On Constrained Open-World Probabilistic Databases

Anonymous authors

Abstract

Increasing amounts of available data have led to a heightened need for useful ways of representing large-scale probabilistic knowledge bases. One such approach is to use a probabilistic database, a model with strong assumptions that allow for efficiently answering many interesting queries. Recent work on open-world probabilistic databases strengthens the semantics of these probabilistic databases by discarding the assumption that any information not present in the data must be false. In this paper we build upon open-world probabilistic databases, strengthening the semantics further by using constraints to restrict this open world in a reasonable way. We establish a basic hardness result subject to these constraints, showing that the addition of constraints causes a fundamental change in the difficulty of query evaluation and opening an interesting new theoretical direction. Finally, we propose an efficient approximation for a large class of interesting queries, providing bounds with their guaranteed tightness.

1. Introduction

An ubiquitous pursuit in the study of knowledge base representation is the search for a model that can represent uncertainty while simultaneously answering interesting queries efficiently. The key underlying challenge is that these goals are at odds with each other. Modelling uncertainty requires additional model complexity. Similarly, the ability to answer meaningful queries usually demands fewer model assumptions. Both of these properties are at odds with the key limiting factor of tractability: success in the first two goals is not nearly as impactful if it is not achieved efficiently.

One approach towards achieving this goal is to begin with a simple model such a probabilistic database (PDB) [Suciu et al., 2008, Van den Broeck and Suciu, 2017, Suciu et al., 2011]. A PDB models uncertainty, but is inherently simple and makes very strong independence assumptions and closed-world assumptions allowing for tractability on a very large class of queries [Dalvi and Suciu, 2007, 2012]. However, PDBs can fall short under non-ideal circumstances, as their semantics are brittle to incomplete knowledge bases [Ceylan et al., 2016].
To bring PDBs closer to our desired goal, Ceylan et al. [2016] propose open-world probabilistic databases, wherein the semantics of a PDB are strengthened to relax the closed-world assumption. While open-world probabilistic databases maintain a large class of tractable queries, the semantics of queries are such that they are now significantly less meaningful. This is true both in an intuitive and numerical sense.

We propose taking the next step in this direction, further strengthening the semantics of open-world probabilistic databases by constraining the average probability allowed for a relation. These constraints work at the schematic level, meaning no additional per-item information is required. They are practically motivated by knowledge of summary statistics. A theoretical analysis shows that constraints fundamentally changes the difficulty landscape of queries, leading us to propose a general purpose approximation scheme.

The rest of the paper is organized as follows: Section 2 provides necessary background on relational logic and probabilistic databases, as well as a motivation and introduction to open-world probabilistic databases. Section 3 motivates and introduces our construction for constraining open-world probabilistic databases. Section 4 analyses exact solutions subject to these constraints, showing that the problem is in general hard, even in some cases where standard probabilistic database queries are tractable. Section 5 investigates efficient and provably bounded approximation schemes. Section 6 provides a holistic discussion of our findings, and summarizes interesting directions that we leave as open problems.

2. Background & Motivation

This section provides background in probabilistic databases, as well as reintroducing and motivating their open-world counterparts. Much of the notation and definitions are adapted from Ceylan et al. [2016].

2.1 Relational Logic

We now describe necessary background from function-free finite-domain first order logic. An atom \( R(x_1, x_2, \ldots, x_n) \) consists of a predicate \( R \) of arity \( n \), together with \( n \) arguments. These arguments can either be constants or variables. A ground atom is an atom which contains no variables. A formula is a series of atoms combined with conjunctions (\( \land \)) or disjunctions (\( \lor \)), and with quantifiers \( \forall, \exists \). A substitution \( Q[x/t] \) replaces all occurrences of \( x \) by \( t \) in a formula \( Q \).

A relational vocabulary \( \sigma \) is comprised of a set of predicates \( \mathcal{R} \) and a domain \( \mathcal{D} \). Using the Herbrand semantics [Hinrichs and Genesereth, 2006], the Herbrand base of \( \sigma \) is the set of all ground atoms possible given \( \mathcal{R} \) and \( \mathcal{D} \). A \( \sigma \)-interpretation \( \omega \) is then an assignment of truth values to every element of the Herbrand base of \( \sigma \). We say that \( \omega \) models a formula \( Q \) whenever \( \omega \models Q \). This is denoted by \( \omega \models Q \).

2.2 Probabilistic Databases

A reasonable starting point for the target knowledge base to construct would be to use a traditional relational database. Using the standard model-theoretic view [Abiteboul et al., 1995], a relational database for a vocabulary \( \sigma \) is a \( \sigma \)-interpretation \( \omega \). Less formally, a relational database consists of a series of relations, each of which corresponds to a predicate.
(a) Example relational database. Notice that the first row of the right table corresponds to the atom \( \text{CoAuthor}(\text{Einstein}, \text{Erdős}) \).

