput. Sci.* 837 (2020), 115–157. https://doi.org/10.1016/j.tcs.2020.05.016

[38] 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.

[39] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. 2009. Metric functional dependencies. In *ICDE*.

[40] Sanjay Krishnan, Jiannan Wang, Michael J. Franklin, Ken Goldberg, and Tim Kraska. 2016. PrivateClean: Data Cleaning and Differential Privacy. In *SIGMOD*. ACM, 937–951. https://doi.org/10.1145/2882903.2915248

[41] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. 2020. The Shapley Value of Tuples in Query Answering. In *ICDT*, Vol. 155. 20:1–20:19.

[42] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. 2020. Approximate Denial Constraints. *Proc. VLDB Endow.* 13, 10 (2020), 1682–1695.

[43] Ester Livshits and Benny Kimelfeld. 2022. The Shapley Value of Inconsistency Measures for Functional Dependencies. *Log. Methods Comput. Sci.* 18, 2 (2022).

[44] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2020. Computing Optimal Repairs for Functional Dependencies. *ACM Trans. Database Syst.* 45, 1 (2020), 4:1–4:46. https://doi.org/10.1145/3360904

[45] Ester Livshits, Rina Kochirgan, Segev Tsur, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. 2021. Properties of Inconsistency Measures for Databases. In *SIGMOD*. 1182–1194.

[46] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.* 11, 10 (2018), 1206–1219.

[47] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.

[48] John W Moon and Leo Moser. 1965. On cliques in graphs. *Israel journal of Mathematics* 3, 1 (1965), 23–28.

[49] Department of Transportation. 2020. Flight. Research and Innovative Technology Administration. https://data.transportation.gov/d/7fzd-cqts.

[50] Oleh Onyshchak. 2020. Stock Market Dataset. https://doi.org/10.34740/KAGGLE/DSV/1054465

[51] Francesco Parisi and John Grant. 2019. Inconsistency measures for relational databases. *arXiv preprint arXiv:1904.03403* (2019).

[52] Shweta Patwa, Danyu Sun, Amir Gilad, Ashwin Machanavajjhala, and Sudeepa Roy. 2023. DP-PQD: Privately Detecting Per-Query Gaps In Synthetic Data Generated By Black-Box Mechanisms. *Proc. VLDB Endow.* 17, 1 (2023), 65–78.

[53] Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. 2021. Fast Detection of Denial Constraint Violations. *Proc. VLDB Endow.* 15, 4 (2021), 859–871.

[54] CORGIS Dataset Project. [n. d.]. Hospitals CSV File. https://corgis-edu.github.io/corgis/csv/hospitals/.

[55] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201.

[56] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 81–98.

[57] Yuchao Tao, Xi He, Ashwin Machanavajjhala, and Sudeepa Roy. 2020. Computing local sensitivities of counting queries with joins. In *SIGMOD*. 479–494.

[58] Matthias Thimm. 2017. On the compliance of rationality postulates for inconsistency measures: A more or less complete picture. *KI-Künstliche Intelligenz* 31 (2017), 31–39.

[59] Vijay V Vazirani. 1997. Approximation algorithms. *Georgia Inst. Tech* (1997).

[60] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2019. Differentially private sql with bounded user contribution. *arXiv preprint arXiv:1909.01917* (2019).

[61] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018). arXiv:1802.06739 http://arxiv.org/abs/1802.06739

[62] 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.

[63] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.

[64] Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. PrivTree: A Differentially Private Algorithm for Hierarchical Decompositions. In *SIGMOD*. 155–170.

A Theorems and Proofs

A.1 Proof for Proposition 2

Recall the proposition states that given a database D and a set of DCs Σ, where $|D| = n$, the following holds:

(1) The global sensitivity of $\mathcal{I}_{\mathsf{D}}$ is 1.
(2) The global sensitivity of $\mathcal{I}_{\mathsf{MI}}$ is n.
(3) The global sensitivity of $\mathcal{I}_{\mathsf{P}}$ is n.
(4) The global sensitivity of $\mathcal{I}_{\mathsf{MC}}$ is exponential in n.
(5) The global sensitivity of $\mathcal{I}_{\mathsf{R}}$ is 1.

