SlimShot: Probabilistic Inference for Web-Scale Knowledge Bases

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

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

ABSTRACT

Increasingly large Knowledge Bases are being created, by crawling the Web or other corpora of documents, and by extracting facts and relations using machine learning techniques. To manage the uncertainty in the data, these KBs rely on probabilistic engines based on Markov Logic Networks (MLN), for which probabilistic inference remains a major challenge. Today's state of the art systems use variants of MCMC, which have no theoretical error guarantees, and, as we show, suffer from poor performance in practice.

In this paper we describe SlimShot (Scalable Lifted Inference and Monte Carlo Sampling Hybrid Optimization Technique), a probabilistic inference engine for Web-Scale knowledge bases. SlimShot converts the MLN to a tuple-independent probabilistic database, then uses a simple Monte Carlo-based inference, with three key enhancements: (1) it combines sampling with safe query evaluation, (2) it estimates a conditional probability by jointly computing the numerator and denominator, and (3) it adjusts the proposal distribution based on the sample cardinality. In combination, these three techniques allow us to give formal error guarantees, and we demonstrate empirically that SlimShot outperforms today's state of the art probabilistic inference engines used in knowledge bases.

1. INTRODUCTION

Increasingly sophisticated information extraction and AI techniques have enabled the automatic construction of large knowledge bases, created by crawling the web and extracting facts and relations using machine learning techniques [15, 26, 13, 24, 6]. While conventional fact bases such as YAGO [19] and Freebase [16] contain high-quality, human-curated facts that are guaranteed (with a very high degree of certainty) to be correct, the tuples extracted by automatic methods unavoidably contain noisy and uncertain data. Google's KnowledgeVault [13] and Geo DeepDive [37] are examples of large scale knowbases where automatic methods produce highly calibrated probabilistic annotations for its extracted facts. Such systems store probabilistic data, for example, in KnowledgeVault a fact like `BornIn(Obama, Hawaii)` may have probability 0.7.

To control and manage the quality of the data and query answers, these systems rely critically on background knowledge, which is captured in a formalism like Markov Logic Networks [12] (reviewed in Section 2). An MLN consists of a collection of soft and hard constraints, which are quite similar to constraints in databases, but are annotated with a weight, representing the degree to which they should hold in the KB. An MLN introduces complex correlations between the large collection of facts in the knowledge base. Query answering requires probabilistic inference, and this is a challenging task. Answering even the simplest query that returns a single tuple requires reasoning over a large set of correlated tuples. Scalable and reliable probabilistic inference remains a major unsolved challenge for large scale knowledge bases.

All MLN systems today perform inference using some variant of Monte Carlo Markov Chain (MCMC), to sample from the space of possible worlds. While MCMC provably converge, their theoretical convergence rate is too slow for practical purposes. Instead, systems usually run a fixed number of simulation steps, e.g. 10,000, and return whatever they find. However, in practice they perform much worse than one expects. The problem is that, at their core, they need to sample uniformly from the set of solutions of a Boolean formula, and precise uniformity is critical for convergence. But uniform sampling is a challenging task in itself. The state of the art is SampleSAT [35], which is heuristic-based and can only approximate the uniform distribution. Together, the weak theoretical guarantees coupled with the fact that their main assumption does not hold in practice means that MCMC-based systems have very poor quality in practice (more on this in Section 5).

In this paper we propose a new approach to scalable probabilistic inference over large Knowledge Bases. The key new idea in our approach is to combine sampling with lifted inference (also called safe query evaluation), thus converting the standard 0/1 estimation problem, where each sample leads to either 0 or 1, to the problem of estimating the expected value of a function in $[0, 1]$. This allows us to deploy two powerful optimizations: evaluation of a conditional probability, and importance sampling. Finally, we describe a suite of novel query optimizations to speed up the SQL query performing the lifted inference. We give now an overview of our contributions.

We start by translating the MLN into a tuple-independent probabilistic database: the query probability in the MLN becomes a conditional probability $\mathbf{P}(Q|\Gamma)$ in the tuple independent database, where Γ is a constraint. The (unconditional) probability of a formula $\mathbf{P}(\Phi)$ can be estimated using Monte Carlo simulation (sample N worlds, return the fraction that satisfy the formula), but this requires a number of simulation steps inversely proportional to the probability. This works well when Φ is a Union of Conjunctive Queries (UCQ), which is an $\exists^*$ formula and usually has a high probability. But it fails when applied to Γ (the denominator in $\mathbf{P}(Q|\Gamma)$), which is a $\forall^*$ sentence with a tiny probability (say, 10^{-9}). In other words, even though $\mathbf{P}(Q|\Gamma)$ is relatively large (say 0.1 to 0.9), we need a huge number of steps to compute it, because $\mathbf{P}(\Gamma)$ is tiny. This has prevented MLN systems from adopting a translation into probabilistic databases, relying on MCMC instead.

Our new idea is to combine Monte Carlo sampling with lifted inference (also called safe query evaluation), in a technique we call SafeSample. Fix a subset of the relations such that, once these relations are made deterministic, the query can be evaluated in polynomial time. Then run the Monte Carlo simulation by sampling only over these relations, and computing the probability of each sample using a safe query plan; in other words, instead of estimating a 0/1-random variable, we estimate a random variable with values in $[0, 1]$ (also called discrete integration). Safe query evaluation has been studied both in probabilistic databases [34] and in AI [11] where it is called lifted inference; we will use both terms interchangeably in this paper.

The real power of SafeSample comes from enabling two additional optimizations. CondSample evaluates the numerator and denominator of $\mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma)$ together, by using each sample to increment estimates for both the numerator and the denominator. This technique works only in combination with SafeSample. Otherwise, if both numerator and denominator are 0/1-random variables, CondSample becomes equivalent to *rejection sampling*, which samples worlds, rejects those that do not satisfy Γ, and returns the fraction of worlds that satisfy $Q \wedge \Gamma$. Rejection sampling wastes many samples, leading to poor performance. By using SafeSample instead, we can compute the *exact* probability of both $Q \wedge \Gamma$ and Γ at each sample, and add it to both sums. In other words, we no longer waste samples, and instead *make every sample count*. Our main theoretical result (Theorem 3.6) shows that the convergence rate of CondSample is inverse proportional to the conditional probability of Q given Γ, and a parameter called *output-tilt* of Γ (ratio between the largest and smallest probability over all samples). In other words, we no longer depend on the tiny probability of Γ, but instead on the conditional probability $\mathbf{P}(Q|\Gamma)$, and the output-tilt. The second optimization, ImportanceSample, further decreases the output-tilt by weighting samples in inverse proportion to the probability of Γ.

The entire probabilistic inference in SlimShot is done by a SQL query, representing a safe plan, and thus is pushed inside the database engine; unlike other MLN systems, SlimShot does not require a separate grounding step. The safe plan is evaluated once for every sample, hence optimizations affect critically the performance of probabilistic inference. We describe several optimization techniques to achieve high performance. We validate SlimShot experimentally by comparing it with other popular MLN systems, and show that it has dramatically better accuracy at similar or better performance, and that it is the only MLN system that offers relative error guarantees.

Our approach reduces MLNs to Weighted Model Counting (WMC). Recently, there have been three parallel, very promising developments for both exact and approximate WMC. Sentential Decision Diagrams [9] (SDDs) are an exact model counting approach that compile a Boolean formula into circuit representations, where WMC can be done in linear time. SDDs have state-of-the-art performance for many tasks in exact weighted model counting, but also have some fundamental theoretical limitations: Beame and Liew prove exponential lower bounds even for simple UCQ's whose probabilities are in PTIME. WeightMC [5] is part of a recent and very promising line of work [14, 5], which reduces approximate model counting to a polynomial number of oracle calls to a SAT solver. Adapting this techniques to weighted model counting is non-trivial: Chakraborty [5] proves that this is possible if the models of the formula have a small *tilt* (ratio between the largest and smallest weight of any model). The tilt is a more stringent measure than our output-tilt, which is the ratio of two aggregates and can be further reduced by using importance sampling. Finally, a third development consists of lifted inference [28, 4, 33, 11, 18], which are PTIME, exact WMC methods, but only work for a certain class of formulas: in this paper we combine lifted inference with sampling, to apply to all formulas.

In summary, our paper makes the following contributions:

- We describe SafeSample, a novel approach to query evaluation over MLNs that combines sampling with lifted inference, and two optimizations: CondSample and ImportanceSample. We prove an upper bound on the relative error in terms of the output-tilt. Section 3.

- We describe several optimization techniques for evaluating safe plans in the database engine, including techniques for negation, for tables with sparse content or sparse complement, and for evaluating constraints (hence CNF formulas). Section 4.