(b) Example probabilistic database. Tuples are now of the form \( \langle t : p \rangle \) where \( p \) is the probability of the tuple being present.

Each relation consists of a series of rows, also called tuples, each of which corresponds to an atom of the predicate being true. Any atom not appearing as a row in the relation is considered to be False, which corresponds to making the closed-world assumption [Reiter, 1981]. Figure 1a shows an example relational database.

While relational databases are an attractive option, their deterministic nature leads to a few shortcomings. A common way to gather a large knowledge base is to apply some sort of statistical model [Carlson et al., 2010, Suchanek et al., 2007, Peters et al., 2014, Dong et al., 2014] which returns a probability value for potential tuples. Adapting the output of such a model to a relational databases involves thresholding on the probability value, discarding valuable information along the way. To circumvent this problem, the standard approach is to introduce a probabilistic database (PDB) which assigns each tuple a probability.

**Definition 1.** A (tuple-independent) probabilistic database \( \mathcal{P} \) for a vocabulary \( \sigma \) is a finite set of tuples of the form \( \langle t : p \rangle \) where \( t \) is a \( \sigma \)-atom and \( p \in [0, 1] \). Furthermore, each \( t \) can appear at most once.

Given such a collection of tuples and their probabilities, we are now going to define a distribution over relational databases. The semantics of this distribution are given by treating each tuple as an independent random variable.

**Definition 2.** A probabilistic database \( \mathcal{P} \) for vocabulary \( \sigma \) induces a probability distribution over \( \sigma \)-interpretations \( \omega \):

\[
P_\mathcal{P}(\omega) = \prod_{t \in \omega} P_\mathcal{P}(t) \prod_{t \notin \omega} (1 - P_\mathcal{P}(t))
\]

where

\[
P_\mathcal{P}(t) = \begin{cases} p & \text{if } \langle t : p \rangle \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}
\]

Notice this last statement is again making the closed-world assumption: any tuple that we have no information about is assigned probability zero. Figure 1b shows an example probabilistic database.

### 2.3 Queries

In relational databases, the fundamental task we are interested in solving is how to answer queries. The same is true for probabilistic databases, with the only difference being that we are now interested in probabilities over queries. In particular, we are interested in queries
which are fully quantified - also known as Boolean queries. On a relational database, this corresponds to a query which has an answer of True or False.

For example, on the database given in Figure 1a, we might ask if there is a scientist who is has a coauthor:

$$Q_1 = \exists x. \exists y. S(x) \land \text{CoA}(x, y)$$

If we instead asked this query of the probabilistic database in Figure 1b, we would be computing the probability by summing over the worlds in which the query is true:

$$P(Q_1) = \sum_{\omega \models Q_1} P(\omega)$$

Queries of this form that are a conjunction of atoms are called conjunctive queries. They are commonly shortened as:

$$Q_1 = S(x), \text{CoA}(x, y).$$

A disjunction of conjunctive queries is known as a union of conjunctive queries (UCQ). UCQs have been shown to live in a dichotomy of efficient evaluation [Dalvi and Suciu, 2012]: computing the probability of a UCQ is either polynomial in the size of the database, or it is \#P-hard. This property can be checked through the syntax of a query, and we say that a UCQ is safe if it admits efficient evaluation. In the literature of probabilistic databases [Suciu et al., 2008, Dalvi and Suciu, 2012], as well as throughout the rest of this paper, UCQs are the primary query object studied.

2.4 Open-World Probabilistic Databases

In the context of automatically constructing a knowledge base, as is done in for example NELL [Carlson et al., 2010] or Google’s Knowledge Vault [Dong et al., 2014], making the closed-world assumption is conceptually unreasonable. Conversely, it is also not feasible to include all possible tuples and their probabilities in the knowledge base. The resulting difficulty is illustrated by Figure 2: there are an enormous number of probabilistic facts that can be scraped from the internet, and by definition these tools will keep only those
with the very highest probability. This tells us that the knowledge base we are looking at is fundamentally incomplete. In response to this problem, Ceylan et al. [2016] propose the notion of a completion for a probabilistic database.

**Definition 3.** A λ-completion of a probabilistic database \( \mathcal{P} \) is another probabilistic database obtained as follows. For each atom \( t \) that does not appear in \( \mathcal{P} \), we add tuple \( \langle t : p \rangle \) to \( \mathcal{P} \) for some \( p \in [0, \lambda] \).

Then, we can define the open world of possible databases in terms of the set of distributions induced by all completions.