Proof. Consider two neighbouring datasets, D and D'.

$\mathcal{I}_{\mathsf{D}}$. Adding or removing one tuple from the dataset will affect the addition or removal of all conflicts related to it in the dataset. In the worst case, this tuple could remove all conflicts in the dataset, and the $\mathcal{I}_{\mathsf{D}}$ could go from 1 to 0 or vice versa.

$\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$. The $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ are concerned with the set of minimally inconsistent subsets $MI_\Sigma(D)$. The $\mathcal{I}_{\mathsf{MI}}$ measure computes the total number of inconsistent subsets $|MI_\Sigma(D)|$ and $\mathcal{I}_{\mathsf{P}}$ computes the total number of unique rows participating in $MI_\Sigma(D)$, $|\cup MI_\Sigma(D)|$. Now, without loss in generality, let's assume D' has an additional tuple compared to D. In the worst case, the extra row could violate all other rows in the dataset, adding $|D|$ inconsistent subsets to $MI_\Sigma(D)$. Therefore, in the worst case, the $\mathcal{I}_{\mathsf{MI}}$ measure and $\mathcal{I}_{\mathsf{P}}$ could change by $|D|$.

$\mathcal{I}_{\mathsf{MC}}$. The $\mathcal{I}_{\mathsf{MC}}$ measure is #P-complete and can be computed for a conflict graph $\mathcal{G}_\Sigma^D$ if the dataset has only FDs in the constraint set Σ and $\mathcal{G}_\Sigma^D$ is P_4-free [37]. The maximum number of maximal independent sets [27, 48] $f(n)$ for a graph with n vertices is given by : If $n \geqq 2$, then $f(n) = \begin{cases} 3^{n/3}, & \text{if } n \equiv 0(\text{mod}3); \\ 4.3^{[n/3]-1}, & \text{if } n \equiv 1(\text{mod}3); \\ 2.3^{[n/3]}, & \text{if } n \equiv 2(\text{mod}3). \end{cases}$

Using the above result, we can see that adding or removing a node in the graph can affect the total number of maximal independent sets in the order of 3^n.

$\mathcal{I}_{\mathsf{R}}$. Without loss in generality, let's assume D' has an additional tuple compared to D. This extra row may or may not add extra violations. However, repairing this extra row in D' will always remove these added violations. Therefore, $\mathcal{I}_{\mathsf{R}}$ for D' can only be one extra than D. □

A.2 DP Analysis for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$(Algorithm 2)

A.2.1 Proof for Theorem 1. The theorem states that Algorithm 2 satisfies $(\varepsilon_1 + \varepsilon_2)$-node DP for $\mathcal{G}_\Sigma^D$ and $(\varepsilon_1 + \varepsilon_2)$-DP for the input database D.

Proof. The proof is straightforward using the composition property of DP Proposition 1. □

A.2.2 Proof for Lemma 1. Lemma 1 states that the sensitivity of the quality function $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ in Algorithm 3 defined in Equation (2) is $\theta_{\max} = \max(\Theta)$.

Proof. We prove the lemma for the $\mathcal{I}_{\mathsf{MI}}$ measure and show that it is similar for $\mathcal{I}_{\mathsf{P}}$. Let us assume that $\mathcal{G}$ and $\mathcal{G}'$ are two neighbouring graphs and $\mathcal{G}'$ has one extra node v^*.

