Managing Consent for Data Access in Shared Databases (Full Version)

Osnat Drien
Bar-Ilan University

Antoine Amarilli
LTCI, Télécom Paris, Institut Polytechnique de Paris

Yael Amsterdamer
Bar-Ilan University

***Abstract*—Data sharing is commonplace on the cloud, in social networks and other platforms. When a peer shares data and the platform owners (or other peers) wish to use it, they need the consent of the data contributor (as per regulations such as GDPR). The standard solution is to require this consent in advance, when the data is provided to the system. However, platforms cannot always know ahead of time how they will use the data, so they ofter require coarse-grained and excessively broad consent. The problem is exacerbated because the data is transformed and queried internally in the platform, which makes it harder to identify whose consent is needed to use or share the query results.**

Motivated by this, we propose a novel framework for actively procuring consent in shared databases, focusing on the relational model and SPJU queries. The solution includes a consent model that is reminiscent of existing Access Control models, with the important distinction that the basic building blocks – consent for individual input tuples – are unknown. This yields the following problem: how to *probe* peers to ask for their consent regarding input tuples, in a way that determines whether there is sufficient consent to share the query output, while making as few probes as possible in expectation. We formalize the problem and analyze it for different query classes, both theoretically and experimentally. Our analysis shows that the problem is NP-hard in general even for simple SELECT-PROJECT-JOIN queries, yet we develop a suite of optimal and approximate algorithms, for different query classes of interest. The algorithms connect the two seemingly unrelated research areas of Data Provenance and Interactive Boolean Evaluation. We provide theoretical guarantees as well as empirical evidence for the efficiency of our solutions.

I. Introduction

Data is routinely being shared online by peers on different platforms including social networks, cloud-based file sharing, and messaging applications. Using this data requires the consent of the data owner, according to regulations such as GDPR [1]. Typically, consent is obtained in advance, e.g., when the peer joins the platform (accepting the general platform policy) or when the data is originally uploaded to the platform. This has two main limitations:

1) The future ways in which data is used may be unknown and hard to anticipate. As a consequence, a-priori consent is coarse-grained. For example, a consent form could include a question of the sort: "do you agree that your data will be shared with third parties?". This is suboptimal: a peer might be fine with her data being shared with an NGO, but reluctant to having it shared with the press. At the time of data sharing, we can't get to such granularity, because we may not know yet with which third parties it would make sense to share the data.
2) To share original data as-is, one may simply ask the data owner for their consent when the need arises. However, data is often transferred, processed and combined through complex queries. Now, if one wishes to use a query result, whose consent is needed? On one hand, the query result will often not directly represent the data of any individual user; but on the other hand, the owners of the data from which the result has been derived do have rights with respect to it [1].

Similar questions are studied in the fields of data sharing, privacy and access control (see Section VI), but to our knowledge there has been no study of the combination of these two challenges, namely the need to actively obtain consent and to do so for derived data.

We therefore propose a *novel framework for managing consent in shared databases*, focusing on the relational setting and Select-Project-Join-Union (SPJU) queries. The main components of this framework are as follows.

Consent Model for Derived Data: As mentioned above, it is unclear who is the "owner" of pieces of data that are derived through data transformations. If, for instance, we join two database relations and now wish to share a tuple in the join result, do we need the consent of owners of the two joined tuples, or could one of them suffice? How about if we union two relations (under set semantics) and a tuple appears in both?

In the context of relational databases, we propose a model based on a "possible-worlds-like" semantics: for a database D, a query Q and a query result $t \in Q(D)$, we say that consent is given with respect to t if and only if t appears in the result of evaluating Q over $D' \subseteq D$ containing the input tuples for which consent is given. This in particular means that for a joined tuple we need the consent for both input tuples, and for union only one suffices.

So far, this choice is in line with standard access control models (e.g., [2], [3], [4], [5]). However, in our setting, it is also unknown if we have consent with respect to the *input tuples*, is also unknown – see point (1) above. Thus, we cannot simply propagate known consent from the input to the output, but we need to actively *probe* peers to ask for their consent. We use the notion of probing in an abstract sense, and in practice it may correspond to questions posed to human users [6], requests through automated agents, etc.

But who should we probe? Of course, if we probe all peers with respect to all data in the shared database, we would know precisely the consent status with respect to the entire input

Class	Provenance Shape	Consent for Full Query Result	Consent for Single Tuple
S/SP/SU	Disjunctions, overall read-once	PTIME Exact solution (Section IV-B)	PTIME Exact solution (Section IV-B)
SPU	Disjunctions, per-tuple read-once	NP-hard, approximate solution (Section IV-C)	PTIME Exact solution (Section IV-B)
SJ	Conjunctions, per-tuple read-once	NP-hard, approximate solution (Section IV-C)	PTIME Exact solution (Section IV-B)
SJU	k-DNFs (Section IV-A), per tuple read-once if partitioned	NP-hard, approximate solution (Section IV-C)	Approximate solution (Section IV-C), exact if partitioned (Section IV-B)
SPJ/SPJU	k-DNFs (Section IV-A)	NP-hard (Section IV-D) approximate solution for projection-limited queries (Section IV-C)	NP-hard (Section IV-D), approximate solution (Section IV-D)

TABLE I: Summary of the theoretical guarantees for different query classes and the consent decision problem. We use standard notations for the query operators: S – selection, P – Projection, J – Join and U – Union. See full analysis in Section IV.

database and thereby for any output as well. Probing, however, is a costly operation – in terms of latency and human effort. We thus wish to probe in a frugal manner, which leads to our problem (defined formally in Section II): *how to minimize the expected number of probes required to determine consent with respect to a query result?* We say "expected" (in the probabilistic sense), since the number of probes needed in total will typically depend on the answers that we receive, which of course are not known in advance.

Solution Framework: Towards a solution, we first observe that a key ingredient is *provenance*. The provenance of a query result tells us which input tuples it depends on, and in what way. In settings where we know whether consent is granted for each input tuple, the Boolean provenance of the query output (à la c-tables [7]) tells us whether we have consent to share the query results. Furthermore, when we do not know the consent for the input, we may annotate each input tuple with a variable, and provenance is then Boolean expression over these variables. The problem then reduces to determining the truth value for these Boolean expressions, by probing the truth values of the individual variables. We refer to this problem as *Interactive Boolean Evaluation*, which has been extensively studied in the literature in different contexts and under different names (e.g., [8], [9], [10], [11]). Our solution thus "marries" provenance and Boolean Evaluation, leading to different results for different query classes.

Complexity Analysis and Algorithms: We analyze the complexity of our problem, focusing on SPJU (Select-Project-Join-Union) queries, and considering both the sharing of a single tuple or of multiple tuples from the query result. Our main findings, also summarized in Table I, are as follows. We start with classes of queries (SPU, or so-called *partitioned* SJU) which we show to yield *read-once* provenance, where consent variables occur only once. By combining the provenance construction of [7] with an algorithm adapted from [9] we then achieve an exact PTIME solution. Unfortunately, we show that when variables repeat across tuples (*per-tuple* read-once), exact solution becomes NP-hard in data complexity even for SJ and SPU queries.

We next consider classes of queries that are either projection-free (SJUs) or projection-limited, i.e., the projection groups together a bounded number of tuples (this last characteristic depends on both the query and the database). For these classes we show an approximate algorithm using results from [10] and analyze its approximation ratio.

Finally, we consider the general class of unrestricted SPJU queries. Here, we can show that the problem is NP-hard in data complexity, even for sharing a single tuple and even for SPJs. In contrast, approximation is possible, even for general SPJUs, by adapting an algorithm of [8].

Experimental Results: To our knowledge, there have been no practical implementations, experimental studies or benchmark for Interactive Boolean Evaluation in a setting close to ours, let alone for our problem formulation, which is new. We have thus implemented all algorithms for the different variants and developed a benchmark that allows examining their performance over provenance expressions of various shapes. This includes two datasets: one specifically tailored to allow computation of the optimal probing strategies to be used as a yardstick (recall that this is in general NP-hard); and another that is designed to allow control over the provenance shape. As baselines algorithms, we have used two simple strategies: one that randomly chooses probes (among variables occurring in the provenance) and one greedily probing variables that occur most frequently in the provenance. Our algorithms performance was either optimal or very close to optimal in all cases where we could compare to the optimal strategy. For both datasets, our algorithms performance was always at least as good as that of the baselines algorithms, and in most cases it was significantly superior.

The rest of this paper is organized as follows. In Section II we introduce our consent model and formalize the problem of optimizing the number of probes. We start describing our solution to the problem in Section III by introducing a general framework combining provenance and Interactive Boolean Evaluation. In Section IV we analyze the problem complexity and propose algorithms for different classes of queries. Our experimental results are detailed in Section V. Section VI surveys related work and Section VII discusses alternative design choices and future work. We conclude in Section VIII.

Preliminary results for this work were demonstrated in a prototype user system for calendar data sharing [6], and discussed in a vision paper in [12].

Companies

cid	name
11	PennSolarExperts Ltd.

Vacancies

vid	cid	position	amount
111	11	analyst	3
112	11	supervisor	1

JobSeekers

sid	name	education	agency
1	David	Env. studies	Bob
2	Ellen	Env. studies	Bob
3	Frank	Env. studies	Alice
4	Georgia	Env. studies	Bob

Assignment

sid	vid	status	agency
1	111	hired	Bob
2	112	rejected	Alice
2	111	hired	Bob
3	111	rejected	Alice
4	112	hired	Alice

TABLE II: Example database of Alice's recruitment agency.

II. Model

We assume familiarity with standard relational database terminology [13]. Let $\mathcal{C}$ be a set of Boolean variables called *consent variables*. A *shared database* is a relational database where each tuple is annotated by a (unique) variable from $\mathcal{C}$, intuitively controlling its consent status.

Definition II.1 (Shared Database)**.** *A* shared database $\bar{D} = (D, L)$ *consists of a relational database* D *and of a one-to-one function* $L : \text{tuples}(D) \mapsto \mathcal{C}$ *mapping each tuple* t *in* D *to a consent variable* $\text{L}(t)$ *of* $\mathcal{C}$ *called the* annotation *of* t*.*

Example II.2. *Table II outlines a simple shared DB for recruitment agencies, including details of job seekers with their name, education, and agency to which they have applied; companies and job vacancies in these companies, including the type of position and the number of open positions; and the assignment of seekers to vacancies, including status and the responsible agency. Each tuple is associated with a consent variable (not shown in the table).*

The premise is that there is a hidden truth value to whether or not we are allowed to share each tuple (in a specific context, e.g., with a given third party).

Definition II.3. *A* consent valuation *is a function* $\text{val} : \mathcal{C} \mapsto \{\mathit{True}, \mathit{False}\}$.