**Definition 4.** An open-world probabilistic database (OpenPDB) is a pair \( \mathcal{G} = (\mathcal{P}, \lambda) \), where \( \mathcal{P} \) is a probabilistic database and \( \lambda \in [0, 1] \). \( \mathcal{G} \) induces a set of probability distributions \( K_\mathcal{G} \) such that a distribution \( P \) belongs to \( K_\mathcal{G} \) iff \( P \) is induced by some \( \lambda \)-completion of probabilistic database \( \mathcal{P} \).

OpenPDBs specify a set of probability distributions rather than a single one, meaning that a given query produces a set of possible probabilities rather than a single one. We focus on computing the minimum and maximum possible probability values that can be achieved by completing the database.

**Definition 5.** The probability interval of a Boolean query \( Q \) in OpenPDB \( \mathcal{G} \) is \( K_\mathcal{G}(Q) = [P_\mathcal{G}(Q), \overline{P_\mathcal{G}(Q)}] \), where
\[
P_\mathcal{G}(Q) = \min_{P \in K_\mathcal{G}} P(Q) \quad \quad \overline{P_\mathcal{G}(Q)} = \max_{P \in K_\mathcal{G}} P(Q)
\]

### 2.5 Query Evaluation for OpenPDBs

In general, computing the probability interval for some Boolean \( Q \) is not tractable. As observed in Ceylan et al. [2016] however, for a monotone UCQ, one where either all atoms are negated or none are, the situation is different. For monotone UCQs, the upper and lower bounds are given respectively by the full completion (that is, assume all unknown probabilities are \( \lambda \)), and the closed world database. This is a direct result of the fact that OpenPDBs form a credal set, and for credal sets probability bounds always come from extreme points [Cozman, 2000]. Furthermore, Ceylan et al. [2016] also provide an algorithm for efficiently computing this upper bound corresponding to a full completion, and show that it works whenever the UCQ is safe.

This result and corresponding algorithm provide an appealing case for OpenPDBs. They give a more reasonable semantics for the situation, and for a large class of queries they come at no extra cost in comparison to traditional PDBs. However, in practice computing an upper bound in this way tends to give results very close to 1. Intuitively, this makes sense: our upper bound comes from simultaneously assuming that every possible missing atom has some reasonable probability. While such a bound is easy to compute, it does not correspond to a reasonable relaxation of the closed-world assumption.

### 3. Mean Constrained Completions

To achieve our goals of reducing the upper bound on OpenPDB queries as well as to create a more stringent relaxation of the closed-world assumption, we propose placing constraints
on the overall probability of some relation across the entire population. For example, while we may not know the probability of each person on earth being a scientist individually, we might know that 1% of the general population are scientists. This allows us to include more information at the schema level, without having more information about each individual.

To illustrate this, consider a schema in which we have 3 relations: \( LiLA(x) \) denoting whether one lives in Los Angeles, \( LiSpr(x) \) denoting whether one lives in Springfield, and \( S(x) \) denoting whether one is a scientist. Using a vocabulary of 500 people where each person is present in at most one relation, Table 1 shows the resulting upper probability bound under different model assumptions, where the constrained open-world restricts at most 50% of mass on \( LiLA \), 5% on \( S \), and 0.5% on \( LiSpr \). In particular, notice how extreme the difference is in upper bound with and without constraints being imposed. The closed-world probability of both of these queries is always 0, as each person in our database only has a known probability for at most one relation. It is clear that of these three options, the constrained open-world is the most reasonable – the rest of this section formalizes this idea and investigates the resulting properties.

### 3.1 Formalization

We begin here by defining mean based constraints, before examining some immediate observations about the structure of the resulting constrained database.

**Definition 6.** Suppose we have an OpenPDB \( G = (\mathcal{P}, \lambda) \), and that \( R \) is a relation in \( G \) with \( \text{Tup}(R) \subseteq \mathcal{P} \) being the set of tuples in relation \( R \). Then a mean tuple probability constraint (MTP constraint) \( \varphi \) is a linear constraint of the form

\[
\frac{1}{|\text{Tup}(R)|} \sum_{\{t:p\}\in\text{Tup}(R)} p \leq \bar{p} \quad \text{for some } \bar{p}.
\]

**Definition 7.** We say that a \( \lambda \)-completion is \( \varphi \)-constrained if the \( \lambda \)-completed database satisfies MTP \( \varphi \). If it satisfies all of \( \Phi = (\varphi_1, \varphi_2, \ldots, \varphi_n) \), then we say it is \( \Phi \)-constrained.

Since being \( \varphi \)-constrained is a property of completions of OpenPDBs rather than of the databases themselves, we are interested in the subset of completions which satisfy this property.

**Definition 8.** An OpenPDB \( G = (\mathcal{P}, \lambda) \) together with MTP constraints \( \Phi \) induces a set of probability distributions \( K_G^\Phi \), where distribution \( P \) belongs to \( K_G^\Phi \) iff \( P \) is induced by some \( \Phi \)-constrained \( \lambda \)-completion of \( \mathcal{P} \).