$$\begin{aligned} \|q_{\varepsilon_2}(\mathcal{G}, \theta) - q_{\varepsilon_2}(\mathcal{G}', \theta)\| &\leq -|\mathcal{G}_{\theta_{\max}}.E| + |\mathcal{G}_\theta.E| - \sqrt{2}\frac{\theta}{\varepsilon_1} \\ &+ |\mathcal{G}'_{\theta_{\max}}.E| - |\mathcal{G}'_\theta.E| + \sqrt{2}\frac{\theta}{\varepsilon_1} \\ &\leq \left(|\mathcal{G}'_{\theta_{\max}}.E| - |\mathcal{G}_{\theta_{\max}}.E|\right) - \left(|\mathcal{G}'_\theta.E| - |\mathcal{G}_\theta.E|\right) \\ &\leq \theta_{\max} - \left(|\mathcal{G}'_\theta.E| - |\mathcal{G}_\theta.E|\right) \leq \theta_{\max} \end{aligned}$$

The second last inequality is due to Lemma 2 that states that $|\mathcal{G}'_{\theta_{max}}.E| - |\mathcal{G}_{\theta_{max}}.E| \leq \theta_{max}$. The last inequality is because $|\mathcal{G}'_\theta.E| \geq |\mathcal{G}_\theta.E|$. Note that the neighboring graph $\mathcal{G}'$ contains all edges of $\mathcal{G}$ plus extra edges of the added node v^*. Due to the stable ordering of edges in the edge addition algorithm, each extra edge of v^* either substitutes an existing edge or is added as an extra edge in $\mathcal{G}_\theta$. Therefore, the total edges $|\mathcal{G}'_\theta.E|$ is equal or larger than $|\mathcal{G}_\theta.E|$. We elaborate this detail further in the proof for Lemma 2. For the $\mathcal{I}_{\mathsf{P}}$ measure, the term in the last inequality changes to $|\mathcal{G}'_\theta.V_{>0}| - |\mathcal{G}_\theta.V_{>0}|$ and is also non-negative because $\mathcal{G}'$ contains an extra node that can only add and not subtract from the total number of nodes with positive degree. □

A.2.3 Proof for Lemma 2. This lemma states that the sensitivity of $f \circ \pi_\theta(\cdot)$ in Algorithm 2 is θ, where π_θ is the edge addition algorithm with the user input θ, and $f(\cdot)$ return returns edge count for $\mathcal{I}_{\mathsf{MI}}$ and the number of nodes with positive degrees for $\mathcal{I}_{\mathsf{P}}$.

Proof. Let's assume without loss of generality that $\mathcal{G}' = (V', E')$ has an additional node v^+compared to $\mathcal{G} = (V, E)$, i.e., $V' = V \cup \{v^+\}$, $E' = E \cup E^+$, and $E^+$is the set of all edges incident to v^+in $\mathcal{G}'$. We prove this lemma separately for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$.

For $\mathcal{I}_{\mathsf{MI}}$. Let Λ' be the stable orderings for constructing $\pi_\theta(\mathcal{G}')$, and t be the number of edges added to $\pi_\theta(\mathcal{G}')$ that are incident to v^+. Clearly, $t \leq \theta$ because of the θ-bounded algorithm. Let $e'_{\ell_1}, \ldots, e'_{\ell_t}$ denote these t edges in their order in Λ'. Let Λ_0 be the sequence obtained by removing from Λ' all edges incident to v^+, and Λ_k, for $1 \leq k \leq t$, be the sequence obtained by removing from Λ' all edges that both are incident to $v^+$and come after e'_{ℓ_k} in Λ'. Let $\pi_\theta^{\Lambda_k}(\mathcal{G}')$, for $0 \leq k \leq t$, be the graph reconstructed by trying to add edges in Λ_k one by one on nodes in $\mathcal{G}'$, and λ_k be the sequence of edges from Λ_k that are actually added in the process. Thus λ_k uniquely determines $\pi_\theta^{\Lambda_k}(\mathcal{G}')$; we abuse the notation and use λ_k to also denote $\pi_\theta^{\Lambda_k}(\mathcal{G}')$. We have $\lambda_0 = \pi_\theta(\mathcal{G})$, and $\lambda_t = \pi_\theta(\mathcal{G}')$.