Example II.4. *Recall the database in Table II, and assume that Alice wishes to share the* **JobSeekers** *table with Carol. As the owner of tuple 3, she may give her consent for this sharing, i.e., the consent valuation* val *maps the annotation of this tuple to true. In contrast, she may not know (at this stage) whether Bob agrees to share his tuples with Carol.*

Next, instead of sharing the data as-is, we allow a *query* executed on the database to perform some analysis and consider the sharing of its results. Most aspects of the model may apply to any query language, but we will focus in particular on SPJU (Select-Project-Join-Union) queries [13] and discuss extensions in Section VII.

Example II.5. *Recall our running example and now assume that Alice wishes to share with Carol the names of companies where environmental studies graduates have successfully found jobs. To this end, she runs the query* Q_{ex} *in Figure 1 on the database in Table II. In our simple example the answer is the single company in the database – "PennSolarExperts Ltd.", where David, Ellen and Georgia have been hired.*

```
SELECT DISTINCT c.name
FROM Companies c, JobSeekers j, Vacancies v,
     Assignments a
WHERE c.cid = v.cid AND v.vid = a.vid AND
      a.status = 'hired' AND a.sid = s.sid AND
      s.education = 'Env. studies'
```

Fig. 1: Query Q_{ex} over the example database

The question is then: given a shared database and an SPJU query, are we allowed to share the result? In the spirit of previous work on incomplete databases [7], we define that a query answer is shareable iff it appears in the result of evaluating the query over the sub-database consisting only of the shareable input tuples.

Definition II.6. *Given a shared database* $\bar{D} = (D, L)$*, an SPJU query* Q*, a tuple* $t \in Q(D)$*, and a consent valuation* val*,* t *is* shareable *iff* $t \in Q(D')$*, where* $D' \subseteq D$ *is the database consisting of every* $t' \in D$ *such that* $\text{val}(L(t')) = \text{True}$*.*

Example II.7. *Assume that the consent valuation maps tuple 3 of* **JobSeekers** *to True, maps all other tuples of this table to False (Bob did not allow us to share them), and maps all tuples of the other tables to True. The shareable fragment of* Q_{ex} *for this database is its single result "PennSolarExperts Ltd.", because this result would be returned even if the* **JobSeekers** *table consisted only of tuple 3.*

The challenge, of course, is that we may not know some or all of the consent values, that are initially known only to data owners. To find out which query results can be shared, we can pose questions or *probes* to the owners of relevant tuples, asking whether the tuple can be shared. Our goal is to optimize the number of questions posed in order to determine which query results can be shared.

To formally define this problem, we must talk about the performance of an algorithm to choose probes, which is challenging because this performance will usually depend on the outcome. To measure the performance, we use a simple probabilistic model: we assume that each consent variable $x \in \mathcal{C}$ is given a known probability $\pi(x)$ of being set to True in the consent valuation, independently from all other variables. We discuss below (Section VI, "Predicting probe answers and probabilities") existing means that could potentially be employed for estimating these probabilities based on e.g., peer properties. The performance of a probing algorithm is then the expected number of probes that it makes, under the probability distribution on consent valuations given by the probabilities.

Problem Definition: The problem of OPT-PEER-PROBE (for "optimizing peer probing") is defined as follows, where we use italics to highlight some problem design choices that we will revisit in Section VII:

Definition II.8 (OPT-PEER-PROBE)**.** *We are given a shared database $\bar{D}$, a probability distribution over consent variables in $\bar{D}$, and an SPJU query Q. A consent valuation* val *is drawn at random according to the distribution, but is not given to the algorithm. The problem* OPT-PEER-PROBE *is to define an algorithm that can decide for* every *tuple $t \in Q(D)$ whether it is shareable or not under* val*. To reveal that, at each step, the algorithm may probe a consent variable x of its choice and obtain* $\mathrm{val}(x)$ *as a result. We assume probes are* sequential*, i.e., posed one at a time so that their choice may depend on previous answers. For a (hidden) choice of* val*, the algorithm's total cost is the* total number of probes*. The optimization target is to minimize the algorithm's* expected cost over the random choice of val*.*

We also consider the variant OPT-PEER-PROBE-SINGLE*, where we are additionally given as input one tuple t in the query result $Q(D)$, and we only need to determine whether this specific tuple is shareable.*

In the next section, we propose a general solution framework for OPT-PEER-PROBE (-SINGLE). We then "instantiate" it, in subsequent sections, with exact and approximate algorithms for different query classes.

III. Solution Overview

Our framework leverages techniques from two existing areas, namely relational database provenance and Interactive Boolean Evaluation, which we now present.

A. Provenance

We recast the notion of Boolean provenance tracking (or equivalently, c-tables [7]) to our setting. We start by extending the notion of a shared database from Definition II.1 to derived relations, by supporting annotations that are Boolean combinations of consent variables from $\mathcal{C}$.

Definition III.1. *For any set of variables X, let us denote by* $\mathrm{PosBool}(X)$ *the set of all positive Boolean expressions over X (i.e. constructed via conjunctions and disjunctions over elements of X). An* annotated relation *$\bar{R} = (R, L)$ is a relation R with an annotation function $L : \mathrm{tuples}(R) \mapsto \mathrm{PosBool}(X)$.*

Now, let $\bar{D} = (D, L)$ be a shared database as in Definition II.1, and let Q be an SPJU query. We define the *annotated result* of evaluating Q over $\bar{D}$, denoted $Q(\bar{D})$, as an annotated database $Q(\bar{D}) = (Q(D), L')$ where:

- $Q(D)$ is the result of "standard" query evaluation of Q with respect to D;
- $L' : \mathrm{tuples}(Q(D)) \mapsto \mathrm{PosBool}(\mathcal{C})$ is an annotation function over the tuples of $Q(D)$ that we define below.

The annotations in $Q(\bar{D})$ are inductively defined based on the structure of Q:

- For a *selection* over a relation R, the annotation of each output tuple t is the same as the annotation of the tuple t' of R that is identical to t.
- For the *union* of two relations R and S in $\bar{D}$, the annotation of each output tuple t is the disjunction of the annotation of the tuple t' in R and of that of the tuple t'' in S that are identical to t (or just the annotation of one of them, if the other does not exist).
- For a *projection* applied to a relation R in $\bar{D}$, the annotation of each output tuple t is the disjunction of the annotation of all tuples t' in R that project to t, i.e., have the same values as t on the projected attributes.
- For the *Cartesian product* of relations R, S in $\bar{D}$, the annotation of each output tuple t is the conjunction of the annotations of the tuples $t' \in R$ and $t'' \in S$ such that the concatenation of t' and t'' yields t.

Observe that for each output tuple t, and for any assignment of truth values to the consent variables, the truth value of the Boolean provenance formula labeling t tells us whether the tuple may be shared. In other words:

Proposition III.2 (Adapted from [7], [14])**.** *For every shared database $\bar{D} = (D, L)$, SPJU query Q, and consent valuation* val*, the shareable fragment of $Q(D)$ consists of the tuples in $Q(\bar{D})$ whose annotations evaluate to True under* val*.*

We also note that the provenance construction is asymptotically as efficient as standard query evaluation:

Proposition III.3 (Adapted from [7], [14])**.** *For every shared database $\bar{D} = (D, L)$ and SPJU query Q, the annotated result $Q(\bar{D})$ may be computed in time* $\mathrm{O}(|D|^{|Q|})$*.*

Combined, the above results mean that we may compute, in time $O(|D|^{|Q|})$, a monotone Boolean expression for each tuple, whose truth value under any consent valuation correctly reflects if the output tuple can be shared or not. Thus, our problem can be rephrased as that of probing variables to infer the truth value of this Boolean expression. This is the topic of investigation in an area called *Interactive Boolean Evaluation*, which we now present.

B. Interactive Boolean Evaluation

The field of Interactive Boolean Evaluation focuses on the following problem: given a Boolean formula, determine a probing strategy on its variables to identify its truth value, in a way that minimizes the number of probes. Of course, the probing strategy may depend on the result of previous probes; for instance, in a simple disjunction, the evaluation process can halt as soon as we observe a variable whose value is True.

The probing strategy of a given formula is formally captured by the notion of a *Binary Decision Diagram* (BDD), which we extend here to account for *simultaneous evaluation of multiple formulas*: intuitively, given a set of formulas $\Phi = \varphi_1 \dots \varphi_m$, a BDD is a DAG representing the sequence of probes produced by a deterministic algorithm, such that each inner node represents a probed variable x and its outgoing edges represent the sequences of probes in case $x =$ False and $x =$ True respectively. Each leaf of the DAG represents an outcome that describes the evaluation result (True/False) of every formula $\varphi_i \in \Phi$. Note that BDDs are only a way to formalize evaluation strategies and describe their performance, but the BDD is usually not materialized (its size may be

exponential in Φ) and is only represented implicitly, e.g., as the possible execution traces of a given algorithm.

We measure the efficiency of a BDD is via the notion of *expected cost*:

Definition III.4 (BDD Expected Cost)**.** *Let D be a BDD on a set of variables X for which we have a probability distribution π such that each $x \in X$ has probability $\pi(x)$ of being True, independently from all other variables. The* expected cost *of D under π is the expected total number of variables tested on the path in D from a root node to a leaf node.*

Note that, if we denote by $n := |\mathrm{vars}(\Phi)|$ the number of distinct variables appearing in a given set of formulas Φ, then the expected cost of a BDD of Φ can clearly be bounded by n, and in some cases all BDDs have cost equal to n (e.g., for XOR formulas). However, as stated by the following theorem, some formulas, even monotone ones, have BDDs that are exponentially more efficient.

Theorem III.5. *For arbitrarily large integers n, there is a monotone Boolean formula with n variables such that, for any probability distribution π mapping all variables to values different from 0 and 1, there is a BDD of $\mathrm{O}(\log n)$ expected depth as well as a BDD of expected depth $\Omega(n)$. Moreover, the $\mathrm{O}(\log n)$ BDD is optimal for a constant distribution $\pi : X \to p$ giving the same probability $p \in (0,1)$ to every variable.*

Proof sketch. Define the monotone DNF formula $\psi_0 := (w \wedge x) \vee (x \wedge y) \vee (y \wedge z)$ and recursively define $\psi_{i+1} := (u_i \wedge \psi_i) \vee (u_i \wedge v_i) \vee (v_i \wedge \psi_i')$ where u_i, v_i are fresh variables and ψ_i' is obtained by consistently replacing the variables of ψ_i by fresh variables. We can show that $|\mathrm{vars}(\psi_i)|$ is exponential in i, and that there exists a BDD ψ_i whose expected cost is $\mathrm{O}(i)$ (by probing first the u_i, v_i variables) and a BDD whose expected cost is $\Theta(n)$ even though it does not make useless probes (by probing first the copies of ψ_0). □

This result illustrates that we can achieve large performance savings in some cases if we intelligently choose which variables to probe, motivating the Interactive Boolean Evaluation problem of finding an optimal strategy. As we will show, the interplay of provenance generation and Interactive Boolean Evaluation algorithms thus leads to different performance results when making different assumptions on the database and query. This is what we study in subsequent sections.

