The Most Probable Database Problem

Eric Gribkoff
University of Washington
eagribko@cs.uw.edu

Guy Van den Broeck
KU Leuven, UCLA
guyvdb@cs.ucla.edu

Dan Suciu
University of Washington
suciu@cs.uw.edu

ABSTRACT

This paper proposes a novel inference task for probabilistic databases: the most probable database (MPD) problem. The MPD is the most probable deterministic database where a given query or constraint is true. We highlight two distinctive applications, in database repair of key and dependency constraints, and in finding most probable explanations in statistical relational learning. The MPD problem raises new theoretical questions, such as the possibility of a dichotomy theorem for MPD, classifying queries as being either PTIME or NP-hard. We show that such a dichotomy would diverge from dichotomies for other inference tasks. We then prove a dichotomy for queries that represent unary functional dependency constraints. Finally, we discuss symmetric probabilities and the opportunities for lifted inference.

Keywords

probabilistic databases, statistical relational learning, probabilistic inference, complexity

1. INTRODUCTION

Probabilistic databases are motivated by the need to store large-scale uncertain data, and query it efficiently [34]. A tuple-independent probabilistic database *PDB* associates a probability with each tuple, and each tuple represents an independent random variable. Every probabilistic database thus induces a probability distribution $\Pr_{PDB}(.)$ over deterministic databases. Typical database queries involve computing certain marginal probabilities of this distribution. In this paper, we consider an entirely different task. Given a probabilistic database *PDB*, and a logical constraint or query Q, the *most probable database* (MPD) is the deterministic database DB that satisfies Q and maximizes $\Pr_{PDB}(DB)$.

The MPD problem is related to the *most probable explanation* (MPE) task in probabilistic graphical models [12, 24] and statistical relational learning [18]. It is perhaps the most prominent reasoning task used in practical applications, for solving prediction problems. Examples include image segmentation with probabilistic graphical models and collective classification with Markov logic [31]. We show that MPE problems can be reduced to the MPD task, and illustrate this on an example in Markov logic. Moreover, we show that data repair and cleaning problems [5, 16] on probabilistic databases are natural instances of the general MPD task. In particular, we highlight applications to key and functional dependency repair.

The main contribution of this paper is a series of computational complexity results for the MPD task. We state the problem of a *dichotomy* for classes of MPD problems, and whether there could exist an algorithm that classifies MPD queries as being PTIME or NP-hard in the size of the database. We seek to understand the relative complexity of the MPD task, compared to traditional probabilistic query processing (marginal probabilities), and the Boolean satisfiability task. We show that a dichotomy for MPD would need to be different in the queries it classifies as tractable and intractable. We further show a first MPD dichotomy on the class of functional dependency constraints between single attributes. This dichotomy generalizes complexity results for finding minimal repairs [23, 7, 10], and upgrades these to the probabilistic setting. Finally, we briefly discuss MPD with symmetric probabilities, which make the problem more tractable, by allowing more opportunities for lifted inference [30].

2. PROBLEM STATEMENT

This section introduces the MPD task and its variations. We begin with some necessary background and notation.

2.1 Background and Notation

Throughout this paper, we will work with the relational fragment of first-order logic, which we now briefly review. An *atom* $P(t_1, \ldots, t_n)$ consists of predicate P/n of arity n followed by n arguments, which are either *constants* or *logical variables* $\{x, y, \ldots\}$. A vector of logical variables is denoted $\bar{x}$ or $\bar{y}$. A *literal* is an atom or its negation. A *formula* combines atoms with logical connectives and quantifiers $\exists$ and $\forall$. A *clause* is a universally quantified disjunction of literals. A *unit clause* has length one. A *CNF* is a conjunction of clauses. Semantics are defined in the usual way [26]. An interpretation, or database, DB that satisfies sentence Q is denoted $DB \models Q$. We also assume familiarity with basic notions of computational complexity [4], including the complexity classes PTIME, NP and #P.