In the rest of the proof, we show that $\forall k$ such that $1 \leq k \leq t$, at most 1 edge will differ between λ_k and λ_{k-1}. This will prove the lemma because there are at most t (upper bounded by θ) edges that are different between λ_t and λ_0.

To prove that any two consecutive sequences differ by at most 1 edge, let's first consider how the sequence λ_k differs from λ_{k-1}. Recall that by construction, Λ_k contains one extra edge in addition to Λ_{k-1} and that this edge is also incident to v^*. Let that additional differing edge be $e'_{\ell_k} = (u_j, v^+)$. In the process of creating the graph $\pi_\theta^{\Lambda_k}(\mathcal{G}')$, each edge will need a decision of either getting added or not. The decisions for all edges coming before e'_{ℓ_k} in Λ' must be the same in both λ_k and λ_{k-1}. Similarly, after e'_{ℓ_k}, the edges in Λ_k and Λ_{k-1} are exactly the same. However, the decisions for including

the edges after e'_{ℓ_k} may or may not be the same. Assuming that there are a total of $s \geq 1$ different decisions, we will observe how the additional edge e'_{ℓ_k} makes a difference in decisions.

When $s = 1$, the only different decision must be regarding differing edge $e'_{\ell_k} = (u_j, v^+)$ and that must be including that edge in the total number of edges for λ_k. Also note that due to this addition, the degree of u_j gets added by 1 which did not happen for λ_{k-1}. When $s > 1$, the second different decision must be regarding an edge incident to u_j and that is because degree of u_j has reached θ, and the last one of these, denoted by $(u_j, u_{i\theta})$ which was added in λ_{k-1}, cannot be added in λ_k. In this scenario, u_j has the same degree (i.e., θ) in both λ_k and λ_{k-1}. Now if s is exactly equal to 2, then the second different decision must be not adding the edge $(u_j, u_{i\theta})$ to λ_k. Again, note here that as $(u_j, u_{i\theta})$ was not added in λ_k but was added in λ_{k-1}, there is still space for one another edge of $u_{i\theta}$. If $s > 2$, then the third difference must be the addition of an edge incident to $u_{i\theta}$ in λ_k. This process goes on for each different decision in λ_k and λ_{k-1}. Since the total number of different decisions s is finite, this sequence of reasoning will stop with a difference of at most 1 in the total number of the edges between λ_{k-1} and λ_k.

For $\mathcal{I}_{\mathrm{P}}$. Assume, in the worst case, the graph $\mathcal{G}$ is a star graph with n nodes such that there exists an internal node that is connected to all other $n-1$ nodes. In this scenario, there are no nodes that have 0 degrees, and the $\mathcal{I}_{\mathrm{P}}$ measure $= n - 0 = 0$. If the neighbouring graph $\mathcal{G}'$ differs on the internal node, all edges of the graph are removed are the $\mathcal{I}_{\mathrm{P}} = n$. The edge addition algorithm π_θ would play a minimal role here as θ could be equal to n. □

A.2.4 Proof for Theorem 4. Algorithm 2 with the optimized EM in Algorithm 4 satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.

Proof. The total privacy budget is split in two ways for optimization and measure computation. These budgets can be composed using Proposition 1. □

A.2.5 Proof for Lemma 4. This lemma states that the sensitivity of $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ in the 2-step EM (Algorithm 4) defined in Equation (2) is $\theta_{\max} = \min(\tilde{d}_{\text{bound}}, |V|)$ for 1st EM step and $\theta_{\max} = \theta^*$ for the 2nd EM step.