Once again, much like with standard OpenPDBs, for a Boolean query \( Q \) we are interested in computing bounds on \( P(Q) \).

<table>
<thead>
<tr>
<th>Query</th>
<th>Closed-World</th>
<th>Open-World</th>
<th>Constrained Open-World</th>
</tr>
</thead>
<tbody>
<tr>
<td>( LiLA(x), S(x) )</td>
<td>0</td>
<td>( 1 - 10^{-290} )</td>
<td>( 1 - 10^{-15} )</td>
</tr>
<tr>
<td>( LiSpr(x), S(x) )</td>
<td>0</td>
<td>( 1 - 10^{-191} )</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 1: Comparison of upper bounds for the same query and database with different model assumptions.
Definition 9. The probability interval of a Boolean query $Q$ in OpenPDB $G$ with MTP constraints $\Phi$ is $K_\Phi^G(Q) = [P_\Phi^G(Q), \overline{P_\Phi^G(Q)}]$, where

$$P_\Phi^G(Q) = \min_{P \in K_\Phi^G} P(Q) \quad \text{and} \quad \overline{P_\Phi^G(Q)} = \max_{P \in K_\Phi^G} P(Q)$$

3.2 Completion Properties

A necessary property of OpenPDBs for efficient computation of boundary probability is that they are credal - this is what allows us to consider only a finite subset of possible completions. MTP constrained OpenPDBs maintain this property.

Theorem 1. Suppose we have an OpenPDB $G$ together with MTP constraints $\Phi$. Then the induced set of probability distributions $K_\Phi^G$ is credal.

Proof. To prove this, we need to show that $K_\Phi^G$ is both closed and convex.

Due to the way our constraints are defined, we know that $K_\Phi^G = K_G \cap K^\Phi$, where $K^\Phi$ is the set of all completions satisfying $\Phi$ (but not necessarily having all tuple probabilities $\leq \lambda$). We already know that $K_G$ is credal, and thus closed and convex. $K^\Phi$ is a half-space, which we also know is closed and convex. The intersection of closed spaces is closed, and the intersection of convex spaces is convex, so $K_\Phi^G$ is credal. \(\square\)

This property allows us to examine only a finite subset of configurations when looking at potential completions, since the bounds of a credal set are always achieved at points of extrema [Cozman, 2000]. Next, we would like to characterize these points of extrema, by showing that the number of tuples not on their own individual boundaries (that is, 0 or $\lambda$) is given by the number of MTP constraints.

Theorem 2. Suppose we have an OpenPDB $G$ together with MTP constraints $\Phi$, and a UCQ $Q$. Then there exists $P'$ a $\Phi$-constrained $\lambda$-completion of $P$, and $T$ a subset of missing tuples such that $K_\Phi^G(Q) = [P_P(Q), P_{P'}(Q)]$ and

$$P' = P \cup \{\langle t : p \rangle | t \notin P \cup T, p \in \{0, \lambda\}\} \cup \{\langle t : p \rangle | t \in T, p \in [0, \lambda]\}$$

with $|T| \leq |\Phi|$. \(\text{(2)}\)

In other words, the upper bound is given by a completion which has at most $|\Phi|$ added tuples with probability not exactly 0 or $\lambda$. Intuitively, each MTP constraint contributes a single non-boundary tuple, which can be thought of as the “leftover” probability mass once the rest has been assigned in full.

Proof. Since $K_\Phi^G$ is credal, we are interested here in determining the point of extrema of $K_\Phi^G$, as this will tell us precisely which completions can represent boundaries.

Consider the construction of the set $K_\Phi^G$, and suppose that there are $d$ possible open-world tuples, meaning that $K_\Phi^G \subseteq \mathbb{R}^d$. As we observed in the proof of Theorem 1, $K_\Phi^G = K_G \cap K^\Phi$, where $K^\Phi$ is the set of all completions satisfying $\Phi$. We now make three key observations about these sets:
1. Each individual possible open-world tuple is described by the intersection of 2 half-spaces: that is, the tuple on dimension $i$ is described by $x_i \geq 0$ and $x_i \leq \lambda$. $K_G$ is the intersection of all $2d$ of these half-spaces.

2. For any individual open-world tuple, the boundaries of the two half-spaces that describe it can not intersect each other.

3. An MTP constraint is a linear constraint, and thus can be described by a single half-space. So $K^\Phi$ is described by the intersection of these $|\Phi|$ half-spaces.