2.2 Definition: Most Probable Database

In proposing the most probable database problem, we are inspired by the most probable explanation (MPE) inference task that is well-known in the probabilistic graphical model literature [12, 24], where it is also called maximum a posteriori (MAP) inference. Take for example a Bayesian network $\mathcal{B}$, modeling a distribution $\Pr_{\mathcal{B}}(\mathbf{X})$ over random variables $\mathbf{X}$. For any assignment $\mathbf{e}$ to variables $\mathbf{E} \subseteq \mathbf{X}$, called evidence, the MPE is

$$\text{MPE}(\mathbf{e}) = \arg\max_{\mathbf{y}} \ \Pr_{\mathcal{B}}(\mathbf{y}|\mathbf{e}) = \arg\max_{\mathbf{y}} \ \Pr_{\mathcal{B}}(\mathbf{y}, \mathbf{e}),$$

where $\mathbf{y}$ is an assignment to the variables $\mathbf{Y} = \mathbf{X} \setminus \mathbf{E}$.

The MPE problem does not directly translate to tuple-independent probabilistic databases in a meaningful way. A probabilistic database *PDB* also induces a distribution $\Pr_{PDB}(.)$ over deterministic databases. However, this distribution completely factorizes, which trivializes the above equation: we can simply optimize all tuples independently. The MPE task has therefore received little attention for probabilistic database querying, as also observed by [35].

To make MPE meaningful, one has to either extend the probabilistic database model with complex correlations [21, 17], or generalize the notion of evidence to induce correlations. This is precisely what the following achieves.

Definition 1. Suppose we have a probabilistic database *PDB* representing the distribution $\Pr_{PDB}(.)$. Given a sentence Q in relational logic, called the query or constraints, the *most probable database* (MPD) is defined as

$$\text{MPD}(Q) = \arg\max_{DB \models Q} \ \Pr_{PDB}(DB).$$

Correlations are now induced by the complex logical structure of Q. To connect to the probabilistic graphical model world, it may thus be fruitful to think of Q as representing a probability distribution $\Pr_Q(.)$, to think of the probabilistic database *PDB* as soft evidence [12], and of the MPD task as finding $\arg\max_{DB} \Pr_Q(DB|PDB)$. This switches the roles of the inputs to the MPE task, but is entirely equivalent.

Another benefit of introducing the MPD task as a first-class citizen is that it clearly separates the structure Q of the reasoning task from its parametrization and domain. That permits us to separately analyze query and data complexity, which will help us in Section 4 to show theoretical properties.

2.3 Special Cases

There are three notable special cases of MPD, depending on the assumptions about the probabilistic database.

Asymmetric MPD This is the most general setting, where the probabilistic database can contain arbitrary tuple probabilities. It is the focus of this paper.

Symmetric MPD This is the special case where each table contains all possible tuples, and the probability of all tuples in the same table is identical. While somewhat contrived, we will see that this is the type of MPD problem obtained when translating statistical relational models to probabilistic databases. It is therefore of independent theoretical interest.

Semi-Symmetric MPD This setting is obtained when we translate MPE problems on statistical relational models to MPD. The database consists of only deterministic tables with 0/1 probabilities, and entirely symmetric probabilistic tables, where all tuples have identical probabilities, and no tuples are omitted. It has many applications in statistical relational learning [18].

3. APPLICATIONS

We now highlight two applications of MPD, one in probabilistic databases, and one in statistical relational learning.

3.1 Probabilistic Database Repair

In many settings, data must be incorporated from numerous sources, potentially leading to violations of key or other data constraints. The database community has taken two different approaches to violations of constraints. The practical approach is to do data cleaning: fix the database until it satisfies the constraint [5, 16]. A different approach was taken by the theory community [3], which has advocated keeping the constraint violations in the data and instead computing query answers that are present in every repair. While several notions of "repairs" have been proposed in the literature, they usually tend to be defined as maximal subsets of the database that satisfy the constraints.

As the first application of MPD, we propose to perform *probabilistic database repair*. Given a probabilistic database that expresses the confidence we have in the correctness of each tuple, we can compute the most probable database that adheres to the data constraints Q. MPD thus provides a principled probabilistic framework for repair.