Proof. Using Lemma 1, we know that the sensitivity of quality function $q_{\varepsilon_2}(\mathcal{G}, \theta_i)$ is given by the max over all candidates in $\max(\Theta)$. For the first step of the 2-step EM, the maximum candidate apart from $|V|$ is given by $\min(\tilde{d}_{\text{bound}}, |V|)$. For the candidate $|V|$, the quality function q differs. It only depends on the Laplace error $\frac{\sqrt{2}|V|}{\varepsilon_2}$ and has no error from e_{bias} as no edges are truncated due to $|V|$. For the 2nd step of EM, we truncate all values in the set greater than the output of the first step, i.e., θ^*. Therefore, the sensitivity becomes θ^*. □

A.2.6 Utility Analysis for Algorithm 2. Theorem 2 states that on any database instance D and its respective conflict graph $\mathcal{G}_\Sigma^D$, let o be the output of Algorithm 2 with Algorithm 3 over D. Then, with a probability of at least $1 - \beta$, we have

$$|o - a| \leq -\tilde{q}_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}| \cdot \beta}) \tag{13}$$

where a is the true inconsistency measure over D and $\beta \leq \frac{1}{e^{\sqrt{2}}}$.

Proof. By the utility property of the exponential mechanism [47], with at most probability $\beta/2$, Algorithm 3 will sample a bad θ^* with a quality value as below

$$q_{\varepsilon_2}(\mathcal{G}_\Sigma^D, \theta^*) \leq q_{\text{opt}}(D, \varepsilon_2) - \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) \tag{14}$$

which is equivalent to

$$e_{\text{bias}}(\mathcal{G}, \theta^*) \geq -q_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) - \frac{\sqrt{2}\theta^*}{\varepsilon_2}. \tag{15}$$

With probability $\beta/2$, where $\beta \leq \frac{1}{e^{\sqrt{2}}}$, we have

$$\text{Lap}(\frac{\theta^*}{\varepsilon_2}) \geq \frac{\ln(1/\beta)\theta^*}{\varepsilon_2} \geq \frac{\sqrt{2}\theta^*}{\varepsilon_2} \tag{16}$$

Then, by union bound, with at most probability β, we have

$$\begin{aligned}
& |o - a| \\
= \;& |f(\mathcal{G}_{\theta^*}) + \text{Lap}(\frac{\theta^*}{\varepsilon_2}) - a| \\
\geq \;& a - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \;& f(\mathcal{G}) - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \;& f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + f(\mathcal{G}_{\theta_{\max}}) - f(\mathcal{G}_{\theta^*}) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
= \;& f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + e_{\text{bias}}(\mathcal{G}, \theta^*) + \frac{\sqrt{2}\theta^*}{\varepsilon_2} \\
\geq \;& -q_{\text{opt}}(D, \varepsilon_2) + f(\mathcal{G}) - f(\mathcal{G}_{\theta_{\max}}) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta}) \\
= \;& -\tilde{q}_{\text{opt}}(D, \varepsilon_2) + \frac{2\theta_{\max}}{\varepsilon_1}(\ln \frac{2|\Theta|}{|\Theta_{\text{opt}}|\beta})
\end{aligned} \tag{17}$$

□

A.3 DP Analysis for $\mathcal{I}_{\mathrm{R}}$ (Algorithm 5)

A.3.1 Proof for Theorem 5. Algorithm 5 satisfies ε-node DP and always outputs the size of a 2-approximate vertex cover of graph $\mathcal{G}$.

Proof. The privacy analysis of Algorithm 5 is straightforward as we calculate the private vertex cover using the Laplace mechanism with sensitivity 2 according to Proposition 3. The utility analysis can be derived from the original 2-approximation algorithm. The stable ordering Λ in Algorithm 5 can be perceived as one particular random order of the edges and hence has the same utility as the original 2-approximation algorithm. □

A.3.2 Proof for Proposition 3. Algorithm 5 obtains a vertex cover, and its size has a sensitivity of 2.

Proof. Let's assume without loss of generality that $\mathcal{G}' = (V', E')$ has an additional node v^+compared to $\mathcal{G} = (V, E)$, i.e., $V' = V \cup \{v^+\}$, $E' = E \cup E^+$, and E^+ is the set of all edges incident to v^+ in $\mathcal{G}'$. We prove the theorem using a mathematical induction on i that iterates over all edges of the global stable ordering Λ.