- We conduct a series of experiments comparing SlimShot to other MLN systems, over several datasets from the MLN literature, proving significant improvements in precision at similar, or better, runtime. Section 5

2. BACKGROUND

We fix a relational vocabulary $\sigma = (R_1, \ldots, R_m)$, and denote $\mathtt{DB} = (R_1^{\mathtt{DB}}, \ldots, R_m^{\mathtt{DB}})$ a database instance over σ. We identify $\mathtt{DB}$ with the set of all tuples, and write $D \subseteq \mathtt{DB}$ to mean $R_i^D \subseteq R_i^{\mathtt{DB}}$ for all i. A First Order formula with free variables $\mathbf{x} = (x_1, \ldots, x_k)$ in *prenex-normal form* is an expression:

$$\Phi(\mathbf{x}) = E_1 y_1 E_2 y_2 \ldots E_\ell y_\ell \varphi(\mathbf{x}, \mathbf{y})$$

where each E_i is either $\forall$ or $\exists$ and φ is a quantifier-free formula using the logical variables $x_1, \ldots, x_k, y_1, \ldots, y_\ell$. A *sentence* is a formula without free variables. In this paper we consider two kinds of formulas: a *query* is a formula with quantifier prefix $\exists^*$, and a *constraint* is a formula with quantifier prefix $\forall^*$; note that both queries and constraints may have free variables. A query can be rewritten as $Q = C_1 \vee C_2 \vee \cdots$ where each C_i is a conjunctive query with negation, while a constraints can be written as $\Delta_1 \wedge \Delta_2 \wedge \cdots$ where each Δ_i is a clause with quantifier prefix $\forall^*$.

Equivalence of queries $Q \equiv Q'$ is NP-complete [31], and, by duality, equivalence of constraints $\Gamma \equiv \Gamma'$ is coNP-complete.

2.1 Probabilistic Databases

A *tuple-independent probabilistic database*, or probabilistic database for short, is a pair $(\mathtt{DB}, p)$ where $p : \mathtt{DB} \to [0, 1]$ is a function that associates to each tuple $t \in \mathtt{DB}$ a probability $p(t)$. It defines a probability space on the set of possible worlds, where each tuple t is included independently, with probability $p(t)$. Formally, for each subset $D \subseteq \mathtt{DB}$, called a *possible world*, its probability is $\mathbf{P}_{\mathtt{DB},p}(D) = \prod_{t\in D} p(t) \cdot \prod_{t\in \mathtt{DB}-D}(1 - p(t))$. The probability of a sentence Φ is:

$$\mathbf{P}_{\mathtt{DB},p}(\Phi) = \sum_{D\subseteq \mathtt{DB}: D\models\Phi} \mathbf{P}_{\mathtt{DB},p}(D)$$

If $Q(\mathbf{x})$ is a query, then its output is defined as the set of pairs $(\mathbf{a}, p)$, where $\mathbf{a}$ is a tuple of constants of the same arity as $\mathbf{x}$, and p is the probability of the Boolean query obtained by substituting the free variables $\mathbf{x}$ with the constants $\mathbf{a}$, $p \overset{\text{def}}{=} \mathbf{P}_{\mathtt{DB},p}(Q[\mathbf{a}/\mathbf{x}])$. We drop the subscripts from $\mathbf{P}_{\mathtt{DB},p}$ when clear from the context.

A relation R is called *deterministic*, if for every tuple $t \in R^{\mathtt{DB}}$, $p(t) \in \{0, 1\}$, otherwise it is called probabilistic. We sometimes annotate with a subscript R_d the deterministic relations. We denote A the active domain of the database, and $n = |A|$.

Query Evaluation. In general, computing $\mathbf{P}(\Phi)$ is #P-hard in the size of the active domain[1]. The standard approach is to first compute the *grounding* of Φ on the database $\mathtt{DB}$, also called the *lineage* [1], which is a Boolean formula, then compute the probability of this Boolean formula; we do not use lineage in this paper and will not define it formally. If Φ is an $\exists^*$ sentence, then the lineage is a DNF formula, which admits an FPTRAS using Karp and Luby's sampling-based algorithm [23]. But the lineage of a $\forall^*$ sentences is a CNF formula, and even very restricted classes of CNF formulas do not admit an FPTRAS unless P=NP [30].

An alternative approach to compute $\mathbf{P}(\Phi)$ is called *lifted inference* in the Statistical Relational Learning literature [11], or *safe query evaluation* in probabilistic databases [34]. It always runs in PTIME in n, but only works for some sentences Φ. Following [34], lifted inference proceeds recursively on Φ, by applying the rules in Table 1, until it reaches ground atoms, whose probabilities are looked up in the database. Each rule can only be applied after checking a certain syntactic condition on the formula Φ; if no rule can be applied, lifted inference fails. When the rules succeed we call Φ *safe*, or *liftable*; otherwise we call it *unsafe*. For a simple illustration, if T_d is a deterministic relation, then $\Gamma_1 = \forall x\forall y(R(x) \vee S(x, y) \vee T_d(y))$ is safe, because x is a separator variable, in other words $\mathbf{P}(\Gamma_1) = \prod_{a\in A} \mathbf{P}(R(a) \vee S(a, y) \vee T_d(y))$, where $\mathbf{P}(R(a) \vee S(a, y) \vee T_d(y)) = 1 - (1 - \mathbf{P}(R(a))) \cdot \prod_{b\in A}(1 - \mathbf{P}(S(a, b))) \cdot (1 - \mathbf{P}(T_d(b)))$. On the other hand, if all three relations are probabilistic, then $\forall x\forall y(R(x) \vee S(x, y) \vee T(y))$ is #P-hard: we call it unsafe, or non-liftable. We refer the reader to [34] for more details on lifted inference. We note that in the literature the term lifted inference sometimes refers to symmetric databases [10]: a relation R is called *symmetric* if all ground tuples $R(t)$ over the active domain have the same probability, and a probabilistic database is called symmetric if all its relations are symmetric. In this paper we do not restrict databases to be symmetric, and will use lifted inference to mean the same thing as safe query evaluation.

[1]It remains #P-hard in the size of the database, since these two are polynomially related.

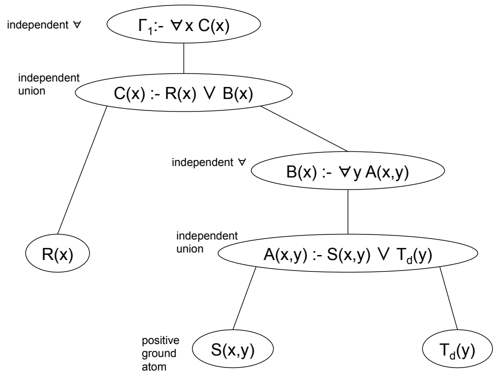


Figure 1: Safe plan for $\Gamma_1 = \forall x\forall y(R(x) \vee S(x, y) \vee T_d(y))$.

Safe plans. Following other systems [3, 27], SlimShot performs lifted inference by rewriting the query into a *safe query plan*, which is then evaluated inside a relational database engine. The leaves of the plan are relations with a special attribute called p, representing the probability of the tuple. There is one operator for each rule in Table 1, which computes the probabilities of the output in terms of the input. For example the independent join operator multiplies the probabilities of the left and the right operand, while the independent $\forall$ aggregates all probabilities in a group by multiplying them. We describe more details of the safe plans in Sec.4. For example, the query Q_1 has the safe plan shown in Figure 1.

2.2 Markov Logic Networks

An MLN is a set of pairs $(w, \Delta(\mathbf{x}))$, where $\Delta(\mathbf{x})$ is a constraint with free variables $\mathbf{x}$, and $w \in [0, \infty]$ is a weight. If $w = \infty$ then we call Δ a *hard constraint*, otherwise a *soft constraint*. For example:

$$(3, \text{Smoker}(x) \wedge \text{Friend}(x, y) \Rightarrow \text{Smoker}(y)) \tag{1}$$

is a soft constraint with weight $w = 3$ saying that, typically, friends of smokers are smokers.

For a fixed domain A, a possible world D is a set of ground tuples over the domain A that satisfies all hard constraints. Its weight is computed as follows: for each soft constraint $(w, \Delta(\mathbf{x}))$ and for each tuple of constants $\mathbf{a}$ such that $\Delta(\mathbf{a})$ holds in D, multiply its weight by w:

$$W_{MLN}(D) = \prod_{(w,\Delta(\mathbf{x}))\in MLN, \mathbf{a}\in A^{|\mathbf{x}|}: w<\infty \wedge D\models\Delta[\mathbf{a}/\mathbf{x}]} w \tag{2}$$

For example, considering an MLN that consists only of the soft constraint (1), the weight of a possible world is 3^N, where N is the number of pairs a, b such that the implication $\text{Smoker}(a) \wedge \text{Friend}(a, b) \Rightarrow \text{Smoker}(b)$ holds in D. The probability of a possible world D is defined as the normalized weight: $\mathbf{P}_{MLN}(D) = W_{MLN}(D)/Z$, where

Φ	$\mathbf{P}(\Phi)$	Rule name	Conditions to check
$R(t)$	$p(R(t))$	Positive ground atom	–
$\neg R(t)$	$1 - p(R(t))$	Negated ground atom	–
$\Phi_1 \wedge \Phi_2$	$\mathbf{P}(\Phi_1) \cdot \mathbf{P}(\Phi_2)$	Independent join	no common probabilistic relation symbol in Φ_1, Φ_2
$\Phi_1 \vee \Phi_2$	$1 - (1 - \mathbf{P}(\Phi_1)) \cdot (1 - \mathbf{P}(\Phi_2))$	Independent union	no common probabilistic relation symbol in Φ_1, Φ_2
$\forall x \Phi$	$\prod_{a \in A} \Phi[a/x]$	Independent $\forall$	x is a separator variable (see caption)
$\exists x \Phi$	$1 - \prod_{a \in A}(1 - \Phi[a/x])$	Independent $\exists$	x is a separator variable (see caption)
$\Phi_1 \vee \Phi_2$	$\mathbf{P}(\Phi_1) + \mathbf{P}(\Phi_2) - \mathbf{P}(\Phi_1 \wedge \Phi_2)$	I/E for constraints	–
$\Phi_1 \wedge \Phi_2$	$\mathbf{P}(\Phi_1) + \mathbf{P}(\Phi_2) - \mathbf{P}(\Phi_1 \vee \Phi_2)$	I/E for queries	–

Table 1: Save Query Evaluation Rules for $\mathbf{P}_{\mathtt{DB},p}(\Phi)$. A *separator variable* is one that occurs in every probabilistic atom, and, if two atoms have the same relational symbol, then it occurs in the same position in both.