IV. Complexity and Algorithms

A. General Characterization

In our framework, the key to understanding the complexity of our problem is to characterize which Boolean expressions may be obtained as provenance, depending on the query class. For SPJU queries, we propose such a characterization, based on the notion of *k-DNFs*.

Definition IV.1. *Let k be a constant. A Boolean formula is in k-DNF if (a) it is in a* Disjunctive Normal Form *(disjunction of conjunctions) and (b) the size of each term, i.e., the number of (distinct) variables in a conjunction, is bounded by k.*

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
$\pi : X \to [0,1]$ the probability of each variable to be True,
dnfs – m monotone DNF formulas over X.
$\mathrm{val} \leftarrow$ partial valuation setting all $x \in X$ to Unknown;
while $\exists \psi \in \mathrm{dnfs}, \quad \mathrm{val}(\psi) =$ *Unknown* **do**
  $T \leftarrow \{t \in \mathrm{terms}(\psi) \mid \psi \in \mathrm{dnfs} \wedge \mathrm{val}(t) =$ Unknown$\}$;
  $t \leftarrow \arg\max_{t' \in T} \frac{1}{|t'|} \prod_{x \in t' \wedge \mathrm{val}(x) = \text{Unknown}} \pi(x)$;
  $x \leftarrow \arg\min_{x \in t} \pi(x)$;
  $b \leftarrow$ probe x;
  $\mathrm{val} \leftarrow \mathrm{val}$ setting $x := b$;
return $[\mathrm{val}(\varphi_1), \ldots, \mathrm{val}(\varphi_m)]$ *for* $\mathrm{dnfs} = [\varphi_1, \ldots, \varphi_m]$.

Algorithm 1: Algorithm RO for OPT-PEER-PROBE

We show a two-way correspondence between the provenance of SPJUs and the class of Boolean k-DNF formulas.

Proposition IV.2. *1) For each SPJU query Q there exists a value k (intuitively, the maximal number of joins in Q) such that for every shared database $\bar{D}$, the provenance of each tuple in $Q(\bar{D})$ may be represented in a monotone k-DNF form. This form may further be constructed in PTIME in data complexity (i.e., as a function of $\bar{D}$).*

2) Conversely, for every monotone k-DNF formula φ, there exists an SPJ query Q depending only on k, a shared database $\bar{D}$ whose size is linear in φ and a probability distribution π such that the query output $Q(\bar{D})$ is a singleton tuple whose provenance is equivalent to φ.

We next consider different classes of queries, starting from a class for which we show an exact optimal solution, and ending with general SPJU queries in Section IV-D. But first, for convenience, we extend the notion of valuation to account for unknown consent values.

Definition IV.3. *A* partial consent valuation *is an assignment of truth values to $\mathcal{C}$, $\mathrm{val} : \mathcal{C} \mapsto \{$True, False, Unknown$\}$, where Unknown stands for unknown consent value. Any valuation can be extended to Boolean expressions over $\mathcal{C}$ via* Kleene three-valued logic*, e.g., True $\wedge$ Unknown $=$ Unknown, etc.*

B. Read-once

In Interactive Boolean Evaluation, the class of read-once DNF formulas, where each variable occurs only once, is known to have an exact (i.e., optimal) solution that minimizes the number of probes [9]. Algorithm RO (outlined in Algorithm 1) builds on this technique to solve OPT-PEER-PROBE, assuming the provenance of the query output is computed according to Section III. It orders the DNF terms of all the expressions (that are not evaluated yet) by their probability to be True (as a product of variable probabilities), then chooses the term with the highest fraction of probability over size. It then probes the variables in increasing order of probability for an affirmative answer, until the term is evaluated (and if it is True so are the expressions containing it).

In the multi-expression setting, we distinguish cases where each individual tuple has read-once DNF provenance, termed here *per-tuple RO*, from the more restrictive condition where we require the provenance of the entire query result to be read-once, termed *overall RO* (where each variable only occurs once and in only one tuple). Clearly, running an SPJU query on a database may *happen* to produce read-once provenance, in which case Algorithm RO is optimal for overall RO and OPT-PEER-PROBE, or per-tuple RO and OPT-PEER-PROBE-SINGLE. However, this does not identify the cases where RO is guaranteed to be optimal, i.e., the queries whose provenance is guaranteed to have this form on every database.

While previous work on provenance considered read-once provenance (e.g., [15], [16]), to our knowledge there has been no characterization of which queries always yield read-once *DNF*. Still, we can identify concrete practical subclasses of SPJU that have this guarantee.

Proposition IV.4. *Algorithm* RO *is a PTIME exact solution to* OPT-PEER-PROBE *for S/SP/SU queries (Selection/Selection-Projection/Selection-Union) over shared databases.*

By construction, the provenance of each output tuple for such queries is a disjunction of variables, disjoint between tuples – i.e., overall read-once.

We further identify classes of queries that guarantee per-tuple read-once provenance for every database.

Proposition IV.5. *Algorithm* RO *is a PTIME exact solution to* OPT-PEER-PROBE-SINGLE *for SPU and SJ queries over shared databases.*

The provenance of these queries is in the form of disjunctions and conjunctions respectively, and hence per-tuple read-once, but variables may repeat across expressions, e.g., when a tuple is matched to more than one other tuple in a join. When the query involves both join and union, another syntactic property is required to ensure read-once provenance.

Definition IV.6. *We say that a SPJU query is* partitioned *if for every relation of the shared database, for every relation, all of its occurrences appear in a single SPJ query (this query may feature self-joins).*

Example IV.7. *Queries without union, such as the query in Figure 1, are trivially partitioned. A* non-partitioned *query would be, e.g., taking the union of this query with another that uses one of its relations, e.g.,* SELECT DISTINCT c.name FROM Companies WHERE c.name LIKE 'Penn%'.

If the query is partitioned and has no projection, then we have an efficient and optimal algorithm for our problem.

Proposition IV.8. *Algorithm* RO *is a PTIME exact solution to* OPT-PEER-PROBE-SINGLE *for partitioned SJUs over shared databases.*

Unfortunately, this optimality result does not extend to OPT-PEER-PROBE. In fact, the following theorem proves that OPT-PEER-PROBE is intractable even in a case where we have per-tuple read-once provenance.

Theorem IV.9. *There is a fixed SJ query Q for which the* OPT-PEER-PROBE *problem is NP-hard, even when all variables have the same probability.*

The proof (omitted) is by reduction from VERTEX COVER. We can construct a query Q such that for every instance of VERTEX COVER we construct an input shareable database $\bar{D}$ such that each tuple of $Q(\bar{D})$ correspond to an edge of the graph. We then show that an exact solution to OPT-PEER-PROBE corresponds to a minimal vertex cover. The latter part of the proof is partly based on the proof from [8] for the hardness of Boolean evaluation for k-DNF formulas.

Similarly, we can prove the hardness of OPT-PEER-PROBE for SPU queries, whose provenance is in the form of disjunctions (since they are is join-free) and thus per-tuple read-once.

Theorem IV.10. *There is a fixed SPU query Q for which the* OPT-PEER-PROBE *problem is NP-hard, even when all variables have the same probability.*

C. Queries with Limited Projection

The next fragment that we consider is that of *projection-free queries*, i.e., SJUs, not necessarily partitioned. Importantly, for such queries, the provenance size for each individual tuple is in fact *independent of the database size* – the number of terms is bounded by the number of unions, and the size of terms is bounded by the number of joins in a single conjunctive query.

We will use this property to design an approximation algorithm, specifically the Q-value Algorithm, whose details appear in Algorithms 2 and 3, and which incorporates some techniques of [10]. The main idea is as follows: given both the CNF and DNF of some formula, termed *CDNF* formulas, compute exactly how many DNF terms (conjunctions) and CNF clauses (disjunctions) are eliminated (evaluated to True/False) if a given variable value is False or True, respectively. A greedy selection of concepts as a function of the expected number of clauses/terms they eliminate, termed *Q-value*, then yields an expected number of probes which approximates the optimal BDD. This is shown in [10] based on [17], and presented below.

We first overview Algorithm 2 for a greedy selection of the next probe, then explain how it is used in the full Algorithm 3. The outermost loop iterates over variables x_i whose value is Unknown – namely, variables that have not been probed yet. For each such variable we iterate over possible probe responses (b =True/False) and create a hypothetical updated assignment val' where $x_i = b$. We then iterate (innermost loop) over input formulas. For each formula we compute the number of terms/clauses that are *not* eliminated by setting $x := b$, which we denote respectively by t_j and c_j. The formula $\mathrm{terms}[j] \cdot \mathrm{clauses}[j] - t_j \cdot c_j$ computes the Q-value of a single CDNF, reaching its maximum value when all terms or clauses are eliminated, and strictly increasing with each elimination (Theorem 1 of [10]). Similarly, the sum of these

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
val a valuation for X,
$\pi : X \to [0,1]$ the probability of each variable to be True,
$[\varphi_1, \ldots, \varphi_m]$ – array of m monotone DNF formulas over X,
$[\mathrm{CNF}(\varphi_1), \ldots, \mathrm{CNF}(\varphi_m)]$ – (monotone) CNF of $\varphi_1, \ldots, \varphi_m$ respectively.
Output: $x \in X$ – the next variable to probe
$\mathrm{terms}, \mathrm{clauses} \leftarrow$ arrays containing at the jth index the number of terms and clauses in φ_j and $\mathrm{CNF}(\varphi_j)$ resp.;
$\mathrm{QVal} \leftarrow$ array of size $|X|$;
for $x_i \in X$ *s.t.* $\mathrm{val}(x) =$ *Unknown* **do**
  for $b \in \{$*False*, *True*$\}$ **do**
    $Q_b \leftarrow 0$;
    $\mathrm{val}' \leftarrow \mathrm{val}$ setting $x_i := b$;
    for $j \in 1 \ldots m$ **do**
      $t_j \leftarrow$ # terms in φ_j evaluated to Unknown by val';
      $c_j \leftarrow$ # clauses in $\mathrm{CNF}(\varphi_j)$ evaluated to Unknown by val';
      $Q_b \leftarrow Q_b + \mathrm{terms}[j] \cdot \mathrm{clauses}[j] - t_j \cdot c_j$;
    $\mathrm{QVal}[i] \leftarrow \Pr(x_i) \cdot Q_{\mathrm{True}} + (1 - \Pr(x_i)) \cdot Q_{\mathrm{False}}$;
return $\arg\max_i \mathrm{QVal}[i]$

Algorithm 2: Algorithm Q-value-next (pick next probe).