Base: At step 0, the value of c and c' are both 0.

Hypothesis: As the algorithm progresses at each step i when the edge e_i is chosen, either the edges of graph $\mathcal{G}'$ which is denoted by E'_i has an extra vertex or the edge of graph $\mathcal{G}$ has an extra vertex. Thus, we can have two cases depending on some node v^* and its edges $\{v^*\}$. Note that at the beginning of the algorithm, v^* is the differing node v^+ and $\mathcal{G}'$ has the extra edges of v^*/v^+, but v^* may change as the algorithm progresses. The cases are as illustrated below:

- Case 1: E_i does not contain any edges incident to v^*, $E'_i = E_i + \{v^*\}$ and the vertex cover sizes at step i could be $c_i = c'_i$ or $c_i = c'_i + 2$.
- Case 2: E'_i does not contain any edges incident to v^*, $E_i = E'_i + \{v^*\}$ and the vertex cover sizes at step i could be $c_i = c'_i$ or $c'_i = c_i + 2$.
- Case 3: $E_i = E'_i$ and the vertex cover sizes at step i is $c_i = c'_i$. This case occurs only when the additional node v^+ has no edges.

Induction: At step $i + 1$, lets assume an edge $e_{i+1} = \{u, v\}$ is chosen. Depending on the i^{th} step, we can have 2 cases as stated in the hypothesis.

- Case 1 (When E'_i has the extra edges of v^*): We can have the following subcases at step $i + 1$ depending on e_{i+1}.
  a) If the edge is part of E'_i but not of E_i ($e_{i+1} \in E'_i \setminus E_i$): Then $e_{i+1} = \{u, v\}$ should not exist in E_i (according to the hypothesis at the i step) and one of u or v must be v^*. Let's assume without loss of generality that v is v^*. The algorithm will add (u, v) to C' and update $c'_{i+1} = c_i + 2$. Hence, we have either $c'_{i+1} = c_{i+1}$ or $c'_{i+1} = c + 2$.
  In addition, all edges of u and v/v^* will be removed from E'_{i+1}. Thus, we have $E_{i+1} = E'_{i+1} + \{u\}$, where $\{u\}$ represent edges of u. Now u becomes the new v^* and moves to Case 2 for the $i + 1$ step.
  b) If the edge is part of both E'_i and E_i($e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): In this case (u, v) will be added to both C and C' and the vertex sizes with be updated as $c_{i+1} = c_i + 2$ and $c'_{i+1} = c' + 2$.
  Also, the edges adjacent to u and v will be removed from E_i and E'_i. We still have $E'_i = E + v^*$ (the extra edges of v^* and remain in case 1 for step i+1.
  c) If the edge is part of neither E'_i nor E_i (If $e_{i+1} \in E'_i$ and $e_{i+1} \in E_i$): the algorithm makes no change. The previous state keep constant: $E'_{i+1} = E'_i, E_{i+1} = E_i$ and $c'_{i+1} = c'_i, c_{i+1} = c_i$. The extra edges of v^* are still in E'_{i+1}.
- Case 2 (When E_i has the extra edges of v^*) : This case is symmetrical to Case 1. There will be three subcases similar to Case 1 – a) in which after the update the state of the algorithm switches to Case 1, b) in which the state remains in Case 2, and c) where no update takes place.
- Case 3 (When $E_i = E'_i$): In this case, the algorithm progresses similarly for both the graphs, and remains in case 3 with equal vertex covers, $c_{i+1} = c'_{i+1}$.

Our induction proves that our hypothesis is true and the algorithm starts with Case 1 and either remains in the same case or oscillates between Case 1 and Case 2. Hence as per our hypothesis statement, the difference between the vertex cover sizes are upper bounded by 2. □