Next, we illustrate this approach by showing examples of individual data constraints, and how they are represented in a logical sentence Q. When multiple such constraints are involved in a data repair task, Q is simply a conjunction of individual constraints.

Key The most basic constraint type is the *key constraint*. If the first attribute is a key in table R, then Q is

$$\forall x, \bar{y}, \bar{y}', \quad R(x, \bar{y}) \wedge R(x, \bar{y}') \Rightarrow (\bar{y} = \bar{y}').$$

Functional Dependency For the *functional dependency* repair task on relation R, when $x \mapsto y$ is a single functional dependency from the first attribute to the second, Q has the form

$$\forall x, y, y', \bar{z}, \bar{z}', \quad R(x, y, \bar{z}) \wedge R(x, y', \bar{z}') \Rightarrow (y = y').$$

More generally, functional dependencies exist between sets of attributes. Conditional functional dependencies replace some logical variables above by constants. The existence of a functional dependency depends on the values of certain attributes [6].

Inclusion Dependency To encode an *inclusion dependency*, Q has the form

$$\forall \bar{x}, \bar{y}, [R(\bar{x}, \bar{y}) \Rightarrow \exists \bar{z}, S(\bar{x}, \bar{z})] \,.$$

Generalized Constraint We can represent expressive *generalized constraints* [1], where Q has the form

$$\forall \bar{x}, [\exists \bar{y}, \phi(\bar{x}, \bar{y}) \Rightarrow \exists \bar{z}, \psi(\bar{x}, \bar{z})]$$

and ϕ and ψ are themselves logical formulas.

3.2 Statistical Relational MPE

As our second application, we show how MPE inference in statistical relational models [18] can be reduced to MPD inference. Several such models, including parfactor graphs [30], Markov logic networks (MLNs) [31] and probabilistic logic programs [13], can be reduced to a weighted first-order model counting representation [38, 19, 37], whose weights can subsequently be normalized to obtain a probabilistic database and query. We refer to the literature for the details, and here show the process for the following example MLN.

$$2 \quad \texttt{Prof}(x) \wedge \texttt{Advises}(x, y) \Rightarrow \texttt{Student}(y)$$

It states that the probability of a world increases by a factor e^2 with every pair of people x, y for which the formula holds.

Our reduction to MPD consists of a formula Q equal to

$$\forall x, \forall y, \texttt{F}(x, y) \Leftrightarrow [\texttt{Prof}(x) \wedge \texttt{Advises}(x, y) \Rightarrow \texttt{Student}(y)]$$

and probabilistic database *PDB* that assigns probabilities $e^2/(1+e^2)$ to all tuples in table `F`, and probability 0.5 to all tuples in tables `Prof`, `Advises`, and `Student`. The solution to this symmetric MPD problem is the most probable state of the MLN model.

We have so far assumed that the MPE evidence e is empty. In practice, however, e typically consists of a full deterministic database for a subset of the tables. For example, when classifying people as students or professors, we would be given the full `Advises` table to base our predictions on. This evidence is set by replacing the symmetric `Advises` table in our MPD encodings by this deterministic table, thus obtaining a semi-symmetric MPD problem. The solution of that problem is the MPE state for the given evidence.

A General Algorithm for MPD.

There exists a reduction from MPD to MPE that – in theory – can be used to solve MPD problems using existing algorithms. For any MPD problem, we can construct a Markov logic network whose MPE state is the MPD. The structure of this MLN is a function of both the probabilistic database and the query. It contains a set of hard clauses that represent Q. Such clauses have infinite weight and must always be satisfied in a possible world. For every database tuple t with probability p, the MLN additionally contains the unit clause t with weight $\log(p/(1-p))$. Several algorithms have been proposed to (approximately) solve MPE problems, using local search [22, 31] and cutting plane inference [32, 29]. Even though these algorithms have become very efficient, many of them were not designed to operate at the scale of probabilistic databases and handle the large number of unit clauses in the reduction sketched above.