Observations 1 and 3, together with Lemma 1 tells us that any point of extrema of $K^\Phi_G$ must be given by the intersection of the boundaries of at least $d$ of the half-spaces that form $K^\Phi_G$. Observation 3 tells us that at most $|\Phi|$ of these half-spaces come from MTP constraints, leaving the boundaries of at least $d - |\Phi|$ half-spaces which come from $K_G$. Finally, observation 2 tells us that each of these $d - |\Phi|$ half-spaces is describing a different open world tuple. But this means we must have at least $d - |\Phi|$ tuples which lie on the boundary of one of their defining half-spaces: they must be either 0 or $\lambda$.

Lemma 1. If $S \subseteq \mathbb{R}^n$ is a set formed by the intersection of $k < n$ half-spaces, $S$ has no points of extrema.

Proof. Written as a set of linear equalities, the solution clearly must have at least 1 degree of freedom. This indicates that for any potential extrema point $x$, one can move in either direction along this degree of freedom to construct an open line intersecting $x$, but entirely contained in $S$.

Finally, so that we can treat MTP query evaluation as a combinatorial optimization problem, we only consider the case where achieving the mean tuple probability exactly leaves us with every individual tuple at its boundary. To see that we can do this, we observe that Theorem 2 leaves a single tuple per MTP constraint not necessarily on the boundary. But this tuple can always be forced to be on the boundary by very slightly increasing the mean of the constraint.

Corollary 1. Suppose we have an OpenPDB $G$ together with MTP constraints $\Phi$ such that each relation in $G$ has at most 1 constraint, and that we have a UCQ $Q$. Suppose further that each constraint allows adding open-world probability mass exactly divisible by $\lambda$. Then there exists $P'$ a $\Phi$-constrained $\lambda$-completion of $P$, and $T$ a subset of missing tuples such that

$$P' = P \cup \{ \langle t \in \{0, \lambda\} \rangle | t \in T \}$$

(3)

with $K^\Phi_G(Q) = [P_P(Q), P_{P'}(Q)]$.

4. Exact MTP Query Evaluation

With Section 3 formalizing MTP constraints and showing that computing upper bounds subject to MTP constraints is a combinatorial problem of choosing which $\lambda$-probability tuples to add in the completion, we now investigate exact solutions. We will show that
efficient MTP query evaluation is not in one-to-one correspondence with efficient query evaluation for unconstrained queries. That is, MTP constraints fundamentally change the difficulty landscape of query evaluation.

To show this, we propose a representative hard query for evaluation subject to MTP constraints.

**Definition 10.** Let \( M_0 = R(x_1, y_1, z_1), U(x_1) \lor R(x_2, y_2, z_2), V(y_2) \lor R(x_3, y_3, z_3), W(z_3) \lor U(x_4), V(y_4) \lor U(x_5), W(z_5) \lor V(y_6), W(z_6). \)

A key observation here is that the query \( M_0 \) is both safe and monotone. That is, if we ignore constraints and evaluate it subject to the open-world semantics, computing the probability of the query would be polynomial in the size of the database. We now show that this is not the case subject to a single MTP constraint on \( R \).

**Theorem 3.** Evaluating \( \overline{P}(M_0) \) subject to an MTP constraint on \( R \) is NP-hard.

We will prove this by using \( M_0 \) with the correct constraint on \( R \) as an oracle to solve the 3-dimensional matching problem.

**Definition 11.** Let \( X, Y, Z \) be finite disjoint sets representing nodes, and let \( T \subseteq X \times Y \times Z \) be the set of available hyperedges. Then \( M \subseteq T \) is a matching if for any distinct triples \((x_1, y_1, z_1) \in M, (x_2, y_2, z_2) \in M\), we have that \( x_1 \neq x_2, y_1 \neq y_2, z_1 \neq z_2 \). The 3-dimensional matching decision problem is to determine for a given \( X, Y, Z, T \) and positive integer \( k \) if there exists a matching \( M \) with \( |M| \geq k \).

We now underline the intuitive correspondence between \( M_0 \) and the 3-dimensional matching problem. The set of available tuples for \( R \) will correspond to all edges in \( T \), and thus an MTP constraint forces a decision on which subset of \( T \) to take. However, if we simple queried to maximize \( R(x, y, z) \), this need not correspond to a matching. Instead, we have the conjunct \( R(x_1, y_1, z_1), U(x_1) \) which is maximized when each triple chosen from \( R \) has a different \( x \) value. Similar conjuncts for \( y \) and \( z \) ensure that the query is maximized when each triple uses a different \( y \) and \( z \) value. Putting all of these together ensures that the query probability is maximized when the subset of triples chosen to complete \( R \) form a matching.

Finally, the last part of the query \( U(x_4), V(y_4) \lor U(x_5), W(z_5) \lor V(y_6), W(z_6) \) ensures that inference on \( M_0 \) is tractable, but it is unaffected by the choice of triples in \( R \).

Before we present the formal proof, we state and prove 2 Lemmas we will need.