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables
$\pi : X \to [0,1]$ the probability of each variable to be True
dnfs – m monotone DNF formulas over X.
$\mathrm{val} \leftarrow$ partial valuation setting all $x \in X$ to Unknown;
$\mathrm{cnfs} \leftarrow$ monotone CNF for every $\varphi \in \mathrm{dnfs}$;
while $\exists \varphi \in \mathrm{dnfs}, \quad \mathrm{val}(\varphi) =$ *Unknown* **do**
  $x \leftarrow$ Q-value-next$(X, \pi, \mathrm{dnfs}, \mathrm{cnfs})$;
  $b \leftarrow$ probe x;
  $\mathrm{val} \leftarrow \mathrm{val}$ setting $x := b$;
return $[b_1, \ldots, b_m]$ *the consent values in* $\{$*True*, *False*$\}$ *for the formulas of* dnfs.

Algorithm 3: Algorithm Q-value for OPT-PEER-PROBE

individual CDNF Q-values is a Q-value for evaluating all of them, reaching its maximum when all CDNFs are evaluated (using a construction like that of Section 7 of [10]). To put everything together, Algorithm 3 first translates the input DNF formulas into CNF, and then repeatedly probes the variable selected by Algorithm 2 until all formulas are evaluated.

The following proposition summarizes the time complexity and approximation ratio of this algorithm. In particular, for OPT-PEER-PROBE-SINGLE we show that these depend only on the query size.

Proposition IV.11. *Let Q be a projection-free query with j joins and u unions over $\bar{D}$.*

1) *The* Q-value *Algorithm is a* $\mathrm{O}(\left|\bar{D}\right|^{|Q|})$ *time* $\log\left|Q(\bar{D})\right|$*-approximation for* OPT-PEER-PROBE.
2) *Given a tuple $t \in Q(D)$, the* Q-value *Algorithm is a* $\mathrm{O}(j^u \left|Q\right|^2)$ *time* $(u \log j)$*-approximation for* OPT-PEER-PROBE-SINGLE.

The proof follows from the algorithm structure and from the approximation ratios of the general Q-value approach [10], [17]. The time complexity for OPT-PEER-PROBE is dominated by the cost of evaluating Q over $\bar{D}$; note that for OPT-PEER-PROBE-SINGLE we can compute the provenance of the specific output tuple t we are interested in, without evaluating the whole query. As for the CNF representation, in the worst case, we can compute it for each tuple using a brute force algorithm that checks all the combinations of variables from different terms; this costs $\mathrm{O}(j^u)$.

Extending the fragment: We have devised an algorithm for projection-free queries but it is also applicable, with the same complexity guarantees, to a larger class of instances to the OPT-PEER-PROBE problem, where the number of tuples that agree on the projected attributes is bounded by a small constant p. We call this fragment "projection-p-limited". This fragment can occur quite naturally in practical settings:

Example IV.12. *Considering the Example DB in Table II, the number of possible values in some attributes is practically a small constant, e.g., the* agency *which refers to a collaborator of Alice, and the* status *of the job application. Assume that Alice has x collaborators (including herself) and that there are y possible statuses. In the query* SELECT DISTINCT sid, vid FROM Assignment*, at most xy tuples can agree on the columns that are projected out. Thus, each answer tuple can have at most xy terms in its DNF annotation. As long as xy is small it may still be feasible to compute the corresponding CNF of this DNF.*

Proposition IV.13. *Let Q be a projection-p-limited query with j joins and u unions over $\bar{D}$.*

1) *The complexity and approximation ratio for* Q-value *Algorithm and* OPT-PEER-PROBE *is the same for Q and $\bar{D}$ as in Prop. IV.11.*
2) *Given a tuple $t \in Q(D)$, the* Q-value *Algorithm is a* $\mathrm{O}(j^u p \left|Q\right|^2)$ *time* $(pu \log j)$*-approximation for* OPT-PEER-PROBE-SINGLE.

The proof is by arguments similar to Prop. IV.11, but noticing that the number of DNF terms is a product of the number of unions in the query and of p.

Finally, recall that for SPU queries (without joins), provenance is in the form of disjunctions of individual variables, so it is already provided both in DNF and in CNF form. Consequently, Q-value provides an approximate solution for this class of queries as well, even when projection is not limited (while by Theorem IV.10 exact solution is NP-hard).

Proposition IV.14. *Let Q be an SPU query over $\bar{D}$. The* Q-value *Algorithm is a PTIME* $(\log|D|)$*-approximation for* OPT-PEER-PROBE.

Input: $X = \{x_0, x_1 \ldots\}$ – a set of variables,
$\pi : X \to [0,1]$ the probability of each variable to be True,
dnfs – m monotone DNF formulas over X.
$\mathrm{val} \leftarrow$ partial valuation setting all $x \in X$ to Unknown;
$\mathrm{cost1}, \mathrm{cost2} \leftarrow 0$;
while $\exists \varphi \in \mathrm{dnfs}, \quad \mathrm{val}(\varphi) = \textit{Unknown}$ **do**
  if $\mathrm{cost1} \geq \mathrm{cost0}$ **then**
    $x \leftarrow$ choose probe using Alg0 from [8], Section 5.1 on $\bigvee_{\psi \in \mathrm{dnfs} \mid \mathrm{val}(\psi) = \mathrm{Unknown}} \psi$;
  else
    $x \leftarrow$ choose probe using RO on dnfs, val;
  $b \leftarrow$ probe x;
  $\mathrm{val} \leftarrow \mathrm{val}$ setting $x := b$;
return $[b_1, \ldots, b_m]$ *the consent values in* $\{\textit{True}, \textit{False}\}$ *for the formulas of* dnfs.

Algorithm 4: Algorithm General for OPT-PEER-PROBE

This approximation ratio comes from the fact that, when we allow projection, the provenance of a single output tuple may be a disjunction of all the input annotations.

D. General SPJU Queries

To conclude our analysis of the complexity of our problem, we consider the general problem setting, where the class of queries that is considered is SPJU, with no restrictions imposed. In Section IV-B we have already shown that OPT-PEER-PROBE is hard. For general SPJUs (and even just SPJs), this hardness also holds for OPT-PEER-PROBE-SINGLE:

Theorem IV.15. OPT-PEER-PROBE *and* OPT-PEER-PROBE-SINGLE *are NP-hard (in data complexity) for SPJ queries, even on shared databases where all tuples have the same probability.*

The proof follows the characterization of Prop. IV.2 and the hardness proof of Theorem IV.9, by projecting out all the variables (Boolean query) to obtain a single output tuple, whose provenance is "as hard" for OPT-PEER-PROBE-SINGLE as it was without projection for OPT-PEER-PROBE.

Due to the hardness of the problem, we propose Algorithm General (outlined in Algorithm 4) as an approximate solution for OPT-PEER-PROBE, by extending the algorithm of [8]. The algorithm of [8] alternates between two sub-algorithms, alg0 and alg1, which respectively try to show that the formula is False and True; we halt as soon as one of them succeeds. To allow simultaneous evaluation of multiple formulas, we apply alg0 on the disjunction of tuple provenances in DNF (such that it tries to prove every DNF term in the entire provenance is False, each time discarding evaluated terms and formulas). We then replace alg1 by the extension of RO to multiple formulas, as they operate on a similar principle (greedily selecting the term by which proving a formula is True is cheapest, and sequentially probing its variables).

Based on the result of [8], we obtain the following.

Theorem IV.16 ([8], Theorem 5.3)**.** *Algorithm* General *is a PTIME constant-factor (Data Complexity) approximation for* OPT-PEER-PROBE-SINGLE.

The problem of proving approximation guarantees for OPT-PEER-PROBE over general SPJUs remains open. However, in our experimental study in Section V we observe that General performs almost as well as Q-value, which does have approximation guarantees.

Beyond syntactically-defined fragments: We have provided syntactic characterizations of queries for which our algorithms have proven approximation/optimality guarantees. However the algorithms are in fact more general, and rely on the provenance structure. It may e.g. be the case that an SPJU query yields read-once provenance for some input database, or that provenance becomes read-once at some point of the evaluation process. Thus, we can perform runtime checks of the provenance structure and decide accordingly which of the algorithms to use.

V. Experimental Study

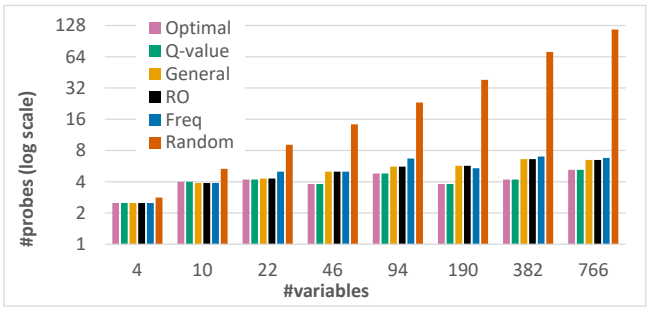
We have implemented all algorithms described in Section IV and examined their performance. We start by describing our experimental settings, and then present the results.

A. Experimental Settings

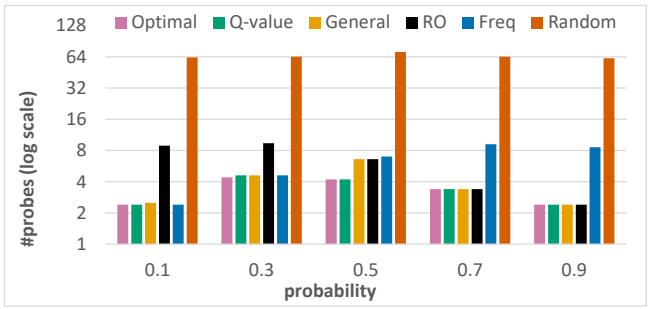
To our knowledge, the problem that we study has not been investigated before, so we are not aware of a standard benchmark or of existing algorithms to use as competitors. We have thus designed a dedicated benchmark for the problem.

Datasets: Our first dataset, called the **ψ-dataset**, is used to compare our algorithms to a known *optimal strategy*. As we explained in Section IV, computing optimal strategies is NP-hard in general, hence our choice to design a dataset where the optimal solution is known by construction. The **ψ-dataset** consists of single-tuple query outputs whose provenance is of the form ψ_i as defined in the proof sketch for Theorem III.5. The size of ψ_i increases exponentially with i while the optimal strategy makes $\mathrm{O}(i)$ probes. By default we use ψ_6 with 382 distinct variables. The total DNF/CNF provenance size for this experiment was up to 4.3K.