$Z = \sum_D W_{MLN}(D)$. The probability of a query Q is $\mathbf{P}_{MLN}(Q) = \sum_{D:D\models Q} \mathbf{P}_{MLN}(D)$.

Notice that a tuple-independent probabilistic database is a special case of an MLN, where each soft constraint consists of a ground tuple, $(w, R(\mathbf{a}))$, where $w = p(R(\mathbf{a}))/(1 - p(R(\mathbf{a})))$ is the odds of that tuple.

State of the art. MLN's have been used in information extraction, record linkage, large scale text processing, and data collection from scientific journals [12, 25, 37]. MLN systems like Tuffy [25] and DeepDive [37] scale up by storing the *evidence* (hard constraints consisting of a single ground tuple) in a relational database system, and split query evaluation into two parts: grounding and probabilistic inference. Grounding is performed in the database engine [25], probabilistic inference is done entirely outside the database engine. Inference remains the major challenge to date: all MLN systems use MCMC, which, as we show in Section 5, can suffer from poor accuracy in practice.

Translation to Probabilistic Databases. Somewhat surprisingly, every MLN can be converted into a tuple-independent probabilistic database. One simple way to do this is to replace each soft rule $(w, \Delta(\mathbf{x}))$ with two new rules:

$$(w, R(\mathbf{x})) \qquad (\infty, \forall \mathbf{x} R(\mathbf{x}) \Leftrightarrow \Delta(\mathbf{x})) \tag{3}$$

where $R(\mathbf{x})$ is a new relational symbol (a new symbol for each rule), of the same arity as the free variables of $\Delta(\mathbf{x})$. After this transformation, the new MLN consists of the new tuple-independent relations $R(\mathbf{x})$, plus hard constraints of the form (3); denote Γ the conjunction of all hard constraints. Let $\mathbf{P}_{MLN}$ and $\mathbf{P}$ be the probability space defined by the MLN, and by the tuple-independent probabilistic database. The following can be easily checked, for any query Q:

$$\mathbf{P}_{MLN}(Q) = \mathbf{P}(Q|\Gamma) = \mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma) \tag{4}$$

In other words, we have reduced the problem of computing probabilities in the MLN to the problem of computing a conditional probability over a tuple-independent probabilistic database. Notice that Γ is a $\forall^*$ sentence, hence $\mathbf{P}_{MLN}(Q)$ no longer admits an FPTRAS, because the grounding of a $\forall^*$ sentence is a CNF formula.

One disadvantage of the translation (3) is that the hard rule has two negations: $(\neg R(\mathbf{x}) \vee \Delta(\mathbf{x})) \wedge (R(\mathbf{x}) \vee \neg\Delta(\mathbf{x}))$. In this paper we use a more effective translation from MLN's to probabilistic databases, adapted from [22]: replace each soft rule $(w, \Delta(\mathbf{x}))$ by the following two new rules,

$$(w - 1, R(\mathbf{x})) \qquad (\infty, \forall \mathbf{x} \neg R(\mathbf{x}) \vee \Delta(\mathbf{x})) \tag{5}$$

The new hard constraint is simpler: if Δ is a single clause (the typical case in MLN), the translation is also a single clause. Eq.(4) still holds for this translation. To see this, consider a world D over the vocabulary of the old MLN, and a tuple of constants, $\mathbf{a}$. If $D \not\models \Delta(\mathbf{a})$, then $\mathbf{a}$ does not contribute to the weight of D in Eq.(2): in the new MLN, the hard constraint (5) requires $R(\mathbf{a})$ to be false, and $\mathbf{a}$ also does not contribute any factor to the weight. If $D \models \Delta(\mathbf{a})$, then in the old MLN the constants $\mathbf{a}$ contributed a factor of w, and in the new world there are two possibilities $R(\mathbf{a})$ is true or is false, and these two worlds contribute jointly a weight $(w - 1) + 1 = w$.

2.3 Chernoff Bound and Monte Carlo Simulation

If $X_1, \ldots, X_N \in [0, 1]$ are i.i.d. with mean x, then Chernoff's Bound is [20]:

$$\mathbf{P}(\sum_{i=1,N} X_i \geq (1 + \delta)Nx) \leq \mathtt{exp}\,(-N \cdot D((1 + \delta)x||x)) \tag{6}$$

where $D(z||x) = z \cdot ln(\frac{z}{x}) + (1 - z) \cdot ln(\frac{1-z}{1-x})$ is the binary relative entropy. By using the inequality $D((1+\delta)x||x) \geq x \cdot h(\delta)$, where $h(x) = (1 + x)ln(1 + x) - x$, and further using $h(\delta) \geq \delta^2/3$ for $\delta \leq 1/2$, the probability on the right simplifies to $\mathtt{exp}(-Nx\delta^2/3)$. All variants of Chernoff bounds discussed in this paper have both upper bounds ($\mathbf{P}(\sum X_i \geq \cdots)$) and lower bounds ($\mathbf{P}(\sum X_i \leq \cdots)$), with slightly different constants, but to simplify the presentation we follow common practice and discuss only the upper bound.

If f is a real-valued random variable, the Monte Carlo estimator for $x = \mathbf{E}[f]$ consists of computing N independent samples of f, denoted $X_1, \ldots, X_N$, then returning:

$$\hat{x} = \frac{\sum_{i=1,N} X_i}{N} \tag{7}$$

If $f \in [0, 1]$, then Chernoff's bound applies, and it follows that we need $N \gtrsim 1/(x\delta^2)$ samples (ignoring a small constant factor) in order for the estimator $\hat{x}$ to have relative error δ with high probability. In practice, of course, x is unknown, however Dagum [7] describes a dynamic stopping condition that guarantees the error bound δ, without knowing x. In summary, the Monte Carlo estimator guarantees an error bound, requiring $1/(x\delta^2)$ simulation steps.

3. SlimShot

SlimShot performs query evaluation on MLNs. The data (evidence) is stored in a relational database, and the probabilistic inference is pushed inside the relational database engine; there is no separate grounding step. The MLN is translated into a tuple-independent probabilistic database, using Eq.(5), then $P_{MLN}(Q)$ is computed by the conditional probability:

$$\mathbf{P}(Q|\Gamma) = \frac{\mathbf{P}(Q \wedge \Gamma)}{\mathbf{P}(\Gamma)} \tag{8}$$

We denote $x = \mathbf{P}(Q|\Gamma)$ throughout this section.

A naive approach to compute (8) is to estimate the numerator and denominator separately, using Monte Carlo sampling. Throughout this section we define f as the 0/1-function $f(D) = 1$ when $D \models \Gamma$, and $f(D) = 0$ otherwise, where D is a possible world, and denote $y = \mathbf{E}[f] = \mathbf{P}(\Gamma)$. We could use a Monte Carlo estimator for $\mathbf{E}[f]$, but this is impractical because Γ is a $\forall^*$ sentence, hence y is a tiny quantity, say, 10^{-9}, requiring $N \gtrsim 10^{11}$ simulation steps to guarantee a relative error $\delta = 0.1$. In contrast, $x = \mathbf{P}(Q|\Gamma)$ is relatively large, say 0.1 to 0.9, but to compute it we need to divide two tiny quantities.

SlimShot performs three extensions to the naive Monte Carlo simulation, each with a provable reduction in the required number of steps: (1) SafeSample: combines sampling with exact lifted inference, (2) CondSample: guarantees an error bound for the conditional directly, without relying on separate error bounds for the numerator and denominator, and (3) ImportanceSample: deploys importance sampling to further improve the convergence rate.

3.1 SafeSample

The main idea of our entire approach is to combine sampling with lifted inference; we call this SafeSample. We start with a definition:

Definition 3.1. *Let Φ be a sentence, and $\mathbf{T}$ be a set of relation names. We say that Φ is* safe modulo $\mathbf{T}$ *if it becomes safe after making all relation names in $\mathbf{T}$ deterministic.*

Throughout this section we denote $g(\mathbf{T}^D) = \mathbf{P}(\Gamma|\mathbf{T} = \mathbf{T}^D) = \mathbf{E}_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]$, where $\mathbf{T}^D$ is a possible world for the relations $\mathbf{T}$. If Γ is safe modulo T, then the function $g(\mathbf{T}^D)$ can be computed in polynomial time. For example the constraint $\forall x\forall y(R(x) \vee S(x,y) \vee T(y))$ is unsafe, but it is safe modulo T, because once we make T deterministic, the constraint becomes safe; the function $g(T^D)$ is computed by the safe plan in Figure 1. We will usually denote T a relation in $\mathbf{T}$, and denote R any other relation[2]

SafeSample is the naive Monte Carlo algorithm applied to the $[0,1]$-valued function g, instead of the 0/1-function f, to compute $\mathbf{P}(\Gamma)$. It samples N possible world $\mathbf{T}^{D_i}$, $i = 1, N$, then returns the estimate

$$\hat{y} = \frac{\sum_{i=1,N} g(\mathbf{T}^{D_i})}{N} \tag{9}$$

This is an unbiased estimator, because $E_{\mathbf{T}}[g] = \mathbf{E}_{\mathbf{T}}[\mathbf{E}_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]] = \mathbf{E}[f] = y$.