**Lemma 2.** Suppose we have two completions \( P_1 \) and \( P_2 \) of \( R \), which only differ on a single triple, that is \( P_1 = P_0 \cup \{x_1, y_1, z_1\} \) and \( P_2 = P_0 \cup \{x_2, y_2, z_2\} \). Further suppose that \( y_1 = y_2, z_1 = z_2 \), and that \( P_0 \) contains no triples with \( x \)-value \( x_1 \), but does contain at least 1 triple with \( x \)-value \( x_2 \). Then \( P_1(M_0) > P_2(M_0) \).

**Proof.** We will apply a similar technique here to the one used to prove Theorem 4, where we directly examine \( \Delta \), the logical formula found by grounding \( M_0 \). Since \( M_0 \) is a union of conjunctive queries, \( \Delta \) must be a DNF. Each conjunct either does not contain \( R \), in which case it does not vary with the choice of completion, or it contains it exactly once. Any conjunct containing an atom of \( R \) not assigned probability by a completion is logically false.
In order to prove that $P_1(M_0) > P_2(M_0)$, let us compare the ground atoms that result from each. It is clear that the only spot on which they differ is on conjuncts involving $R(x_1, y_1, z_1)$ or $R(x_2, y_1, z_1)$. Any conjuncts involving one of these and $V$ or $W$ will also have an identical effect on the probability of the query, since the completions are identical over $y$ and $z$.

Finally these means we need to compare the term $R(x_1, y_1, z_1), U(x_1)$ with the term $R(x_2, y_1, z_1), U(x_2)$. Observe that we know $P_0$ contains triples with $x$-value $x_2$, which means the term only contributes new probability mass when $U(x_2)$ is true and none of the other triples involving $x_2$ are true. However, $P_0$ does not contain any triples with $x$-value $x_1$, so the term $R(x_1, y_1, z_1), U(x_1)$ contributes the maximum probability possible. Thus, for any choice of probabilities on $U$ such that $U(x_2) < 1$, we have that $P_1(M_0) > P_2(M_0)$.

**Lemma 3.** The upper bound $\overline{P}(M_0)$ is maximized if and only if $P$ is a completion formed by a matching of size $k$, where $k$ is the maximum number of tuples with probability $\lambda$ that can be added to $R$ in $M_0$.

**Proof.** Observe that if we begin with a completion given by a matching, we can repeatedly apply Lemma 2 to arrive at any completion. Thus a completion given by a matching must have higher probability than any completion not given by a matching.

Finally, we are ready to present the proof of Theorem 3.

**Proof.** Suppose we are given an instance of a 3-dimensional matching problem $X, Y, Z, T$ and an integer $k$. Let $U(x), V(y), W(z)$ be 0.8 wherever $x \in X, y \in Y$, or $z \in Z$ respectively, and 0 everywhere else. Additionally, let $R(x, y, z)$ be unknown for any $(x, y, z) \in T$, and 0 otherwise. Finally, we place an MTP constraint on $R$ ensuring that at most $k$ tuples can be added, and let $\lambda = 0.8$. Then Lemma 3 tells us that $M_0$ evaluated on this database will be maximized if and only if the completion used corresponds to a matching of size $k$. We determine this probability $P_{\text{max}}$ using a standard probabilistic database query algorithm, and fixing $R$ to have entries 0.8 for some disjoint set of triples.

Finally, we use our oracle for MTP constrained query evaluation to check $\overline{P}(M_0)$ with the database we constructed from the matching problem. We compare the upper bound given by the oracle, and if it is equal to $P_{\text{max}}$, Lemma 3 tells us that a matching of size $k$ does exist. Similarly, if the upper bound given by the oracle is lower than $P_{\text{max}}$, Lemma 3 tells us a matching of size $k$ does not exist.

**5. Approximate MTP Query Evaluation**

With the discussion in Section 4 answering definitively that we will not be able to find a general purpose algorithm for evaluating MTP query bounds, even when restricted to safe queries, an approximation is the logical next step. We now restrict our discussion to situations where we constrain a single relation, and dig deeper into the properties of MTP constraints to show the structure is submodular. We then exploit this property to achieve efficient bounds with guarantees.
5.1 On the submodularity of adding tuples

Consider that Theorem 2 tells us that assigning probabilities subject to MTP constraints is a problem of making discrete choices of which tuples to add. This means that we can analyze query evaluation as a set function on adding tuples. We begin with a few relevant definitions.