4. COMPLEXITY QUESTIONS

The introduction of the MPD problem and its variants raises several new complexity questions, which we will now investigate. We pose the problem of a dichotomy for MPD, and provide some evidence that such a dichotomy would be novel and interesting. We then focus on a small but non-trivial class of functional dependency MPD problems and prove a dichotomy. Finally, we discuss complexity questions related to (semi-)symmetric MPD.

4.1 The Quest for an MPD Dichotomy

The typical inference task in probabilistic databases is to compute the probability of a query. A sharp dichotomy theorem exists for this task [11], stating that the probability of any union of conjunctive (UCQ) queries (corresponding to a monotone DNF constraint) can either be computed in time polynomial in the size of the database, or is #P-hard. Moreover, [11] present an algorithm that efficiently computes PTIME queries and reports failure on #P-hard queries.

A compelling open complexity question is whether there exists *a dichotomy for the MPD task* on an large class of queries; for example UCQ queries. Such a dichotomy would state that all MPD queries are either NP-hard or PTIME in the size of the database. A second open problem is to find an *algorithm* that can perfectly classify queries as such.

One may wonder why we pose the dichotomy for MPD as a new problem, and if we have any reason to expect the MPD dichotomy to be syntactically different from existing dichotomies. We will now show evidence that a dichotomy for MPD would be different from the dichotomy for computing query probabilities [11]. We also show evidence that it would be different from a dichotomy for the satisfiability task (cf. Schaefer's dichotomy theorem [33]).

4.1.1 A Separation from Query Probability

Let us consider the following query, encoding a bidirectional functional dependency, with both $x \mapsto y$ and $y \mapsto x$.

$$Q_{match} = \begin{bmatrix} \forall x, y\ , y', & R(x, y) \wedge R(x\ , y') \Rightarrow (y = y') \\ \forall x, x', y\ , & R(x, y) \wedge R(x', y\ ) \Rightarrow (x = x') \end{bmatrix}$$

This query represents a *bipartite matching*, that is, it allows for edges $R(x, y)$ to exist between x and y, but no two edges can share the same node x or y. We now have the following.

THEOREM 1. *The MPD for Q_{match} can be computed in time polynomial in the size of the database, yet computing the probability of Q_{match} is #P-hard.*

PROOF. We will show that computing the probability of Q_{match} is #P-hard by reduction from counting the number of (imperfect) matchings in a bipartite graph, which is #P-complete [36]. For a given bipartite graph G with m edges, set the probability of $R(x, y)$ to 0.5 when G has an edge between x and y, and set it to 0 otherwise. Each possible world now has probability 0.5^m when it encodes a subgraph of G, and probability 0 otherwise. All words that satisfy Q encode a matching of G. Hence, we can compute the number of bipartite matchings for G as $2^m \cdot \Pr(Q_{match})$, and computing $\Pr(Q_{match})$ is #P-hard.

We will show that computing the MPD for query Q_{match} is tractable with any probabilistic database. The proof is by reduction to the maximum weighted bipartite matching problem, which is also known as the assignment problem, and can be solved in polynomial time using the Hungarian algorithm [25]. Given an arbitrary probabilistic database, observe that we can set all probabilities less than 0.5 to 0 without changing the MPD for Q_{match}. In their most likely state, these atoms are false, and they do not make Q_{match} unsatisfied. Observe also that optimizing the probability of a database is equivalent to optimizing its log-probability. Each tuple with associated probability p adds $\log(p/(1-p))$ to the log-probability of a database. We can maximize the log-probability by constructing a weighted bipartite graph as

follows. Add an edge between x and y iff the probabilistic database has tuple $R(x, y)$ with probability $p_{xy} > 0.5$. The weight associated with each edge is $\log(p_{xy}/(1 - p_{xy}))$. The maximum weighted matching of this graph has precisely one edge for every tuple in the MPD. The weight of that matching is the log-probability of the MPD. □

4.1.2 A Separation from Query Satisfiability

Horn clauses have at most one positive literal. For example, let us consider the following two Horn clauses.

$$Q_{horn} = \begin{bmatrix} \forall x, y, & \neg P(x) \vee \neg P(y) \vee \neg R_1(x, y) \\ \forall x, y, & \neg P(x) \vee \quad P(y) \vee \neg R_2(x, y) \end{bmatrix}$$

This CNF gives the following separation.