Estimating $\mathbf{P}(\Gamma)$ using (9), and similarly $\mathbf{P}(Q \wedge \Gamma)$, then dividing them, leads to very poor performance. We never use SafeSample in isolation, but only in combination with CondSample and ImportanceSample, described in the next sections. However, we can prove that, theoretically, SafeSample alone reduces the number of steps over the (even more naive) 0/1-Monte Carlo estimator for $\mathbf{P}(\Gamma)$. While this is implied by Rao-Blackwell's theorem, we can characterize the speedup precisely.

Proposition 3.2. *Let $g \geq 0$ be a random variable s.t. $g \leq c$ from some constant c, with mean $y = \mathbf{E}[g]$ and variance $\sigma^2 = \mathbf{E}[g^2] - \mathbf{E}^2[g]$. Let $\hat{y}$ be the estimator in Eq.(9). Then, for all $\delta \leq \sigma^2/(2cy)$:*

$$\mathbf{P}(\hat{y} \geq N(1+\delta)y) \leq 2\,exp(-\frac{N\delta^2 y^2}{3\sigma^2})$$

Proof. Bennett's theorem states that, if $X_1, \ldots, X_N$ are iid's s.t. $|X_i| \leq c$ with mean 0 and variance σ^2, then $\mathbf{P}(\sum_i X_i \geq t) \leq \texttt{exp}(-\frac{N\sigma^2}{c^2}h(\frac{ct}{N\sigma^2}))$. By setting $X_i = g(D_i) - y$, $t = N \cdot \delta \cdot y$ we obtain $\mathbf{P}(\hat{y} \geq N(1+\delta)y) \leq \texttt{exp}(-N\frac{\sigma^2}{c^2}h(\frac{ct}{N\sigma^2})) = \texttt{exp}(-N\frac{\sigma^2}{c^2}h(\frac{c\delta y}{\sigma^2}))$, and finally we use the fact that $h(x) \geq x^2/3$ for $0 \leq x \leq 1/2$. $\square$

Thus, the number of steps required by (9) to estimate y with an error $\leq \delta$ is $N \gtrsim \sigma^2/(y^2\delta^2)$. SafeSample is faster than a 0/1-Monte Carlo estimator by a factor equal to the ratio of the variances, $\sigma^2(f)/\sigma^2(g)$. Notice that $\sigma^2(f) - \sigma^2(g) = \mathbf{E}[f^2] - \mathbf{E}_{\mathbf{T}}[g^2] = \mathbf{E}_{\mathbf{T}}[E_{\mathbf{R}}[f^2|\mathbf{T} = \mathbf{T}^D]] - \mathbf{E}_{\mathbf{T}}[\mathbf{E}^2_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]]$. In other words, the variance due to the random choices of $\mathbf{R}$ is eliminated (since we don't sample $\mathbf{R}$ but use lifted inference instead); the only variance that remains is due to $\mathbf{T}$. This implies that the speedup $\sigma^2(f)/\sigma^2(g)$ is always ≥ 1. We show two examples: one in which the speedup improves exponentially with the size of the domain, the other where it is only a small constant factor.

Example 3.3. *Consider $\Gamma = \forall x(R(x) \vee T(x))$, and a symmetric probabilistic database over a domain of size n, where, for all $i \in [n]$, the tuple $R(i)$ has probability $p(R(i)) = r$ and $T(i)$ has probability t. We show that the speedup $\sigma^2(f)/\sigma^2(g)$ grows exponentially with the domain size. We have $\mathbf{P}(\Gamma) = \mathbf{E}[f] = \mathbf{E}[f^2] = (r + t - rt)^n$, hence:*

$$\sigma^2(f) = (r + t - rt)^n - (r + t - rt)^{2n}$$

If T^D has size $|T^D| = n - k$, then $g(T^D) = \mathbf{E}[f|T = T^D] = r^k$, which implies $\mathbf{E}_T[g] = \sum_k \binom{n}{k} t^{n-k}(1-t)^k r^k = (t + (1-t)r)^n$, and similarly $\mathbf{E}_T[g^2] = (t + (1-t)r^2)^n$, or

$$\sigma^2(g) = (t + (1-t)r^2)^n - (t + (1-t)r)^{2n}$$

When $r = t = 1/2$, then the variance decreases from $\sigma^2(f) = (3/4)^n - (9/16)^n = (12^n - 9^n)/16^n$ to $\sigma^2(g) = (5/8)^n - (3/4)^{2n} = (10^n - 9^n)/16^n$. Their ratio $\sigma^2(f)/\sigma^2(g) = (12^n - 9^n)/(10^n - 9^n) \approx (6/5)^n$.

Example 3.4. *Consider now $\Gamma = \forall x\forall y(R(x) \vee S(x,y) \vee T(y))$. As before, we consider for illustration a symmetric database, where $R = T = [n]$, $S = [n] \times [n]$, and the tuples in R, S, T have probabilities r, s, t respectively. We show that here the speedup is only a small constant factor. We sample T, and let R, S be the random relations. If $|T^D| = n - k$, then $g(T^D) = \mathbf{P}(\Gamma|T^D) = (r + s^k(1-r))^n$, because for every value*

[2]Suggesting de**T**erministic and **R**andom, although the relations $\mathbf{T}$ are *not* deterministic.

of the variable $x = i$, the sentence $\forall y \in [k](R(i) \vee S(i,y))$ must be true. Thus, we have

$$\mathbf{P}(\Gamma) = \mathbf{E}[f] = \mathbf{E}[f^2] = \sum_{k=0,N} \binom{n}{k} t^{n-k}(1-t)^k(r+s^k(1-r))^n$$

$$\mathbf{E}_T[g^2] = \sum_{k=0,N} \binom{n}{k} t^{n-k}(1-t)^k(r+s^k(1-r))^{2n}$$

Here the decrease in variance is only by a constant factor because, if we group the terms in $\mathbf{E}[f^2]-\mathbf{E}_T[g^2]$ by k, then for each $k > 0$, the difference $(r+s^k(1-r))^n - (r+s^k(1-r))^{2n}$ is $\leq (r+s^k(1-r))^n(1-r)(1-s)$. That, is except for the first term $k = 0$ (whose contribution to the sum is negligible), all others decrease by a factor of at most $(1-r)(1-s)$.

As the last example suggest, SafeSample alone is insufficient, which justifies our second technique.

3.2 CondSample

The main advantage of SafeSample is that it leads to dramatic speedup when we compute the numerator and denominator of the conditional probability in Eq.(8) together: we call this technique CondSample. As we will show, the improvement happens only if we use SafeSample: otherwise, if the denominator consists of 0/1-random variables, computing the numerator and denominator together is equivalent to rejection-sampling, which has very poor performance.

Given a set of relations $\mathbf{T}$ such that both $Q \wedge \Gamma$ and Γ are safe modulo $\mathbf{T}$, CondSample estimates Eq.(8) by the following quantity:

$$\hat{x} = \frac{\sum_{i=1,N} \mathbf{P}(Q \wedge \Gamma | \mathbf{T}^{D_i})}{\sum_{i=1,N} \mathbf{P}(\Gamma | \mathbf{T}^{D_i})} \qquad (10)$$

Notice that, for any fixed N, $\hat{x}$ is a biased estimator of x; however, $\hat{x}$ converges to x when $N \to \infty$.

Chakraborty [5] define the *tilt* of a Boolean formula as the ratio between the largest and smallest weight of any of its models. We adapt their terminology to a random variable:

Definition 3.5. *The* output-tilt *of a random variable $Y \geq 0$ is $T = \max Y / \min Y$.*

We prove:

Theorem 3.6. *Let $(X_1, Y_1), \ldots, (X_N, Y_N)$ be i.i.d. such that for all i, $X_i \in [0,1]$ and has mean $x = \mathbf{E}[X_i]$, and $Y_i \geq 0$. Let T be the output-tilt of Y_i. Then:*

$$\mathbf{P}\left(\frac{\sum_{i=1,N} X_i Y_i}{\sum_{i=1,N} Y_i} \geq (1+\delta)x\right) \leq \mathit{exp}(-ND/T) \qquad (11)$$

where $D = D((1+\delta)x||x)$.

Proof. We use the following lemma:

Lemma 3.7. *Let $y_1, \ldots, y_N \geq 0$ be N real numbers, and let $M = (\sum_i y_i)/(\max_i y_i)$; notice that $M \leq N$. Let $X_1, \ldots, X_N$ be i.i.d.'s, where each $X_i \in [0,1]$ and has mean $x = \mathbf{E}[X_i]$. Then, for any $\delta > 0$:*

$$\mathbf{P}(\sum_i X_i y_i > (1+\delta)x(\sum_i y_i)) \leq \mathit{exp}(-M \cdot D)$$

(where $D = D((1+\delta)x||x)$).