**Definition 12.** Suppose that we have an OpenPDB $G$, with an MTP constraint $\varphi$ on a single relation $R$, and we let $O$ be the set of possible tuples we can add to $R$. Then the set query probability function $S_{P,Q} : 2^O \to [0, 1]$ is defined by

$$S_{P,Q}(X) = P_{P \cup \{(t, \lambda) \mid t \in X\}}(Q)$$

This function provides us a way to formally reason about the combinatorial properties of query probabilities with respect to adding tuples. Intuitively, it describes the probability of a query on a database as a function of which open tuples we have added. Observe that $S_{P,Q}(\emptyset)$ is the closed world probability of the query, while $S_{P,Q}(O)$ is the open world probability of the query.

Now, we would like to show that $S_{P,Q}$ is a submodular set function.

**Definition 13.** A submodular set function is a function $f : 2^\Omega \to \mathbb{R}$ such that for every $X \subseteq Y \subseteq \Omega$, and every $x \in \Omega \setminus Y$, we have that $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$.

**Theorem 4.** The set query probability function $S_{P,Q}$ is submodular for any tuple independent probabilistic database $P$ and UCQ query without self-joins $Q$.

**Proof.** Without directly computing probabilities, let us inspect $\Delta$, the logical formula we get by grounding $Q$. $Q$ is a union of conjunctive queries, and thus $\Delta$ is a very large disjunction of conjuncts. Each conjunct can contain our constrained relation $R$ at most once due to the query not having self-joins, and any one of these conjuncts containing an atom of $R$ not assigned any probability is logically false.

Next, to show that $S_{P,Q}$ is submodular, let $X \subseteq Y \subseteq O$, and let $x \in O \setminus Y$ be given. We assign names to the following subformulas of $\Delta$

- $\alpha$ ($\beta$) is the disjunction of all conjuncts of $\Delta$ which are not logically false due to missing $R_B$ tuples in $X$ ($Y$)
- $\gamma$ is the disjunction of all conjuncts of $\Delta$ containing the tuple $x$

Additionally, since $X \subseteq Y$, we also know that $\alpha \Rightarrow \beta$. Now, we make a few observations relating these quantities with our desired values for submodularity:

- $S_{P,Q}(X) = P(\alpha)$
- $S_{P,Q}(Y) = P(\beta)$
- $S_{P,Q}(X \cup \{x\}) = P(\alpha \lor \gamma)$
- $S_{P,Q}(Y \cup \{x\}) = P(\beta \lor \gamma)$
Finally, we have the following:

\[SP, Q(X \cup \{x\}) - SP, Q(X) = P(\alpha \lor \gamma) - P(\alpha)\]
\[= P(-\alpha \land \gamma)\]
\[\geq P(-\beta \land \gamma)\]
\[= P(\beta \lor \gamma) - P(\beta)\]
\[= SP, Q(Y \cup \{x\}) - SP, Q(Y)\]

This gives us the desired property of submodularity on adding tuples, so we can proceed with exploiting submodularity for an efficient approximation.

### 5.2 From Submodularity to Approximation

Given the knowledge that the probability of a safe query without self-joins is submodular in the completion of a single relation, we are now tasked with using this to construct an efficient approximation. Since we further know the probability is also monotone as we have restricted our language to monotone UCQs, Nemhauser et al. [1978] tells us that we can get a \(1 - \frac{1}{e}\) approximation using a simple greedy algorithm. The final requirement to achieve the approximation described in Nemhauser et al. [1978] is that our set function must have the property that \(f(\emptyset) = 0\). This can be achieved in a straightforward manner as follows.

**Definition 14.** Suppose that our setup is identical to that of Definition 12. Then the normalized set query probability function \(SP', Q : 2^O \rightarrow [0, 1]\) is defined by

\[SP', Q(X) = PP_{\cup\{t: \lambda\} | t \in X}(Q) - P_P(Q)\]

**Remark 1.** For any normalized set query probability function \(SP', Q\), we have \(SP', Q(\emptyset) = 0\).

By simply normalizing the set query probability function, we can now directly apply the greedy approximation described in Nemhauser et al. [1978]. In order to make this useful, we first need an efficient algorithm for computing the probability of a query subject to adding to our database any possible tuple from the constrained relation. This procedure is given by Algorithm 1.

Algorithm 1 is extremely similar in structure to the standard closed-world PDB query evaluation algorithm [Gribkoff et al., 2014, Dalvi and Suciu, 2012], as well the standard algorithm for evaluating upper bounds on OpenPDB queries [Ceylan et al., 2016]. The primary difference is that it constructs a list of probabilities rather than a single one, representing the open-world probability given that we added any possible single tuple to the constrained relation.