Our second dataset, **skewed**, is a parametrized dataset that we generate randomly based on parameters such as the number of tuples, the query projection limit (p for projection-p-limited queries, as defined in Section IV-C, affecting the number of terms), the number of joins, and the average number of repetitions per variables. To allow ensure that our algorithms can handle complex Boolean expression structure that also occurs in real queries, the variables in this dataset are split in four types: frequent/infrequent variables that co-occur with frequent/infrequent variables. E.g., in the provenance formula $(a \wedge b \wedge c) \vee (a \wedge e \wedge f) \vee (g \wedge h \wedge i) \vee (g \wedge h \wedge j)$, a is frequent and co-occurs with infrequent variables, while g and h are frequent and co-occur with each other. The default parameters we have used are 1000 query output rows, 4 joins, 8 as projection limit, and where each variable repeats 2.6 times on average. The average total DNF/CNF provenance sizes for the reported



(a) Number of probes for varying expressions size



(b) Number of probes for varying True probabilities

Fig. 2: Comparing to Optimal over the ψ-**dataset** (less probes is better).

experiments (summed over all rows, averaged over experiment repetitions) were up to 0.4M and 2.2M, respectively.

Algorithms: The algorithms that we have compared are:

- Random. A baseline probing variables in a random order.
- Freq. A baseline that greedily probes a variable with the maximal number of occurrences in the DNF provenance.
- RO. Algorithm 1 from Section IV-B
- Q-value. Algorithm 3 from Section IV-C.
- General. Algorithm 4 from Section IV-D.

Of course, all the algorithms that we benchmark will maximally simplify expressions after each probe answer, so that they never make useless probes. For fairness, all algorithms break ties by the same arbitrary criterion.

We have implemented all algorithms in node.js using Express, and in Java 13. Experiments were run on a Windows 10 machine using an Intel Core i7 5600U processor with 8 GB of DDR4 memory. Each experiment was executed at least 10 times (at least 50 times for Random) and the reported results are the average over these executions, each time drawing a valuation uniformly at random according to the variable probabilities, and executing all algorithms over this valuation. For the **skewed** dataset we have further re-generated the data for each execution. In every experiment we have used the same probability for all variables. Except when stated otherwise, we have used probability 0.5 for the ψ-**dataset** and 0.7 for **skewed** (leading to a roughly even number of shareable and unshareable tuples in most cases).

B. Experimental Results

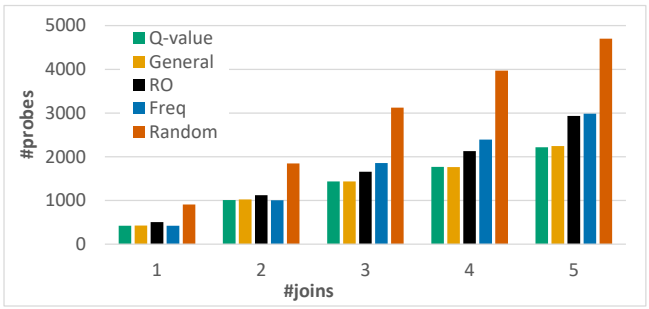
We next describe the results of our experiments. We evaluate our algorithms based on the *number of probes* that they issue, which is the criterion that we are trying to optimize; we discuss execution times at the end of the section. Note that all algorithms can be used even in cases where they do not have optimality guarantees – an algorithm with no optimality guarantees can still turn out to be efficient in practice.

We start by comparing algorithm performance to the optimal algorithm for the ψ-**dataset** (for which, as we explain above, the optimal solution is known by construction). Figure 2a shows the performance of all algorithms for provenance expressions of varying sizes of this dataset. Note that our algorithms fare well with respect to the optimal strategies, and in particular Q-value matched the optimal average number of probes in all our experiments. General and RO deviated from the optimal by at most 50%. Even as the dataset grows, the required number of probes remains almost constant. In contrast, the number of probes made by Random grows linearly with the data size, which serves to show the importance of informed probing choices. Freq behaves well for this setting, and deviated from the optimal by at most 43%.

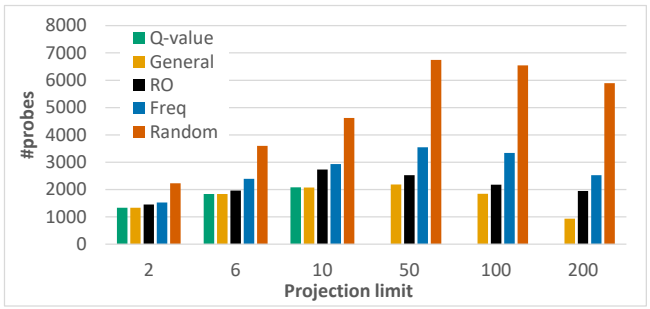
Figure 2b shows the performance of the algorithms for varying probabilities that a probe is answered affirmatively. Again, random choices of probes perform poorly, whereas our algorithms are close-to-optimal: the observed deviation from the optimal strategy were at most 4.5% for Q-value and similarly for General except for probability 0.5 where the latter deviated by 57%. We can also observe here a general trend: for non-read-once provenance, RO is less efficient in proving an expression is False, since it ignores variable frequencies and thus eliminates less terms with each probe returning False. Therefore its comparative performance deteriorates when the probability decreases (up to a 270% deviation). Conversely, Freq performs poorly at proving an expression is True, since it does not account for the likelihood of terms to be True (up to 258% deviation).

We then show experimental results for the **skewed** dataset described above. In this experiment, the optimal strategies are not known, so they do not appear in the figures.

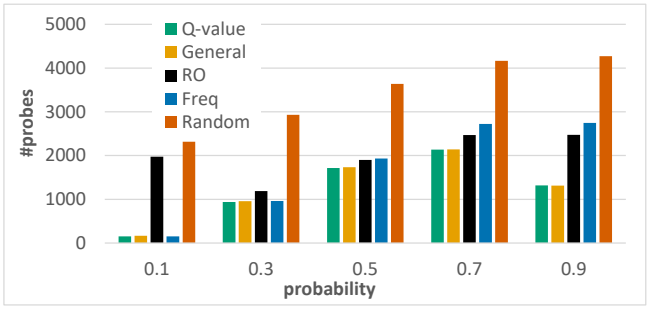
Figure 3a shows the number of probes issued by each of the algorithms when varying the number of joins (corresponding to the DNF term sizes) from 1 to 5. Here again, observe that in all cases, an informed choice of probes is significantly superior to a simple random choice. When the provenance expressions are very simple (at most 2 variables per clause), then choosing the most frequent variable performs well. As expressions become more complex, though, the algorithms that we have developed become significantly superior. In particular, General and Q-value perform the smallest number of probes, since they perform a finer analysis of the provenance structure, and in particular rely on techniques for choosing variables whose probing is effective for either proving True or False. They deviate by only up to 1.3% from the best performing algorithm for any probability.



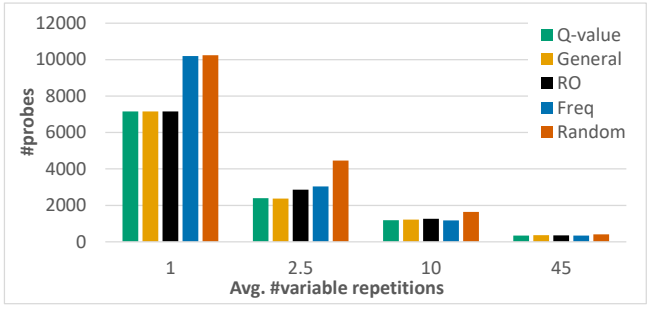
(a) Number of probes for varying number of joins



(b) Number of probes for varying projection limit



(c) Number of probes for varying probabilities



(d) Number of probes for varying number of variable repetitions

Fig. 3: Qualitative experiments for the **skewed** dataset (less probes is better).

In Figure 3b we vary the projection limit as defined in Section IV-C. The provenance of such queries is characterized by a bounded number of DNF terms, and as explained in Section IV-C, while this number is small, brute-force computation of CNF is feasible and hence Q-value is applicable. For larger limits, Q-value is no longer applicable, so we compare the other algorithms. The trend we observe is that as the limit grows, the advantage of our algorithms (Q-value – when applicable) over Freq and Random becomes larger (up to 196% and 528% respectively). Generally, the larger expressions are, the more optimizations our algorithm can perform, because one term/variable evaluated to True can eliminate many other terms. For this experiment we have also tested a *hybrid* approach, which acts like General but switches to Q-value as soon as possible and to RO as soon as the provenance is overall read-once, but as its performance was very close to General, we omit it from the figure.

Figure 3c shows the number of probes issued by each algorithm for valuations drawn at random according to the stated variable probabilities. Similarly to the corresponding experiment over the first dataset, we observe that the advantage of our algorithms over Random is steady and large, whereas the advantage with respect to Freq increases as the probability increases. RO performs comparatively poorly for both low and high probabilities, since term sizes are mostly equal and thus its choice of term is essentially arbitrary.

In Figure 3d we vary the average number of times that a variable is repeated. When this number is low (i.e., the expression is "close to" read-once), the advantage of our solutions is most significant. In particular, when there are no repetitions, provenance becomes overall read-once and RO is provably optimal; in contrast, Freq and Random perform equally badly (deviating by 42% from RO). As observed in the Figure, expressions that are close to read-once are "more difficult", since when variables are often repeated a single probe can eliminate many terms. This exacerbates the importance of our optimality results for read-once expressions.

To conclude, for their respective subclasses, Algorithms RO and Q-Value achieved optimal or near-optimal results – for RO this is guaranteed, and for Q-value this was observed in cases when the optimal is known. Within these subclasses, Algorithm General performed well and deviated from Q-value by at most 10% except for two cases on the ψ-dataset; and beyond these subclasses, it significantly outperformed the alternatives.

Execution time: Our experiments have focused on measuring the number of probes performed by the algorithms, and how well they achieved their optimization goal. The algorithms' execution time, i.e., the time it took them to choose the next probe, was typically a few milliseconds, and up to 1.3 seconds in all of our experiments – so it is much less than the latency of obtaining probe answers in realistic scenarios, e.g., over the Web or with manual answers from peers.