Intuitively, the lemma generalizes two extreme cases: when all weights y_i are equal, then $M = N$ and the bound becomes the Chernoff bound for N items; and when the weights are as unequal as possible, $y_1 = y_2 = \ldots = y_M$, $y_{M+1} = \ldots = y_N = 0$, then the bound becomes the Chernoff bound for M items. The proof is included in [17].

To prove the theorem, we condition on the outcomes of the variables Y_i, then apply the lemma:

$$\begin{aligned}\mathbf{P}(\cdots) =& \mathbf{E}_{Y_1,\ldots,Y_N}\left[\mathbf{P}_{X_1,\ldots,X_N}\left(\frac{\sum_{i=1,N} X_i Y_i}{\sum_{i=1,N} Y_i} \geq (1+\delta)x\right)\right] \\ \leq& \mathbf{E}_{Y_1,\ldots,Y_N}[\mathtt{exp}(-(\sum_{i=1,N} Y_i)/(\max_{j=1,N} Y_j) \cdot D)] \\ \leq& \mathbf{E}_{Y_1,\ldots,Y_N}[\mathtt{exp}(-N/T \cdot D)] = \mathtt{exp}(-N/T \cdot D)\end{aligned}$$

proving the claim. □

By setting $X_i = \mathbf{P}(Q|\Gamma, T^{D_i})$ and $Y_i = \mathbf{P}(\Gamma|T^{D_i})$, Eq.(10) becomes $\sum_i X_i Y_i / \sum_i Y_i$, which gives us the error bound (11) for the estimator $\hat{x}$. It suffices to run $N \gtrsim T/D \approx T/(x\delta^2)$ simulation steps, in other words the number of steps depends on the mean x of X_i and the output-tilt of Y_i; it does not depend on the mean y of Y_i. The variables X_i, Y_i in Theorem 3.6 do not have to be independent[3], which justifies using the same sample $\mathbf{T}^{D_i}$ both in the numerator and the denominator.

The reader may wonder at this point why we don't estimate $x = \mathbf{P}(Q|\Gamma)$ directly, as $\sum_i \mathbf{P}(Q|\Gamma, T^{D_i})/N$: this only requires $N \gtrsim 1/(x\delta^2)$ steps, and in practice x is large enough. The problem is, however, that we need to sample possible worlds $T^{D_1}, T^{D_2}, \ldots$ from the *conditional* distribution $\mathbf{P}(T^D|\Gamma)$, and this task is as difficult as estimating $\mathbf{P}(\Gamma)$ [21], an NP-hard problem.

The speedup given by Theorem 3.6 is only possible in combination with SafeSample. If Y_i is a 0/1-variable, then the output-tilt is infinite, and the theorem becomes vacuous. In fact, in that case CondSample becomes *rejection sampling* for computing $\mathbf{P}(Q|\Gamma)$: repeatedly sample a world D_i, ignore worlds that do not satisfy Γ, and return the fraction of worlds that satisfy Q. Rejection sampling is known to require $N \gtrsim 1/\mathbf{P}(\Gamma)$ steps, because we need as many steps in expectation to hit Γ once. Rejection sampling wastes simulation steps. In contrast, when using SafeSample, *we make every sample count*: even if our sample gives a very small probability $\mathbf{P}(\Gamma|\mathbf{T}^D)$, it is still useful for us, because we can compute it exactly and add to the sum.

The speedup achieved by CondSample depends on the output-tilt; in the next section we show how importance sampling can further reduce the output-tilt. We end this section showing two examples. In the first the output-tilt is large, and CondSample is not much better than SafeSample, the second where the output-tilt is exponentially smaller.

Example 3.8. *Consider first $\Gamma = \forall x \forall y(R(x) \vee S(x,y) \vee T(y))$ in Example 3.4. As we have seen, if $|T^D| = n-k$, then $Y = \mathbf{P}(\Gamma|T^D) = (r+s^k(1-r))^n$, (because for every value of the variable $x = i$, the sentence $\forall y \in [k](R(i) \vee S(i,y))$ must be true). The maximum value of Y is 1 (for $k = 0$) and the minimum is $(r+s^n(1-r))^n \approx r^n$ respectively, thus the output-tilt is $1/r^n$, which is much bigger than $1/\mathbf{P}(\Gamma)$. In general, when* $\max(Y) = 1$, *then the output-tilt is* $1/\min(Y)$ *and this is bigger than* $1/\mathbf{E}[Y]$ *because the* $\min(Y) \leq \mathbf{E}[Y] \leq \max(Y)$.

[3]But (X_i, Y_i) have to be independent of (X_j, Y_j).

Example 3.9. *Consider next $\Gamma = \forall x \forall y(R_1(x) \vee S_1(x,y) \vee T(y)) \wedge (R_2(x) \vee S_2(x,y) \vee \neg T(y))$. This constraint is safe modulo T, but now we no longer have $\mathbf{P}(\Gamma|T^D) = 1$, for any value T^D, and we show that the output-tilt is much smaller than $1/\mathbf{E}[Y]$. Notice that, one can show (by repeating the argument in Example 3.4) that SafeSample alone is insufficient to compute this query much faster than a naive Monte Carlo simulation, so CondSample is necessary for a significant speedup. To compute the output-tilt, note that $Y = \mathbf{P}(\Gamma|T^D) = (r_1 + s_1^k(1-r_1))^n(r_2 + s_2^{n-k}(1-r_2))^n$ and, assuming $r_1 = r_2$ and $s_1 = s_2$, the maximum/minimum values are $(r + s^n(1-r))^n \approx r^n$ (for $k = 0$ or $k = n$) and $(r + s^{n/2}(1-r))^{2n} \approx r^{2n}$ (for $k = n/2$) respectively. The output-tilt is $1/r^n$ and it is much smaller than $1/\mathbf{E}[Y]$. To see this, assume for illustration that $t = 1/2$, then we claim that $\mathbf{E}[Y] \le 3r^{2n}$. Expand $\mathbf{E}[Y] = \sum_k \binom{n}{k}\frac{1}{2^n}(r + s^k(1-r))^n(r + s^{n-k}(1-r))^n$, and split the sum into two regions: for $k \in [n(1-\delta)/2, n(1+\delta)/2]$ the contribution of the sum is $\le (r + s^{n(1-\delta)/2}(1-r))^{2n} \approx r^{2n}$, while for $k \notin [n(1-\delta)/2, n(1+\delta)/2]$ the contribution of the sum is[4] is $\le 2r^n \mathbf{exp}(-n\delta^2)$. It suffices to choose δ such that $e^{-\delta^2} \le r$ (which is possible when $r > 1/e \approx 0.36$) to prove our claim.*

The examples suggest that CondSample works best for complex MLNs, where no setting of the relations $\mathbf{T}$ can make Γ true (and, thus, $\max Y \ll 1$); still, it is insufficient for speeding up all queries. Our third technique further improves the convergence rate by adding importance sampling.

3.3 ImportanceSample

Importance sampling [8] chooses a proposal distribution for the random variable $\mathbf{T}$, $\mathbf{P}'(\mathbf{T})$, then computes the expected value $\mathbf{E}'[g']$ of a corrected function, $g'(\mathbf{T}^D) = g(\mathbf{T}^D) \cdot \mathbf{P}(\mathbf{T}^D)/\mathbf{P}'(\mathbf{T}^D)$'. This is an unbiased estimator: to see this, we apply directly the definition of $\mathbf{E}[g]$ as a sum over all possible worlds:

$$\mathbf{E}[g] = \sum_{T^D} g(\mathbf{T}^D)\mathbf{P}(\mathbf{T}^D) = \sum_{T^D} g'(\mathbf{T}^D)\mathbf{P}'(\mathbf{T}^D)$$

Ideally we would like $\mathbf{P}'(\mathbf{T}^D)$ to be proportional to $g(\mathbf{T}^D) \cdot \mathbf{P}(\mathbf{T}^D)$, because in that case g' is a constant function, thus has output-tilt 1, but computing such a proposal distribution is infeasible, because of the cost of computing the normalization factor.

Instead, we define $\mathbf{P}'(\mathbf{T}^D)$ as function of the cardinalities of the relations T^D in $\mathbf{T}^D$. For presentation purposes assume $\mathbf{T} = \{T\}$ consists of a single relation, and first describe a naive ImportanceSample. For every $k = 1, n^{\mathtt{arity}(T)}$ (recall: n is the size of the active domain), sample one relation T^{D_k} of size k, and compute $p_k = \mathbf{P}(\Gamma|T^{D_k})$. Let $q = \sum_k \binom{n^{\mathtt{arity}(T)}}{k} p_k$ be the normalization factor. Then the proposal distribution is $\mathbf{P}'(T^D) = p_k/q$, where $k = |T^D|$. Intuitively, this decreases the output-tilt of the correct function g', because the spread of probabilities $\mathbf{P}(\Gamma|T^D)$ decreases if we fix the cardinality of T^D. We prove that, in a special case, the output-tilt becomes 1:

[4]Let Z_i be i.i.d. in $\{0,1\}$ s.t. $\mathbf{P}(Z_i = 0) = \mathbf{P}(Z_i = 1) = 1/2$. Then the following stronger version of Chernoff's bound holds: $\mathbf{P}(\sum Z_i \ge (1+\delta)n/2) \le e^{-n\delta^2}$. Thus, $\sum_{k=n(1+\delta)/2,n} \binom{n}{k}\frac{1}{2^n} \le e^{-n\delta^2}$.