THEOREM 2. *Computing the MPD for Q_{horn} is NP-hard in the size of the database, yet deciding its satisfiability (i.e., whether $\Pr(\text{MPD}(Q_{horn})) > 0$) is done in polynomial time.*

PROOF. We show that satisfiability of Q_{horn} can be decided in polynomial time by reduction to Horn clause satisfiability, which can be checked in polynomial time [15]. The theory in this reduction consists of Q_{horn} and unit clauses for the database tuples with probability zero and one.

We show that computing the MPD of Q_{horn} is NP-hard by reduction from the MAX-SAT problem on quadratic Horn clauses (at most two literals per clause), which is NP-complete [20]. Quadratic Horn clauses all have the form $(\neg x \vee \neg y)$, $(\neg x \vee y)$, x, or $\neg x$. The reduction sets the probability of $P(x)$ to 0.9 when there is a clause x, to 0.1 when there is a clause $\neg x$, and to 0.5 otherwise. It sets the probability of $R_1(x, y)$ to 0.9 when there is a clause $(\neg x \vee \neg y)$, to 0 otherwise. Set the probability of $R_2(x, y)$ to 0.9 when there is a clause $(\neg x \vee y)$ and to 0 otherwise. Let k be the number of tuples with probability 0.5, and n the number of MAX-SAT clauses. The MAX-SAT problem now has a satisfiable set of m clauses iff there is a database with probability $0.5^k \cdot 0.9^m \cdot 0.1^{n-m}$ that satisfies Q_{horn}. The largest possible m maps to the probability of the MPD for Q_{horn}, which is therefore NP-hard to compute. □

4.2 Unary Functional Dependency Dichotomy

We prove a dichotomy for unary functional dependencies on a single relation R. These have the form $x \mapsto \bar{y}$, where x is a single attribute of R and $\bar{y}$ is a set of attributes. Our dichotomy results are summarized in Table 1.

4.2.1 The Tractable Cases

Theorem 1 shows that the MPD for two FDs $x \mapsto y, y \mapsto x$ is computable in PTIME. The following theorem establishes the second tractable case:

THEOREM 3. *The MPD of a key or single functional dependency repair constraint can be computed in time polynomial in the size of the database.*

PROOF. Remove all tuples with probability less than 0.5. For the key repair MPD task, group tuples by their key value. The MPD task then reduces to selecting the most probable tuple in each group. For the FD $\bar{x} \mapsto \bar{y}$ MPD task, group tuples by their value for $\bar{x}$ and select within each group the most probable set of tuples with identical values for $\bar{y}$. □

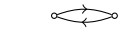
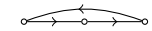
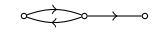
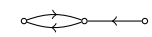
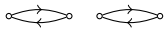
Dependencies	Complexity	Theorem
	PTIME	3
	PTIME	3
	PTIME	1
	NP-hard	4
	NP-hard	4
	NP-hard	4
	NP-hard	5
	NP-hard	6
	NP-hard	6
	NP-hard	7

Table 1: Individual complexity results for unary functional dependencies, represented by arrows.

4.2.2 The Intractable Base Cases

We prove that computing the most probable database with functional dependencies $x \mapsto z, y \mapsto z$ is NP-hard by reduction from the MAX-2-SAT problem. The same construction, with slight modifications, suffices to show NP-hardness for $x \mapsto y \mapsto z$ and $x \mapsto y, z \mapsto w$.

THEOREM 4. *Let $Q_n =$*

$$\begin{bmatrix} \forall x, y, y', z, z' & R(x, y, z) \wedge R(x, y', z') \Rightarrow (z = z') \\ \forall x, x', y, z, z' & R(x, y, z) \wedge R(x', y, z') \Rightarrow (z = z') \end{bmatrix}$$

Then computing the MPD for Q_n is NP-hard in the size of the database.