Finally, combining Algorithm 1 with the submodularity and monotonicity of adding a tuple leaves us with a straightforward greedy algorithm. We use Algorithm 1 to find the next best tuple to add based on the current database, and then add it. This is repeated until adding another tuple would violate the MTP constraint. Finally, we say that \(P_{\text{Greedy}}(Q)\) is the approximation given by this greedy algorithm and recall that the true upper bound is
Algorithm 1 LiftGreedy(Q, P, λ, D, R), abbreviated by L(Q, P)

Require: CNF Q, prob. tuples P, threshold λ, domain D, and relation R.
Ensure: The probability Pr(P∪{x}(Q) for each x ∈ O (i.e. tuples that can be added to R)

1:  Step 0 Base of Recursion
2:  if Q is a single ground atom t ∈ R then
3:     if ⟨t : p⟩ ∈ P then
4:         return [p]
5:     else
6:         return [λ]
7:  Step 1 Rewriting of Query
8:     Convert Q to union of CNFs: QUCNF = Q_1 ∨ ... ∨ Q_m
9:  Step 2 Decomposable Disjunction
10:    if m > 1 and QUCNF = Q_1 ∨ Q_2 then
11:       q_1 ← L(Q_1, P|Q_1) and q_2 ← L(Q_2, P|Q_2)
12:       Only one of the disjuncts can contain R, so only one will return non-singleton list
13:       return {1 − (1 − q_1) · (1 − q_2)|q_1, q_2 ∈ (q_1 × q_2)}
14:  Step 3 Inclusion-Exclusion
15:    if m > 1 but QUCNF has no independent Q_i then
16:       Some of these will be lists, some will be singletons. Those that are lists will be the same shape.
17:       return ∑_{s⊆m}(-1)^{|s|+1} · L(∧_{i∈s}Q_i, P|∧_{i∈s}Q_i)
18:  Step 4 Decomposable Conjunction
19:    if Q = Q_1 ∧ Q_2 where Q_1 ⊥ Q_2 then
20:       Only one of the conjuncts can contain R, so only one will return a non-singleton list
21:       q_1 ← L(Q_1, P|Q_1) and q_2 ← L(Q_2, P|Q_2)
22:       return {q_1 · q_2|q_1, q_2 ∈ (q_1 × q_2)}
23:  Step 5 Decomposable Universal Quantifier
24:    if Q has a separator variable x then
25:       let T be all constants as x-argument in P
26:       A ← [L(Q[x/t_i], P)|x=t_i] for x ∈ T
27:       B ← [D \ T] ∗ [L(Q[x/t], 0) for some t ∈ D \ T]
28:       return A :: B
29:  Step 6 Fail

We observe that P_{Greedy}(Q) ≤ P^ϕ_G(Q). Furthermore, Nemhauser et al. [1978] tells us the following:

P_{Greedy}(Q) − P_P(Q) ≥ (1 − \frac{1}{e})(P^ϕ_G(Q) − P_P(Q))

Combining these and multiplying through gives us the following bound:

P_{Greedy}(Q) ≤ P^ϕ_G(Q) ≤ \frac{e · P_{Greedy}(Q) − P_P(Q)}{e − 1}
It should be noted that it is possible for this upper bound to be above 1.

6. Discussion & Open Problems

This work proposes the novel problem of constraining open-world probabilistic databases at the schema level, without having any additional ground information over individuals. We introduced a formal mechanism for doing this, by limiting the mean tuple probability allowed in any given completion, and then sought to compute bounds subject to these constraints. Section 4 provided an example of a query for which computing bounds subject to a constraint is hard, showing that constrained query evaluation may not be tractable, even when the query can be computed efficiently without a constraint. Section 5 provided one possible path for an approximation algorithm, restricting the constraint to a single relation and showing that under these conditions the problem is submodular. We now detail some of the gaps in our work that we consider interesting open problems.

Dichotomy for Exact Query Evaluation Section 4 showed us that some queries are NP-hard to compute exactly, but we can also see analytically that some queries can be computed exactly in an efficient manner. For example, Section 3 uses the query $LiS(x), S(x)$ as a motivation for the use of constraints. It is clear here that with only a single constrained relation, the exact solution could be found simply applying the greedy technique proposed in Section 5.2. Can an exact characterization be given for when a query is hard and when it can be efficiently solved?

General Solution for Tractable Exact Queries In a similar vein, can a single algorithm be proposed that can efficiently compute the upper bound subject to a constraint, for all queries that admit such a tractable solution? Moreover, can this polynomial-time algorithm be made to have a low degree?

More Reliable Approximation Scheme Section 5 proposes an approximation scheme that, while provably within a constant factor of the true bound, can also give probabilities above 1. Is there an approximation scheme that reliably gives values close to the true upper bound, and is also guaranteed to give a bound less than 1?

More Efficient Approximation Scheme The approximation scheme described in Section 5 describes a polynomial time approximation scheme, where the degree of the polynomial is the arity of the relation being constrained. However, Corollary 14 of Ceylan et al. [2016] tells us that open-world query evaluation is in fact linear time. Is it possible to do this same approximation in linear time? If not in general, then for which class of queries is it possible?

References