Monte Carlo	Safe sample	Cond sample	Importance sample
$\frac{1}{y\delta^2}$	$\frac{\sigma^2(Y)}{y^2\delta^2}$	$\frac{T}{x\delta^2}$	$\frac{T'}{x\delta^2}$

$X = \mathbf{P}(Q|\Gamma, T^D), x = \mathbf{E}[X]$ $\quad Y = \mathbf{P}(\Gamma|T^D), y = \mathbf{E}[Y]$

$T =$output-tilt of Y $\quad T =$output-tilt of corrected Y'

Table 2: Number of simulation steps N (omitting constant factors) to compute $x = \mathbf{P}(Q|\Gamma)$ with relative error δ.

Proposition 3.10. *If the probabilistic database is symmetric, and Γ is safe modulo a set of unary relations, then the output-tilt of g' is 1.*

The proof follows from the observation that, if T is a unary relation, then in a symmetric database $\mathbf{P}(\Gamma|T^D)$ depends only on the cardinality of T^D.

To reduce the output-tilt on asymmetric databases, we optimize ImportanceSample as follows. First, we transform all non-sampled relations $\mathbf{R}$ into symmetric relations, by setting $\mathbf{P}(R(\mathbf{a}))$ to the average probability of all tuples in R. Then we compute $p_k = \mathbf{P}(\Gamma||T^D| = k)$: we compute the latter probability exactly, even if the relations $\mathbf{T}$ are not symmetric, by adapting techniques from lifted inference over symmetric databases [10]. Notice that, when all relations are symmetric, then the optimized ImportanceSample coincides with the naive ImportanceSample.

Example 3.11. *Continuing Example 3.8, ImportanceSample computes $p_k = \mathbf{P}(\Gamma||T^D| = k) = (r + s^{n-k}(1-r))^n$, for each $k = 0, n$. Define $q = \sum_k \binom{n}{k} p_k$. The proposal distribution is $\mathbf{P}(T^D) = p_{|T^D|}/q$, and the corrected function is $g'(T^D) = (r + s^{n-k}(1-r))^n t^k(1-t)^{n-k}/p_k$. It each iteration step $i = 1, N$, SlimShot samples a value $k = 0, n$ with probability $\binom{n}{k}p_k/q$, then uses reservoir sampling to sample a set T^{D_i} of cardinality k, and computes $X_iY_i = \mathbf{P}(Q \wedge \Gamma|T^{D_i})$ and $Y_i = \mathbf{P}(\Gamma|T^{D_i})$ using lifted inference, adding the quantities to Eq.(11). The value of Y_i is constant (it is always q), because the relations R, S, T are symmetric; if the query Q contains any non-symmetric relations, then X_iY_i is a random variable, and Eq.(11) converges to x after $N \gtrsim 1/(x\delta^2)$ steps; if Q uses only symmetric relations, then X_iY_i is also a constant, and Eq.(11) converges after 1 step.*

We note that ImportanceSample is, again, only possible in combination with SafeSample. If g were a 0/1-random variable, then the corrected function g' is also a 0/1-random variable, and its output-tilt remains infinity.

3.4 Summary

Table 2 summarizes the number of simulation steps N needed to estimate $\mathbf{P}(Q|\Gamma)$ with a relative error δ. Notice that the number of steps depends only on Γ, and not on Q.

4. SYSTEM ARCHITECTURE

SlimShot is written in Python and relies on Postgres 9.3 as its underlying query processing engine. Any other database engine could be used, as long as it supports standard query processing capabilities: inner and outer join, group by, random number generation, and mathematical operators such as sum and logarithm. The Markov Logic Network is given in text file containing first-order $\forall^*$ sentences with associated weights. SlimShot converts these rules into a set of hard constraints Γ and a set of new probabilistic tables, using the translation described in Subsection 2.2 (Eq.(4)). The new relations are materialized in Postgres, with tuple weights converted to probabilities ($p = w/(1 + w)$), stored in an additional attribute.

SlimShot supports unions of conjunctive queries, but in typical MLN applications the query Q consists of a single relation name, e.g. $Q(x) = \texttt{Smokes}(x)$, and the inference engine returns the per-tuple probabilities of all tuples in the `Smokes` relation. SlimShot uses conditional sampling to compute $\mathbf{P}(Q|\Gamma)$ as the ratio of $\mathbf{P}(Q \wedge \Gamma)$ and $\mathbf{P}(\Gamma)$.

4.1 Choosing the relations $\mathbf{T}$

Upon receiving a query Q, SlimShot chooses a minimal set of relation names $\mathbf{T}$ such that both $Q \wedge \Gamma$ and Γ are $\mathbf{T}$-safe. This is done by brute force enumeration: in all our experiments, the cost of brute force enumeration is negligible.

4.2 Review of Safe Plan Evaluation

Safe plan evaluation consists of first converting a safe query into a safe query plan. The operators correspond to the eight rules in Table 1, where the first two (positive/negative atom) are leaf operations and the others are unary or binary operators. Next, each node is the plan is converted into a SQL query. For example, referring to Figure 1, if we assumed that the relations R and B have the same sets of tuples, then the *independent-union* operators $R(x) \vee B(x)$ becomes:

```
select R.x, 1-(1-R.p)*(1-B.p) as p
from R join B on R.x = B.x
```

where B is a sub-query. The actual SQL query is more complex because it uses an outer-join instead of a join. Similarly, an independent join performs an join returning the probability p_1p_2, while the independent $\forall$ and $\exists$ are group-by queries returning probabilities $\prod_i p_i$ and $1 - \prod_i(1 - p_i)$ respectively.

4.3 Enhanced Safe Plan Evaluation

Annoyingly, most popular database engines do not have a product-aggregate operator. We considered two options for $\prod_i p_i$. First is to express it using logarithm and sum, more presiely as $\texttt{exp}(\text{sum}_i(\log p_i))$. This requires a slightly more complex logic to correctly account for tuples with probability zero, or close to zero, or for missing tuples. The second is to define a user-defined aggregate function, UDA.

Another optimization concerns the independent union operator, which, if implemented as suggested above, requires a complicated query with outer joins and a long `case` statement. Instead, we simulate it using a group-by and aggregate, for example the SQL expression above becomes:

```
select T.x, 1-prod(1-T.P)
from (R union B) as T group by T.x
```

The MLN semantics is based on the standard active domain semantics; under this semantics, the answer to the expression $S(x, y) \vee T_d(y)$ in Figure 1 includes all tuples (a, b) where a is any constant in the domain, and $b \in T_d$. MLN implementations enforce this semantics easily by simply representing explicitly all tuples over the current domain. Since our goal is to deploy SlimShot in database applications, we made a significant implementation effort to represent only the tuples present in the database, and avoid as much as possible computing the cartesian product with the active domain. For example, if both R and B contained all constants in the active domain, with probability 0 for tuples not present in the relation, then $R(x) \vee B(x)$ could be computed by the simple join query shown earlier (Subsection 4.2); instead our optimizers uses a more complex logic to handle efficiently sparse relations. We have paid special attention to handling negation, e.g. $R(x) \vee \neg B(x)$, which introduce significant complexity if one wants to avoid materializing non-existing tuples.

MLN	Constraint	w
Smokers	$\neg Smokes(x)$	1.4
	$\neg Cancer(x)$	2.3
	$\neg Friends(x, y)$	4.6
	$Smokes(x) \Rightarrow Cancer(x)$	1.5
	$Smokes(x) \wedge Friends(x, y) \Rightarrow Smokes(y)$	1.1
Drinkers	$Drinks(x) \wedge Friends(x, y) \Rightarrow Drinks(y)$	1.1

Table 3: The Smokers MLN and the Drinkers MLN.

4.4 Further Optimizations

We briefly mention here other optimizations in SlimShot. *Generic Constants:* this refers to computing the probability for all query answers using a single query plan. That is, we have a single query plan to compute $\mathbf{P}(Q(x)|\Gamma)$, returning all pairs (a, p), rather than the naive way of iterating over all constants a in the domain and computing $\mathbf{P}(Q[a/x]|\Gamma)$. We note that MC-SAT algorithm used by existing MLN systems already obtains the probabilities of all outputs x at the same time. *QRel* refers to an optimization that is possible when the query relation Q is a member of the sampled relation set $\mathbf{T}$. In the case we can avoid computing $\mathbf{P}(Q \wedge \Gamma|\mathbf{T}^{D_i})$ (since it is either 0 or 1): instead we only need to compute $\mathbf{P}(\Gamma|\mathbf{T}^{D_i})$ and check if $\mathbf{T}^{D_i} \models Q$.

5. EXPERIMENTS

We validated SlimShot through a series of experiments comparing its performance to other MLN systems on several datasets reported in the literature. We addressed the following questions. How accurate are the probabilities computed by SlimShot compared to existing systems? How does its runtime performance compare to that of existing systems? How does SlimShot handle more complex sets of MLN rules? How effective are the optimizations in SlimShot? And how does SlimShot compare to other, general-purpose weighted model counters?