VI. Related Work

Provenance: As described in Section III, we use provenance to track the dependencies of derived data on the input data, and consequently decide on whose consent should be probed. Provenance has been extensively studied, with multiple models and applications, e.g., [18], [19], [7], [4], [20], [16], [21], [15]. Specifically, provenance has been used for access control, which is related to consent management [2],

[22], [3], [5], [23]. Differently from our setting, in these works the collection of atomic permissions is out of scope: they are fully given as input, either before or after the computation of provenance. In contrast, our focus is on achieving consent by probing the permissions of tuples.

Interactive Boolean Evaluation: In this work, we have leveraged previous work to devise efficient algorithms for selecting probes and evaluating Boolean provenance expressions, and specifically [9], [10] and [8] for read-once, DCNF and k-DNF provenance. This problem has been studied in other contexts and under other names, including system testing, e.g., [9], [24] (where it is termed *Sequential System Testing* or *Sequential Diagnosis*), BDD design, e.g., [25], [26], [27] (where it is also called *Discrete Function Evaluation*), active learning [17] (where it is a particular case of *Bayesian Active Learning*) and its connection to other problems such as Stochastic Set Cover [10], [11] (where is is termed *Stochastic Boolean Function Evaluation (SBFE)*). Some work, although in different settings, considered other metrics for BDD efficiency, such as the maximal depth that has been studied in the contexts of probing edges to test graph properties (e.g., [28], [29]) or computing BDDs with minimum depth for input-output sample pairs (e.g., [25], [26]). These works differ from the present work in a few aspects: first, our OPT-PEER-PROBE considers the simultaneous evaluation of multiple, possibly many expressions corresponding to the provenance of multiple tuples, whereas Interactive Boolean Evaluation is concerned with a single formula. The only exception to our knowledge is [10], which proposes constructions for simultaneous evaluation. We have used a similar idea in Algorithm 3. Second, even in OPT-PEER-PROBE-SINGLE the Boolean expressions that we obtain are derived by queries; and there had been no study of the interplay between query classes and the performance of Interactive Boolean Evaluation over their provenance. Last, the works most related to ours are theoretical, and do not include an empirical study of algorithm performance.

Data sharing: The theory and practice of managing sharing permissions have been extensively studied in different contexts, including social networks (see below), distributed systems (e.g., [2], [5]), cloud services (e.g., [22], [30], [31]), Web applications (e.g., [32]), databases (e.g., [3], [4]), and many other areas. Specifically, social networks are commonly used for data sharing while facilitating social interactions in which access control is crucial. Access control in such networks is generally managed in a coarse level, namely, peers who have originally contributed data either have no control over the re-sharing of their data, or give a broad consent to re-sharing within some group (e.g., friends of friends), or disallow re-sharing altogether [33].

Studies on data sharing in social networks aim at refining this approach by studying on how access policies [34], privacy [33], trust [35] and willingness to share data [36] can be defined over the network. Different cryptographic means and implementation designs have been proposed for this purpose [37], [38], [39], [40], [35]. While our work focuses on establishing whether consent for sharing is given or not, cryptographic techniques as in [38] may be employed to *enforce* these policies. Multiple ownership over data items has also been considered in this context [41], [39], [35], focusing on enforcing a policy that adheres to the individual policies of the involved peers. However, these studies do not consider data derivation/querying but rather the sharing of atomic items, which leads to technical problems different from ours.

Predicting probe answers and probabilities: In the present work we have assumed that probabilities for probe answers are given in advance. This could be done, e.g., by coarse means like computing the average likelihood for consent in past probes. Finer-grained estimation could potentially be obtained through work on (semi-)automatically computing access control policies, which captures relationships between peers based on example permissions [42], [43]; evaluating the credibility of peers [36]; mining user roles, which may allow distinguishing peers and their relationships inside an organization [44], [45]; and using game-theoretic considerations with respect to risk minimization [41].

VII. Possible Extensions

In our definition of the problem studied here, we have made several design choices. In this section, we explore how alternative choices could be studied in our framework, and how they would affect our results.

Beyond SPJU: A natural first question is extending our results beyond SPJU queries. Fortunately, our framework is modular, so we could rely on the long line of results on provenance for various query languages, e.g., Datalog [19]. Extending our results to more expressive query languages is an intriguing goal for future research.

Different problem variants: In this work, we have focused on optimizing the expected cost of full evaluation of consent for sharing certain output tuples. Other variants in ongoing research consider other optimization metrics, such as optimizing the number of probes per peer, or the worst-case number of probes; or other optimization variables, such as optimizing the number of evaluated expressions for a fixed number of probes.

Different semantics for consent: As explained in Section II, our semantics for consent naturally follows previously proposed possible-worlds semantics for Access Control. There are other reasonable consent semantics, e.g., based on privacy models such as K-anonymity.

Different models for probes and answers: Our problem the setting involves sequential probing of peers. In general, probing can be asynchronous or sent in batches, to reduce latency at the expense of possibly making unnecessary probes. Similarly, we have assumed that peers always answer, but this may not be the case; and we have assumed a uniform probing cost, whereas the cost could differ across peers.

Beyond independent probabilities: We have assumed that the probabilities for consent for each tuple are given and independent of each other. In general, more complex probability distributions are conceivable and may be concisely represented: e.g., probe answers may be constrained by logical implications captured by a Bayesian Network or an ontology.

Beyond unique annotations: We have further assumed that each tuple is uniquely annotated by a consent variable – but in general there may be logical "blocks" of tuples for which consent is wither given or not given uniformly. This kind of dependency between probe answers is particularly interesting, and it has significant effects on our results as they lead to co-occurrences of variables in the Boolean expressions. Extending our results to such settings would likely require other constraints on the query and/or database shape.

VIII. Conclusion

We have proposed in this paper a new framework for managing consent in shared databases. Consent is managed at the tuple level, and we formalize the problem of determining consent w.r.t. *query output tuples* via probing peers for their consent w.r.t. relevant *input database tuples*. We have studied the complexity of the resulting optimization problem, showing intractability in general and identifying tractable subclasses and approximate solutions. Our experimental study has validated the effectiveness of our algorithms for the various classes, demonstrating their optimal or near-optimal performance in different cases and their superiority with respect to baseline alternatives.

As mentioned above, we have followed in this paper particular design choices, but many others may also make sense. Thus, this paper is by no means the last word on the subject, but rather we view it as opening up a new area of investigation.

References

[1] European Council and European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). European Commission.

[2] S. Abiteboul, P. Bourhis, and V. Vianu, "A formal study of collaborative access control in distributed datalog," in *ICDT*, 2016.

[3] J. N. Foster, T. J. Green, and V. Tannen, "Annotated XML: queries and provenance," in *PODS*, 2008.

[4] G. Karvounarakis, Z. G. Ives, and V. Tannen, "Querying data provenance," in *SIGMOD*, 2010.

[5] V. Z. Moffitt, J. Stoyanovich, S. Abiteboul, and G. Miklau, "Collaborative access control in Webdamlog," in *SIGMOD*, 2015.

[6] Y. Amsterdamer and O. Drien, "PePPer: Fine-grained personal access control via peer probing," in *ICDE*, 2019.

[7] T. Imielinski and W. L. Jr., "Incomplete information in relational databases," *J. ACM*, vol. 31, no. 4, 1984.

[8] S. R. Allen, L. Hellerstein, D. Kletenik, and T. Ünlüyurt, "Evaluation of monotone DNF formulas," *Algorithmica*, vol. 77, no. 3, 2017.

[9] E. Boros and T. Ünlüyurt, "Sequential testing of series-parallel systems of small depth," in *Computing tools for modeling, optimization and simulation*. Springer, 2000.

[10] A. Deshpande, L. Hellerstein, and D. Kletenik, "Approximation algorithms for stochastic Boolean function evaluation and stochastic submodular set cover," in *SODA*, 2014.

[11] H. Kaplan, E. Kushilevitz, and Y. Mansour, "Learning with attribute costs," in *STOC*, 2005.

[12] Y. Amsterdamer and O. Drien, "Towards fine-grained data access control through active peer probing," in *EDBT*, 2020.

[13] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[14] T. J. Green, G. Karvounarakis, and V. Tannen, "Provenance semirings," in *SIGMOD*, 2007.

[15] D. Suciu, D. Olteanu, C. Ré, and C. Koch, *Probabilistic Databases*. Morgan & Claypool, 2011.

[16] S. Roy, V. Perduca, and V. Tannen, "Faster query answering in probabilistic databases using read-once functions," in *ICDT*, 2011.

[17] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *J. Artif. Intell. Res.*, vol. 42, 2011.

[18] Y. Amsterdamer, D. Deutch, and V. Tannen, "Provenance for aggregate queries," in *PODS*, 2011.

[19] D. Deutch, T. Milo, S. Roy, and V. Tannen, "Circuits for datalog provenance," in *ICDT*, 2014.

[20] D. Olteanu and J. Zavodny, "Factorised representations of query results: size bounds and readability," in *ICDT*, 2012.

[21] P. Senellart, L. Jachiet, S. Maniu, and Y. Ramusat, "ProvSQL: Provenance and probability management in PostgreSQL," *PVLDB*, vol. 11, no. 12, 2018.

[22] A. Bates, B. Mood, M. Valafar, and K. R. B. Butler, "Towards secure provenance-based access control in cloud environments," in *CODASPY*, 2013.

[23] J. Park, D. Nguyen, and R. S. Sandhu, "A provenance-based access control model," in *PST*, 2012.

[24] T. Ünlüyurt, "Sequential testing of complex systems: a review," *Discrete Applied Mathematics*, vol. 142, no. 1-3, 2004.

[25] F. Cicalese, E. S. Laber, and A. M. Saettler, "Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost," in *ICML*, 2014.

[26] A. Fiat and D. Pechyony, "Decision trees: More theoretical justification for practical algorithms," in *ALT*, 2004.

[27] O. Keren, "Reduction of average path length in binary decision diagrams by spectral methods," *IEEE Trans. Computers*, vol. 57, no. 4, 2008.

[28] J. Kahn, M. E. Saks, and D. Sturtevant, "A topological approach to evasiveness," in *FOCS*, 1983.

[29] R. Scheidweiler and E. Triesch, "A lower bound for the complexity of monotone graph properties," *SIAM J. Discrete Math.*, vol. 27, no. 1, 2013.

[30] P. Derbeko, S. Dolev, E. Gudes, and S. Sharma, "Security and privacy aspects in mapreduce on clouds: A survey," *Computer Science Review*, vol. 20, 2016.

[31] M. G. Solomon, V. S. Sunderam, and L. Xiong, "Towards secure cloud database with fine-grained access control," in *DBSec*, 2014.

[32] G. S. Tuncay, S. Demetriou, and C. A. Gunter, "Draco: A system for uniform and fine-grained access control for web code on android," in *CCS*, 2016.