PROOF. Given a set $V = \{v_1, \ldots, v_k\}$ of Boolean variables, let Φ be a formula over V in conjunctive normal form with two variables per clause (2-CNF). $\Phi = c_1 \wedge c_2 \cdots \wedge c_m$, where $c_i = \ell_{v_{i_1}} \vee \ell_{v_{i_2}}$ and $\ell_v = v$ or $\bar{v}$, depending on whether v appears positive or negated. The NP-hard MAX-2-SAT problem is to determine the maximum number of clauses of Φ which can be simultaneously satisfied.

Construct a database with a single relation $R(x, y, z)$. For each clause $c_i = \ell_{v_{i_1}} \vee \ell_{v_{i_2}}$ in Φ, add the tuples $(i, v_{i_1}, \ell_{v_{i_1}})$ and $(i, v_{i_2}, \ell_{v_{i_2}})$ to R. Set the probability of all such tuples to $p > 0.5$.

Finding the MPD DB for this database consists of choosing a world containing the maximum number of tuples. The FD $x \mapsto z$, encoded by Q_n, ensures that at most one tuple per clause of Φ is included in DB. The FD $y \mapsto z$ ensures that every variable v_i in position y either maps to literal ℓ_{v_i} or $\ell_{\bar{v}_i}$ but not both.

Together, these imply that the MPD DB corresponds to an assignment $\theta : V \to \{0, 1\}^k$ satisfying at least $n = |DB|$ clauses. Suppose there exists an assignment θ' such that more than n clauses are satisfied. Then θ' gives us a set of more than n tuples that can be included in a world DB' such that $|DB| > n$ and $DB \not\models Q_n$, contradicting the assumption that DB was an MPD. Thus, the number of tuples in DB is equal to the MAX-2-SAT solution for Φ. It follows that computing the MPD DB with the functional dependencies $x \mapsto z$ and $y \mapsto z$ is NP-hard. □

The same encoding works for the functional dependencies $x \mapsto y \mapsto z$. To encode $x \mapsto y, z \mapsto w$, note that we can encode in the database the equivalence $y = z$, and the NP-hardness result follows by reduction from $x \mapsto y \mapsto z$.

THEOREM 5. *Computing the MPD with the functional dependencies $x \mapsto y$ and $y \mapsto z$ and $z \mapsto x$ is NP-hard in the size of the database.*

PROOF. *(Sketch)* By reduction from the 3-dimensional matching problem. Turn every hyperedge (x, y, z) into a tuple (a_x, b_y, c_z) with probability > 0.5. No two tuples can be chosen that have either x, y, or z in common, and the most probable database is the largest 3-dimensional matching. □

THEOREM 6. *Computing the MPD for the functional dependencies (a) $x \mapsto y, y \mapsto x, z \mapsto x$ or (b) $x \mapsto y, y \mapsto x, x \mapsto z$ is NP-hard in the size of the database.*

PROOF. *(Sketch)* By reduction from MAX-2-SAT, similarly to Theorem 4. For (a), z is a greatest attribute (ordered by implication). Let z encode the clauses c_i, x encode the literals and y encode the variables. Then each variable must map to one literal, and each clause can be made true only once. For (b), z is a least attribute, and we let z encode the literals, and x, y encode clauses and variables. □

4.2.3 The Full Dichotomy

We prove that computing the MPD for any set Σ of unary functional dependencies is either in PTIME or NP-hard.

Given a relation R and a set of unary functional dependencies Σ, a *strongly connected component* (SCC) is a maximal subset S of the attributes of R such that $\forall a \in S, a \mapsto S$. For two SCCs S and S', the functional dependency $S \mapsto S'$ defines an order relation $S > S'$. An SCC Y is *between* two SCCs X, Z if $X > Y > Z$ or $Z > Y > X$ holds.

Our result shows that the tractable sets of unary FDs are exactly those described in Section 4.2.1. The proof of the dichotomy relies on the following lemma.