Datasets We used two datasets from the published literature, Table 3, and three queries, Table 4. Smokers MLN [33] models a social network and the influence of friendship on smoking habits and lung cancer, while the Drinkers MLN [11] adds a new *Drinks* relation. SlimShot converts the MLNs to tuple-independent probabilistic databases by introducing a new relation name for each rule in Table 3 with two or more literals. The Smokers MLN is safe modulo *Smokes*, while the Drinker MLN is safe modulo *Smokes* and *Drinks*.

We considered three variations on these datasets: symmetric, asymmetric unary, and asymmetric. In the first, all probabilities are given by the weights in Table 3. In the second, the binary relation *Friends* is symmetric while all unary relations have distinct, randomly-generated probabilities. Finally, in the asymmetric dataset the *Friends* relation is a randomly-generated graph with fixed fan-out 3, and edge probabilities randomly generated. The database applications

MLN	Query
Smokers and Drinkers	$Q1(x) :- Smokes(x)$ $Q2(x) :- Cancer(x)$
Drinkers only	$Q3(x) :- Drinks(x)$

Table 4: Experiment Queries. Answering these queries requires returning the appropriate probabilities for all tuples in each relation

of interest to us are captured by the third scenario (fully asymmetric), but we needed the first two in order to compute the exact probabilities (ground truth) for most experiments. No system to date can compute the exact probabilities for the asymmetric data.

MLN Systems We ran SlimShot using either CondSample only or using ImportanceSample, and report both results; we use "SlimShot" to refer to ImportanceSample. We compared to two popular MLN systems: Alchemy version 2.0 [2] and Tuffy version 0.4 [25]. Both use MC-SAT for probabilistic inference [29], but they differ in how they perform grounding and their internal implementations of SampleSAT [35]. In earlier work, the first author found several flaws in Tuffy's implementation of MC-SAT, the foremost being a failure to perform simulated annealing steps to explore the solution space before returning a sample within the SampleSAT code, and developed a modified version of Tuffy, currently available at the Allen Institute for Artificial Intelligence (AI2): it incorporates a new implementation of MC-SAT along with a number of other performance improvements such as eliminaton of redundant clauses. We refer to the two versions as Tuffy-Original and Tuffy-AI2.

All our experiments were conducted on a RHEL 7.1 system with 2xIntel Xeon E5-2407v2 (2.4GHz) processors and 48GB of RAM.

5.1 Accuracy

We compared the accuracy of SlimShot to the other MLN systems on queries 1 and 2 over the Smokers MLN. We used only symmetric and unary asymmetric data, because we needed to compute the ground truth; we used a domain of size $n = 100$, resulting in 10200 random variables[5]. Figure 2 shows the maximum relative error[6] for all answers returned by the query, as a function of the number of iterations N. The probability of the constraint, $y = \mathbf{P}(\Gamma)$ was around 10^{-10} while the query probability $x = \mathbf{P}(Q(a)|\Gamma)$ ranged between 0.04 and 0.9. In all experiments SlimShot (ImportanceSample) outperformed all others. For SlimShot we also measured the empirical tilt and report the number of iterations where the theoretical formula (11) predicts that the probability of exceeding the relative error $\delta = 0.1$ is < 0.1: this is the empirical stopping condition used in SlimShot. In all cases, the stopping condition for ImportanceSample was around $N = 1000$ iterations. On symmetric data CondSample had a larger tilt, leading to a much worse stopping condition; $\mathbf{P}(\Gamma)$ is less evenly distributed over possible samples for the lower average tuple probabilities in the symmetric data, and CondSample ends up in regions of very small probability for most of its samples.

5.2 Performance and Scalability

Next, we conducted three rounds of experiments to compare SlimShot's runtime performance to the other systems. Figure 3 shows the runtime for a fixed number of samples ($N = 10,000$), on queries 1 and 2 over the symmetric and unary asymmetric Smokers MLN (same as in the previous section). The fact that all binary relations are complete puts SlimShot at a disadvantage: like any relational plan, the safe plans in SlimShot benefit from sparse relations. In contrast, one simulation step in MC-SAT is rather cheap, favoring Alchemy and Tuffy. Nevertheless, in our experiments the runtime per iteration in SlimShot was within the same order of magnitude as the most efficient system (Tuffy-AI2), sometimes even better.

Second, Figure 4 compares the (much more meaningful) runtime to achieve a fixed relative error. While for SlimShot we can derive a stopping condition from Eq.(11), no stopping condition exists for MC-SAT. Instead, we allowed all systems to run until they achieve for all tuples a maximum relative error ≤ 0.1 compared to the ground truth, and to maintain this for at least ten iterations: as before, we had to restrict to symmetric, and unary asymmetric, data. For both queries and both datasets, we can conclude that SlimShot converges faster than the other systems.

So far, all datasets were symmetric or unary asymmetric only. Third, we studied performance on asymmetric data, which is the main application scenario targeted by SlimShot: since we do not have the ground truth we reverted to reporting the runtime for a fixed number of iterations (10,000). Figure 5 shows that SlimShot is orders of magnitude faster than Alchemy and Tuffy over this type of sparse data: note that the scale is logarithmic. The reason for this is that SlimShot scales linearly with the number of probabilistic tuples present in the database. In contrast, Alchemy and Tuffy must include a unique MLN rule for each tuple missing from the *Friends* relation, expressing that it's probability is zero: the runtime per sample increases quadratically with the domain size. While Tuffy-AI2 optimizes handling of deterministic variables, there is still significant overhead compared to SlimShot.

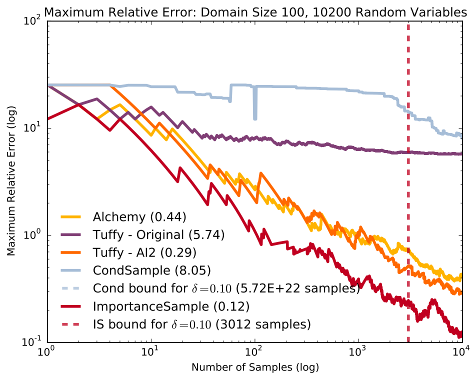
5.3 Richer MLNs

Next, we evaluated SlimShot on a richer MLN: the Drinkers MLN [11]. SlimShot must now simultaneously sample two unary relations, *Smokes* and *Drinks*, which slows down the computation of the proposal distribution. The results for a fixed number of iterations on asymmetric data are shown in Figure 6. While we do not have ground truth for asymmetric data, previous experiments strongly suggests that ImportanceSample is the only system that returns accurate results, so the runtime performance numbers should be interpreted in that light. The first observation is that CondSample is significantly faster than ImportanceSample, because the latter takes time $O(n^3)$ to pre-compute the proposal distribution[7]; the proposal distribution is independent of the query and could be computed and stored offline, but in all our experiments we report it as part of the total runtime.

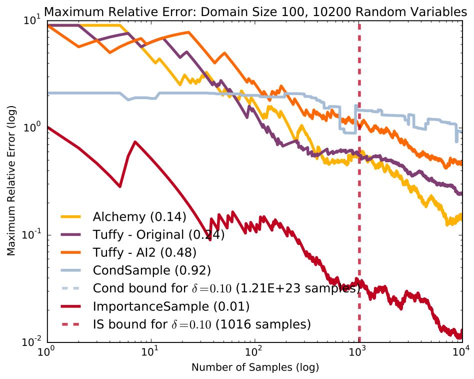
[5]SlimShot's translation to a probabilistic database introduced $10000 + 100$ additional tuples.

[6]We also measured the mean Kullback-Leibler (KL) divergence, which is frequently reported in lifted inference literature. While the overall conclusions remain the same, we found that the KL to be too forgiving by hiding grossly inaccurate probabilities for some outputs.

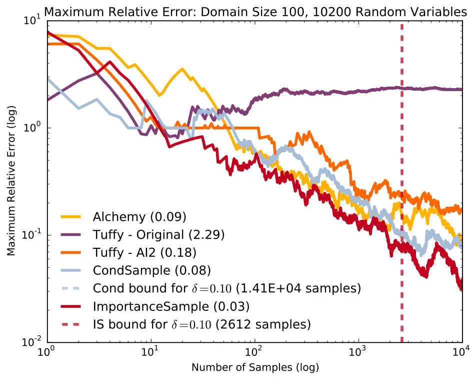
[7]It needs to compute a probability for each combination of three cardinalities, $|Smokes \cap Drinks|, |Smokes - Drinks|, and |Drinks - Smokes|$.



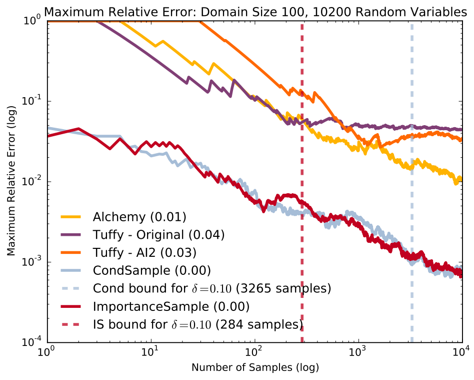
(a) Query 1, Symmetric



(b) Query 2, Symmetric



(c) Query 1, Asymmetric Unary



(d) Query 2, Asymmetric Unary
Figure 2: Maximum Relative Error on the Smokers MLN

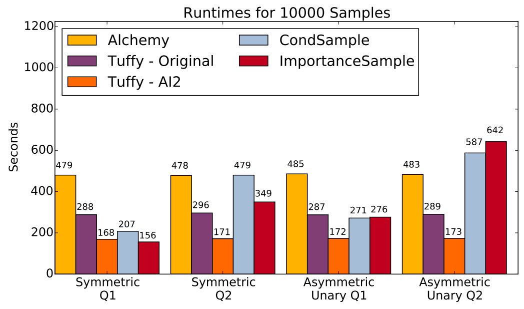


Figure 3: Absolute Runtimes for 10,000 samples.