[33] L. Yu, S. M. Motipalli, D. Lee, P. Liu, H. Xu, Q. Liu, J. Tan, and B. Luo, "My friend leaks my privacy: Modeling and analyzing privacy in social networks," in *SACMAT*, 2018.

[34] Y. Cheng, J. Park, and R. S. Sandhu, "An access control model for online social networks using user-to-user relationships," *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 4, 2016.

[35] N. C. Rathore, P. Shaw, and S. Tripathy, "Collaborative access control mechanism for online social networks," in *ICDCIT*, 2016.

[36] E. Gudes and N. Voloch, "An information-flow control model for online social networks based on user-attribute credibility and connection-strength factors," in *CSCML*, 2018.

[37] A. K. Abdulla and S. Bakiras, "HITC: data privacy in online social networks with fine-grained access control," in *SACMAT*, 2019.

[38] M. Davidson, T. Tassa, and E. Gudes, "Content sharing schemes in DRM systems with enhanced performance and privacy preservation," *Journal of Computer Security*, vol. 24, no. 6, 2016.

[39] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, and S. Ioannidis, "SAMPAC: socially-aware collaborative multi-party access control," in *CODASPY*, 2017.

[40] I. Kayes and A. Iamnitchi, "Privacy and security in online social networks: A survey," *Online Social Networks and Media*, vol. 3-4, 2017.

[41] H. Hu, G. Ahn, Z. Zhao, and D. Yang, "Game theoretic analysis of multiparty access control in online social networks," in *SACMAT*, 2014.

[42] G. P. Cheek and M. Shehab, "Policy-by-example for online social networks," in *SACMAT*, 2012.

[43] P. Iyer and A. Masoumzadeh, "Active learning of relationship-based access control policies," in *SACMAT*, 2020.

[44] N. Gal-Oz, Y. Gonen, and E. Gudes, "Mining meaningful and rare roles from web application usage patterns," *Comput. & Secur.*, vol. 82, 2019.

[45] T. Wang, M. Srivatsa, and L. Liu, "Fine-grained access control of personal data," in *SACMAT*, 2012.

APPENDIX

PROOFS FOR SECTION III (SOLUTION OVERVIEW)

We now give the full proof of Theorem III.5:

Theorem III.5. *For arbitrarily large integers n, there is a monotone Boolean formula with n variables such that, for any probability distribution π mapping all variables to values different from 0 and 1, there is a BDD of $\mathrm{O}(\log n)$ expected depth as well as a BDD of expected depth $\Omega(n)$. Moreover, the $\mathrm{O}(\log n)$ BDD is optimal for a constant distribution $\pi : X \to p$ giving the same probability $p \in (0,1)$ to every variable.*

We define the monotone DNF formula $\psi_0 := (w \wedge x) \vee (x \wedge y) \vee (y \wedge z)$ and recursively define $\psi_{i+1} := (u_i \wedge \psi_i) \vee (u_i \wedge v_i) \vee (v_i \wedge \psi_i')$ where u_i, v_i are fresh variables and ψ_i' is obtained by consistently replacing the variables of ψ_i by fresh variables, as explained in the proof sketch.

By this construction for ψ_i we have 2^i copies of ψ_0, and $2 + 4 + 8 + \ldots + 2^i = 2 \cdot (2^i - 1)$ copies of the variables u_i, v_i, leading to $6 \cdot 2^i - 2 = \Theta(2^i)$ variables in total. Given a natural number m, we can clearly choose a large enough i such that $n = \mathrm{vars}(\psi_i) > m$.

We will next illustrate a BDD of expected depth $\mathrm{O}(i) = \mathrm{O}(\log n)$: start by probing u_i and v_i. By our construction, if $u_i = v_i$ then ψ_i is evaluated and we are done. Otherwise, assume w.l.o.g. that $u_i =$ True and $v_i =$ False. We are left with True $\wedge\, \psi_{i-1} = \psi_{i-1}$, and can proceed recursively to query u_{i-1} and v_{i-1}. At the worst case this recursion reaches ψ_0 which can then be evaluated by 3 probes - leading to a total of $2i + 3 = \mathrm{O}(i)$ probes. The worst possible depth is $O(i)$, so the same is true of the expected depth.

In contrast, consider a BDD that first evaluates the 2^i copies of ψ_0. Their variables are mutually distinct, they are all useful (i.e., it can in fact be important to probe them), and by construction we must evaluate either u_i or v_i or both to evaluate ψ_i, regardless of the values of ψ_{i-1} and ψ_{i-1}'. Consequently, the evaluation of each copy will be done independently, leading to $2 \cdot 2^i = \Theta(n)$ probes in every case, hence also in expectation.

We now argue that the first BDD that we defined is optimal, which is somewhat technical. Fix the value $0 < p < 1$, and write p_i^+ the probability that ψ_i evaluates to true and p_i^- the probability that it evaluates to false, for all i. We first claim:

Claim A.1. *For all p and i, we have: $p_i^+ \leq 2p$.*

Proof. The proof is by induction.

For the base case, consider ψ_0, and distinguish all possible outcomes depending on the value of x and y. If they are both true, with probability p^2, then ψ_0 is true. If they are both false, with probability $(1-p)^2$, then ψ_0 is false. In the other cases, with probability $2p(1-p)$, ψ_0 is true with probability p (by looking at w, or at z, depending on which of x and y is true). In other words, we have $p_0^+ = p^2 + 2p^2(1-p)$. Let us show that $p_0^+ \leq 2p$, by studying the polynomial $p^2 + 2p^2(1-p) - 2p$ and showing that it is always negative or 0. We can simplify by p to obtain: $-2p^2 + 3p - 2$. The discriminant is $3^2 - 4 \times 2 \times 2 = -7$, so this polynomial has no roots and is always negative, so we have $p_0^+ \leq 2p$ for all $0 < p < 1$.

For the induction, consider ψ_i, and distinguish all possible outcomes depending on the value of u_i and v_i. If they are both true, with probability p^2, then the formula is true, if they are both false then the formula is false, and otherwise the formula is true with probability p_{i-1}^+. So we have: $p_i^+ = p^2 + 2p(1-p)p_{i-1}^+$. By induction hypothesis, we have $p_{i-1}^+ \leq 2p$. So we have $p_i^+ \leq p^2 + 2p(1-p)2p$. Let us show that this is $\leq 2p$, by studying the sign of $p^2 + 4p^2(1-p) - 2p$. We can simplify by p again and obtain $-4p^2 + 5p - 2$. The discriminant is $5^2 - 4 \times 2 \times 4 = -7$, so again this polynomial has no roots and is always negative, so we have $p_i^+ \leq 2p$ for all $0 < p < 1$. □

We next claim:

Claim A.2. *For all p and i, we have: $p_i^- \leq 2(1-p)$.*

Proof. This is the same proof as before, but reversing the roles of true and false. Specifically, for ψ_0, it is false with probability $(1-p)^2$ or, with probability $2p(1-p)$, with probability $(1-p)$. So we have $p_0^- = (1-p)^2 + 2p(1-p)(1-p)$, and we are comparing this to $2(1-p)$, so subsituting p for $1-p$ and applying the argument of the base case of the previous claim we have $p^+{-_0} \leq 2(1-p)$.

For the induction case of ψ_i, the formula is false with probability $(1-p)^2 + 2p(1-p)p_{i-1}^-$, so we can again apply the induction hypothesis, compare to $2(1-p)$, and conclude as before. □

We are now ready to argue that our choice of BDD for ψ_n is optimal. The claim is obvious for $n = 0$, so we show it for all $n > 0$. First, notice that you cannot conclude that the formula is true or false without having observed one of u_n or v_n; so there is an optimal BDD which probes one of these first. As they play symmetric roles, say we probe u_n. Now, if u_n is false, then we can simplify and are left with $v_n \wedge \psi_{n-1}$. If u_n is true, then we are left with $v_n \vee \psi_{n-1}$.

These are respectively a conjunction and disjunction, with the two branches not sharing any variables. In either case, we can always choose an optimal BDD that will first test one branch of the operator, and then the other if necessary; the only question is how to order them. We claim that, in both cases, it is never worse to start with v_n. This will conclude the proof, as the rest of the BDD will study the truth value of ψ_{n-1}, and we can finish the argument by induction.

We first consider the case of $v_n \vee \psi_{n-1}$. The first strategy is to first probe v_n, paying 1 with probability p (if it is true), and otherwise probing ψ_{n-1}. Let C_{n-1} be the expected cost of the latter; we pay $C_{n-1}+1$ with probability $1-p$. Hence the expected cost of this strategy is $p+(1-p)(1+C_{n-1})$, or equivalently $1+(1-p)C_{n-1}$. We call this quantity Γ_1.

The second strategy is to first determine the truth status of ψ_{n-1}, and if it is false then probing v_n. Let us write C^+_{n-1} and C^-_{n-1} the expected cost of an optimal decision tree on ψ_{n-1} conditioned respectively by the fact that it will evaluate to true or that it will evaluate to false. We have by definition: $C_{n-1} = p^+_{n-1}C^+_{n-1} + p^-_{n-1}C^-_{n-1}$. This strategy has expected cost: $p^+_{n-1}C^+_{n-1} + p^-_{n-1}(1+C^-_{n-1})$. Let us expand and regroup to obtain: $C_{n-1} + p^-_{n-1}$, or equivalently $C_{n-1} + 1 - p^+_{n-1}$. We call this quantity Γ_2.

Let us compare both strategies by doing the difference $\Gamma_1 - \Gamma_2$. We obtain: $p^+_{n-1} - pC_{n-1}$. Now, observe that for all $n > 0$, we have $C_{n-1} \geq 2$, as we must always test at least two variables to decide (already in the case of ψ_0). So this quantity is $\leq p^+_{n-1} - 2p$. By Claim A.1, we conclude that this is negative or 0, so indeed strategy 1 is no worse than strategy 2.

We now consider the case of $v_n \wedge \psi_{n-1}$. The reasoning is the same: strategy 1 will have performance $(1-p)+p(1+C_{n-1})$, i.e., $1-pC_{n-1}$, and strategy 2 will have performance $C_{n-1}+p^+_{n-1}$, i.e., $C_{n-1}+1-p^-_{n-1}$. The difference is $p^-_{n-1}-(1-p)C_{n-1}$, and again this is $\leq p^-_{n-1} - 2(1-p)$, and we conclude this time by Claim A.2. This concludes the proof.