LEMMA 1. *Let $\mathbb{S}$ denote the set of SCCs of relation R and unary functional dependencies Σ. For $Z \subseteq \mathbb{S}$ s.t $\forall P, Q \in Z$, there does not exist $P' \in \mathbb{S}$ s.t. $P > P' > Q$, let $R[Z]$ denote the projection of R to attributes in Z and $\Sigma[Z]$ denote the restriction of Σ to functional dependencies over the attributes of Z. If computing the MPD over $R[Z], \Sigma[Z]$ is NP-hard, then computing the MPD over R, Σ is also NP-hard.*

PROOF. Let $A = \{s \in \mathbb{S} | s > Z\}$, $B = \{s \in \mathbb{S} | s \not> Z\}$.

Let every tuple in R have a distinct value on each attribute in A. Thus, any FD $a \mapsto a'$ is satisfied for any set of tuples in R. Likewise, let every tuple in R have the same value on each attribute in B. The result of this construction is that every FD, save for those entirely over the attributes of Z, is satisfied for any set of tuples from R. This shows that computing the MPD over R, Σ in PTIME implies the ability to compute the MPD over $R[Z], \Sigma[Z]$ in PTIME, from which the claim follows. □

THEOREM 7. *Given a relation R and a set of unary functional dependencies Σ, computing the MPD is either PTIME or NP-hard.*

PROOF. The proof is a case analysis of the SCCs in R, Σ, making using of Lemma 1 and the above tractable and intractable sets of FDs.

Case 1: There exists an SCC Z with at least three attributes. Then computing the MPD is NP-hard, by Lemma 1 and the reduction of Theorem 5.

Case 2: There exists at least two SCCs with two attributes, Z and Z'. Then computing the MPD is NP-hard: equate one attribute of Z with an attribute of Z' (this is imposed at the database level). Then Z and Z' form an SCC and we proceed as for Case 1.

Case 3: There exists an SCC Z with two attributes and another FD, which necessarily goes from one SCC P to another SCC Q. One of the following must hold: (a) $Z = P$ and $Z > Q$. Choose Q s.t. there does not exist an SCC between Z and Q. (b) $Z = Q$ and $P > Z$. Choose P s.t. there does not exist an SCC between P and Z. (c) $Z \neq P, Z \neq Q$. Choose P, Q s.t. there does not exist an SCC between P, Q and equate P and Z. In (a)-(c), Lemma 1 applies and, by reduction from Theorem 6, computing the MPD is NP-hard.

Case 4: There exists an SCC Z with two attributes and no other FDs. Then computing the MPD is in PTIME by Theorem 1.

Case 5: There exists at least two FDs beginning at distinct SCCs Z, Z'. Then computing the MPD is NP-hard, as we can choose Z, Z' s.t. Lemma 1 and the reduction of Theorem 4 apply.

Case 6: There exists a single attribute SCC Z s.t. every FD is of the form $Z \mapsto P$, for any number of SCCs P. Then computing the MPD is in PTIME by Theorem 3. □

Note that our hardness results hold even in the deterministic setting, where the problem is to find a maximum size consistent repair. Some (not all) of the intractable cases of Section 4 were previously identified [23, 7, 10], but these works focused only on showing the existence of at least one hard set of FDs. To the best of our knowledge, Theorem 7 represents the first complete characterization of the complexity of repairing sets of unary functional dependencies.

4.3 Symmetric MPD

Our analysis so far has exclusively considered the asymmetric MPD problem. However, it is well-known in the lifted inference literature that exploiting symmetries can make otherwise intractable inference problems efficient to compute [30, 38, 19, 28]. For example, symmetry arguments have been used to show that MPE queries can be evaluated in time polynomial in the domain and evidence size for monadic MLNs [28], where all predicates have arity one. Semi-symmetric MPD is therefore also PTIME when Q represents such a monadic MLN, that is, when it has the form $F(\bar{x}) \Leftrightarrow \phi(\bar{x})$, where ϕ is a formula in monadic logic. It is an open question whether there are MPD queries that are tractable in the symmetric or semi-symmetric setting, but intractable in the asymmetric setting.