Proofs for Section IV (Complexity and Algorithms)

Proposition IV.2. *1) For each SPJU query Q there exists a value k (intuitively, the maximal number of joins in Q) such that for every shared database $\bar{D}$, the provenance of each tuple in $Q(\bar{D})$ may be represented in a monotone k-DNF form. This form may further be constructed in PTIME in data complexity (i.e., as a function of $\bar{D}$).*

2) Conversely, for every monotone k-DNF formula φ, there exists an SPJ query Q depending only on k, a shared database $\bar{D}$ whose size is linear in φ and a probability distribution π such that the query output $Q(\bar{D})$ is a singleton tuple whose provenance is equivalent to φ.

Proof. The first part of the proposition holds with k being the maximal number of joins of a conjunctive query within Q. To observe that this is the case, note that conjunctions in the provenance construction are associated with joins and disjunctions are associated with projection and union.

For the second part, we again exploit the correspondence between query and Boolean operations. Given a monotone k-DNF formula f, consider a DB $\bar{D}$ with two relations. Relation R encodes the variables of f, where for each variable x in f we have a tuple $R(x)$ (using x as a value) annotated by x. Relation S encodes the terms of f such that for each term $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ in f we have a tuple $S(x_1, x_2, \ldots, x_k)$ (using $x_1, x_2, \ldots, x_k$ as values) and annotated by a fresh variable y. If a term is of size $< k$ we can repeat one of its variables to obtain an equivalent term of size exactly k. For the probability distribution over variables, we will use a distribution that is identical to the original distribution on f's variables, and the probability of the fresh variables to be True equals 1, which allows us to simplify them away in the provenance.

The query Q is a binary CQ fixed for k (no unions, using only equality joins and a projection on all variables) : $\mathrm{ans}() : -S(z_1, \ldots, z_k)R(z_1)\ldots R(z_k)$. By this construction, each tuple in the join result corresponds to a term in f and has a provenance of the form of a conjunction $x_1 \wedge x_2 \wedge \ldots \wedge x_k \wedge y$, where every x_i stands for an original variable, and y is a fresh variable that can be ignored since its value is fixed to 1. Then projecting out all the variables yields the disjunction of these conjunctions which is equivalent to f. □

Theorem IV.9. *There is a fixed SJ query Q for which the* OPT-PEER-PROBE *problem is NP-hard, even when all variables have the same probability.*

Proof. We reuse the same schema as in the proof of Theorem IV.15, and reuse the same query without the projection, that is, $Q(x,y) : Var(x), Var(y), Clause(x,y)$. The provenance of every tuple of Q on a uniquely 1-annotated database is such that each result tuple has as provenance a conjunction of three variables.

We consider the Boolean function φ with multiple return values defined by the various rows of the result of Q on an input uniquely 1-annotated database. We extend the notion of a certificate to functions with multiple outputs: a *minimum-cardinality* $\mathbf{0}$*-certificate contained in* $\mathbf{0}$ is a subset of the variables having minimal cardinality so that setting them to 0 suffices to witness that all outputs of the function are 0. Formally, all valuations that extend the certificate are such that the value of the multivalued function is $\mathbf{0}$.

We now consider the single-valued Boolean function φ' defined as the disjunction of the output values of φ. Again, this is the same function used in the proof of Theorem IV.15, and we have shown that finding a minimum-cardinality certificate contained in $\mathbf{0}$ for this function is NP-hard. But now, observe that a partial assignment is such a certificate for φ iff it is for φ'.

This is because witnessing that φ' evaluates to 0 means witnessing that every disjunct evaluates to 0, which is the same as witnessing that all output values of φ are 0. Thus, finding a minimum-cardinality $\mathbf{0}$-certificate for φ contained in $\mathbf{0}$ is NP-hard too.

The only remaining thing is to note that the latter problem reduces to OPT-PEER-PROBE, in the same way as in the proof of Theorem 1 of [8]. This is because Lemma 1 straightforwardly extends to decision trees computing multivalued functions: if all variables have sufficiently low probability (the same as in their lemma statement), then the performance of a decision tree is dominated by its performance on the assignment $\mathbf{0}$, so any optimal decision tree must be testing some minimal-cost certificate contained in $\mathbf{0}$. Thus, OPT-PEER-PROBE is also NP-hard, concluding the proof. □

Theorem IV.10. *There is a fixed SPU query Q for which the* OPT-PEER-PROBE *problem is NP-hard, even when all variables have the same probability.*

Proof. We reduce from the VERTEX COVER problem on graphs, which is already NP-hard for cubic graphs, i.e., graphs where all vertices have degree 3.

We consider a table R with 4 attributes, and the SPU query $Q : \pi_2(R) \cup \pi_3(R) \cup \pi_4(R)$. For a given cubic graph G, we create the R-instance having one tuple (v, e_1, e_2, e_3) per vertex v, where e_1 and e_2 and e_3 are its three incident edges. The provenance annotation of every tuple is identified with its first element v.

The result of the query gives us one tuple per edge, whose provenance is the disjunction of the provenance annotation of its two incident vertices.

Let us now argue that finding an optimal decision tree for the multivalued Boolean function where we wish to identify the truth status of all tuples is NP-hard. To do so, as in the proof of Theorem IV.10, we use a variant of Lemma 1 from [8]. First notice that this lemma straightforwardly extends, with the same proof, to variables with a high probability of being true, to argue that on an assignment where all variables are true, an optimal decision tree must be testing the variables of a minimum-cost 1-certificate. Now, for the multivalued function that we consider, on the assignment where all variables are true, an optimal decision tree must be testing all variables of a minimum-cost certificate that shows that all functions evaluate to 1.

We are now ready to conclude the reduction. Assume we can solve OPT-PEER-PROBE efficiently for the query Q. Then, as we argued, for any cubic graph G, we can design an algorithm that follows an optimal decision tree for the multivalued function having one function per disjunctive clause $x \vee y$ corresponding to an edge $\{x, y\}$ of G. As we argued, there is an assignment of probabilities to variables such that the optimal decision tree determines a minimum-cost certificate showing that all functions evaluate to 1, i.e., a minimum subset of vertices such that all edges are covered, which solves the VERTEX COVER problem on the input cubic graph. This concludes the proof. □

Theorem IV.15. OPT-PEER-PROBE *and* OPT-PEER-PROBE-SINGLE *are NP-hard (in data complexity) for SPJ queries, even on shared databases where all tuples have the same probability.*

Proof. Consider a schema having a 1-ary relation $Vars$ and a 2-ary relation $Clauses$, and let us define a Boolean CQ on this schema by: $Q : \exists x\, y\ \ Vars(x), Vars(y), Clauses(x, y)$. We show that the OPT-PEER-PROBE-SIGNLE problem is NP-hard for this query.

To do this, let us describe the Boolean provenance of this query on an arbitrary input uniquely-1-annotated database D. For each row of the $Clauses$ table, we write $t_{a,b}$ for the unique variable annotating a row (a, b), and likewise for each row of the $Vars$ table we write x_a for the unique variable annotating a row (a). We let $Clauses'$ be the subset of $Clauses$ that joins with $Vars$, i.e., those tuples (a, b) of $Clauses$ where a and b also occur in $Vars$. The provenance of Q on a database D is then $\bigvee_{(a,b)\in Clauses'} x_a \wedge x_b \wedge t_{a,b}$. In other words: the Boolean expressions that can be obtained as the provenance of the fixed query Q on some database D are precisely the subclass of monotone 3-DNFs characterized by the following property: they are a monotone 2-DNF on a subset of the variables (the $x_\bullet$'s), and we add to every clause a fresh variable occurring only in that clause (the $t_{\bullet,\bullet}$'s). In other words, they are the monotone 3-DNFs for which there a subset X of variables guaranteeing that every clause consists of exactly two variables from X and a variable not in X which is fresh to that clause, and there are no two clauses containing the same pair of variables from X. Let us call this fragment of Boolean formulas the *distinguished 3-DNFs*.

To conclude the hardness proof, it suffices to argue that the OPT-PEER-PROBE-SIGNLE problem is NP-hard for distinguished 3-DNFs. Throughout the proof, we will always assume that all variables for Boolean formulas have the same probability, when counting the performance of OPT-PEER-PROBE-SIGNLE or the cost of certificates for a formula. We show this hardness claim by an adaptation of the hardness proof in [8], Theorem 1. They reduce from the VERTEX COVER problem, where we are given an undirected graph $G = (V, E)$ and must find a minimum cardinality vertex cover, i.e., a subset of V such that every edge in E contains some vertex of the subset. They observe that, letting $\varphi : \bigvee_{\{u,v\}\in E} x_u \wedge x_v$ be the monotone 2-DNF built from G in the expected way, then minimum cardinality vertex covers are in one-to-one correspondence with minimum-cost certificates for φ contained in $\mathbf{0}$, i.e., a minimum cardinality subset of the variables such that setting all variables of that set to False suffices to witness that φ must evaluate to False. (The intuitive argument for this correspondence is that such

a certificate must contain at least one variable from each clause.) Hence, it is also NP-hard to find such a minimum-cost certificate for a monotone 2-DNF.

We now argue that the same is true of distinguished 3-DNFs, by reducing from the latter problem. Indeed, consider any monotone 2-DNF φ, and build in PTIME a distinguished 3-DNF φ' by adding a fresh variable to every clause. Now, consider a minimum-cost certificate for φ' contained in $\mathbf{0}$. Observe that we can modify it to another such minimum-cost certificate which does not select any of the fresh additional variables of every clause. This is because these variables only occur in that clause, so they can always be replaced by some other variable of the clause (by minimality of the certificate, these variables are not already part of the certificate). So, if we can find a suitable certificate for φ', then we can in PTIME modify it iteratively to a certificate that does not involve the fresh variables. It is now clear that this certificate is also a suitable certificate of φ, and that it has minimum cost (because any certificate for φ also yields a certificate for φ' with the same cost). Thus, we have shown that it in NP-hard to find a minimum-cost certificate contained in $\mathbf{0}$ for a distinguished 3-DNF.

Finally, we show that we can reduce from this problem to the OPT-PEER-PROBE-SINGLE problem. This is exactly as in the proof of Theorem 1 of [8], using Lemma 1 which argues that for any Boolean function, there is a choice of probabilities giving the same probability to every variable (intuitively a very low one), so that any optimal decision tree must be testing on assignment $\mathbf{0}$ exactly the variables of some minimum-cost certificate contained in $\mathbf{0}$ for the function. (Intuitively: the assignment $\mathbf{0}$ is so likely that the performance of the tree is dominated by its performance on that specific assignment, and any tree doing some other test is less efficient on $\mathbf{0}$, which can never be offset by any increase in efficiency even on all other assignments.) Thus, the problem of finding a minimum-cost certificate contained in $\mathbf{0}$ for a distinguished 3-DNF reduces to OPT-PEER-PROBE for the same formula, even if probability given to all variables is the same. This concludes the proof for OPT-PEER-PROBE-SINGLE.

□