Finally, we note that the reduction from MPD to MPE that was proposed in Section 3.2 converts a symmetric MPD problem into an MLN with symmetries. Several lifted MPE algorithms have been proposed that exploit these symmetries [14, 2, 9, 27], and even support limited asymmetry [8]. They are known to give exponential speedups, but a precise characterization of their complexity is an open problem.

5. CONCLUSIONS

We have introduced the most probable database problem, and shown its applications in probabilistic data repair and

statistical relational learning. We briefly showed a general algorithm for MPD, by reduction to MPE in Markov logic. As our main contribution, we investigated the complexity of MPD. We showed a dichotomy for unary functional dependencies, classifying them as PTIME or NP-hard in the size of the database, as well as evidence that a more general dichotomy would be different for the MPD, query probability, and satisfiability tasks.

6. ACKNOWLEDGMENTS

This work was partially supported by ONR grant #N00014-12-1-0423, NSF grants IIS-1115188 and IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen).

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

[2] U. Apsel and R. I. Brafman. Exploiting uniform assignments in first-order mpe. *Proceedings of UAI*, 2012.

[3] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 68–79. ACM, 1999.

[4] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[5] L. Bertossi. Database repairing and consistent query answering. *Synthesis Lectures on Data Management*, 3(5):1–121, 2011.

[6] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755, 2007.

[7] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD Conference*, pages 143–154, 2005.

[8] H. B. Bui, T. N. Huynh, and R. de Salvo Braz. Exact lifted inference with distinct soft evidence on every object. In *AAAI*, 2012.

[9] H. H. Bui, T. N. Huynh, and S. Riedel. Automorphism groups of graphical models and lifted variational inference. *arXiv preprint arXiv:1207.4814*, 2012.

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

[11] N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *Journal of the ACM (JACM)*, 59(6):30, 2012.

[12] A. Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

[13] L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors. *Probabilistic inductive logic programming: theory and applications*. Springer-Verlag, Berlin, Heidelberg, 2008.

[14] R. de Salvo Braz, E. Amir, and D. Roth. Mpe and partial inversion in lifted probabilistic variable elimination. In *AAAI*, volume 6, pages 1123–1130, 2006.

[15] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.

[16] W. Fan and F. Geerts. Foundations of data quality management. *Synthesis Lectures on Data Management*, 4(5):1–217, 2012.

[17] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 2013.

[18] L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[19] V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of UAI*, pages 256–265, 2011.

[20] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters*, 26(1):1–4, 1987.

[21] A. Jha and D. Suciu. Probabilistic databases with markoviews. *Proceedings of the VLDB Endowment*, 5(11):1160–1171, 2012.

[22] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. *The Satisfiability Problem: Theory and Applications*, 17:573–586, 1997.

[23] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, pages 53–62, 2009.

[24] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[26] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[27] M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *Proceedings of AISTATS*, pages 788–797, 2012.

[28] M. Niepert and G. Van den Broeck. Tractability through exchangeability: A new perspective on efficient probabilistic inference. *arXiv preprint arXiv:1401.1247*, 2014.

[29] J. Noessner, M. Niepert, and H. Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2013.

[30] D. Poole. First-order probabilistic inference. In *IJCAI*, volume 3, pages 985–991. Citeseer, 2003.

[31] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[32] S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. *Proceedings of UAI*, 2008.

[33] T. J. Schaefer. The complexity of satisfiability problems. In *STOC*, volume 78, pages 216–226, 1978.

[34] D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[35] M. Theobald, L. De Raedt, M. Dylla, A. Kimmig, and

I. Miliaraki. 10 years of probabilistic querying–what next? In *Advances in Databases and Information Systems*, pages 1–13. Springer, 2013.

[36] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[37] G. Van den Broeck, W. Meert, and A. Darwiche. Skolemization for weighted first-order model counting. In *Proceedings of KR*, 2014.

[38] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of IJCAI*, pages 2178–2185, 2011.