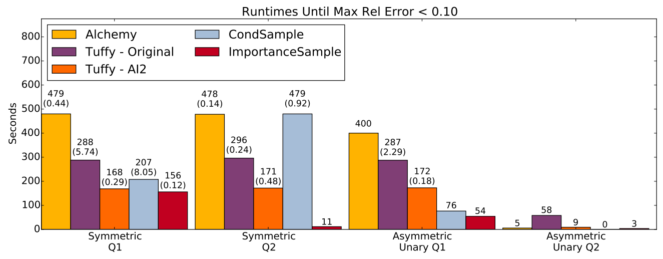


Figure 4: Runtime as a function of accuracy. Times shown are the point when systems first achieve a maximum relative error of 0.10 and maintain this error for at least ten iterations. Some systems fail to achieve this error bound within 10,000 iterations; these bars are annotated with the final maximum error achieved after 10,000 samples.

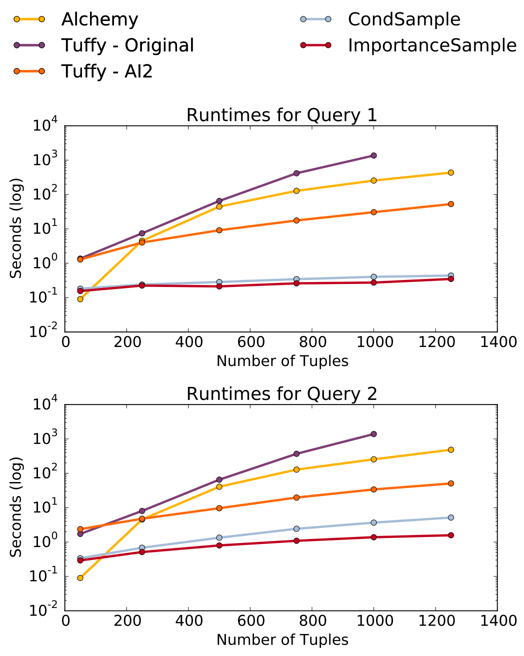


Figure 5: Runtimes for 100 iterations as a function of domain size, keeping a fixed fan-out on the binary *Friends* relation of 3 friends per individual, chosen randomly.

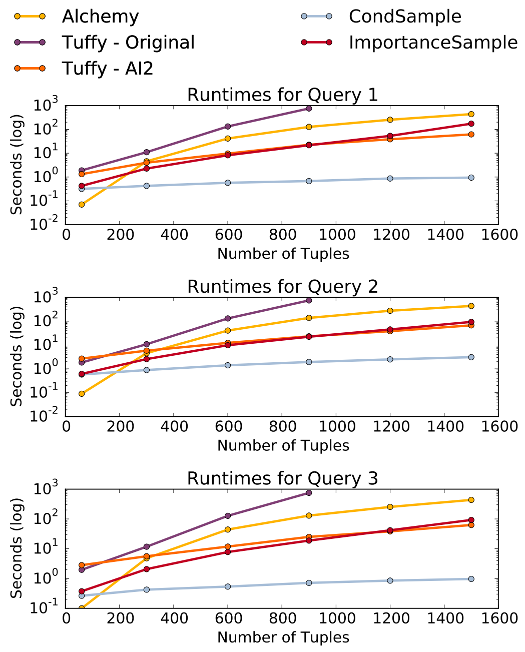


Figure 6: Runtimes for 100 iterations as function of domain size on the Smokers-Drinkers MLN using sparse, asymmetric data. The startup cost of computing the importance distribution increases for ImportanceSample on this more complex set of constraints.

Tuffy-AI2 implements certain logical simplifications, keeping the size of the Smokers-Drinkers network equivalent to that of the Smokers network, improving its performance relative to ImportanceSample.

5.4 Impact of Optimizations

As we developed our system we progressively added optimizations, sometimes replacing rather naive first implementations. We report their effect in Figure 7 on Query 1, over a domain size 100 and asymmetric, but dense data. *Generic constants:* We actually started by computing a non-Boolean query $Q(x)$ as in theory textbooks: for each constant a in the domain, compute $\mathbf{P}(Q[a/x]|\Gamma)$: switching to a safe plan that computes all output probabilities in one query improved the runtime by more than two orders of magnitude. All runtimes in the figure use generic constants. *DNF:* Our first implementation used standard safe plans for UCQ [34], by expressing $\mathbf{P}(\Gamma) = 1 - \mathbf{P}(\neg\Gamma)$. Since $\mathbf{P}(\neg\Gamma)$ is very close to 1.0, it required Postgres's numeric data type to achieve sufficient precision. This first column shows this runtime. *CNF:* Implementing specific operators for CNF reduced the runtime to the second column. Here we used logarithm to express the product aggregate in terms of sum. *Product:* Replacing the log-sum-exp with a UDA for product reduced the runtime by 35% (third column). *QRel:* if the query happens to be the sampled relation, then we can avoid computing the second query $\mathbf{P}(Q \wedge \Gamma|\mathbf{T}^{D_i})$, but instead simply check whether $\mathbf{T}^{D_i} \models Q$. This reduces the runtime by half. *Sparse:* Finally, we show the benefit of adding extra logic to the SQL query to omit tuples with probability 0. Note that the dataset used here is dense: the savings comes entirely from the sampled *Smokes* relation. Significant additional savings occur on sparse data.

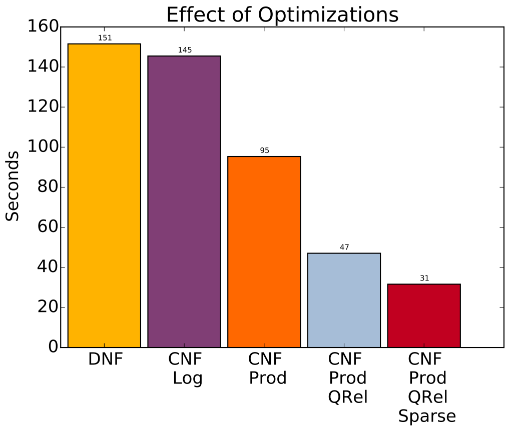


Figure 7: The runtime for 1,000 iterations of SlimShot with progressively more optimizations enabled.

Domain Size (Number of Variables)	Runtime in Seconds
3 (15)	1.3
5 (35)	1.8
8 (80)	11.0
10 (120)	205.5
15 (255)	Did not finish

Table 5: Runtimes for Sentential Decision Diagrams on Query 1 over the Smokers MLN with symmetric data

5.5 Other Weighted Model Counters

Since our approach reduces the query evaluation problem on MLNs to weighted model counting, as a ratio of two probabilities $\mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma)$, we also attempted to compare SlimShot with state of the art general purpose Weighted Model Counting (WMC) systems.

A state of the art system for exact weighted model counting uses Sentential Decision Diagrams [9] (SDDs). They arose from the field of knowledge compilation, and compile a Boolean formula into circuit representations s.t. WMC can be done in linear time in the size of the circuit. SDDs have state-of-the-art performance for many tasks in exact weighted model counting. We use SDD v1.1 [32] and report runtime results in Table 5. While SDDs have been reported in the literature to scale to much larger instances, they fared worse on the formulas resulting from grounding MLNs.

A state of the art system for approximate weighted model counting is WeightMC [5], which is part of a recent and very promising line of work [14, 5]. We downloaded WeightMC from [36], but unfortunately, we were only able to run it on a domain size of 3 before experiencing time-out errors.

Technical difficulties aside, general-purpose WMC tools do not appear well-suited for MLN inference: to approximate the ratio $\mathbf{P}(Q[a/x] \wedge \Gamma)/\mathbf{P}(\Gamma)$ accurately requires extremely accurate approximations of each quantity individually, and one has to repeat this for every possible query answer a.

5.6 Discussion

SlimShot is the only MLN system that can provide guaranteed accuracy: we have validated its accuracy on several symmetric and unary-symmetric datasets (several omitted for lack of space). The theoretical stopping condition is sometimes overly conservative. SlimShot's runtime performance per sample is comparable to other systems, however SlimShot converges much faster than the other systems. The main limitation of SlimShot is its dependency on the structure of logical formula of the MLN. The runtime suffers if two relations need to be sampled instead of one (while still being competitive). At an extreme, one can imagine an MLN where all relations need to be sampled, in which case SlimShot's performance would degenerate.

6. CONCLUSION

We have described SlimShot, a system that computes queries over large Markov Logic Networks. The main innovation in SlimShot is to combine sampling with lifted inference. This reduces the sample space, and thus reduces the variance, and also enables two additional techniques: estimation of a conditional probability and importance sampling. The lifted inference is performed entirely in the database engine, by evaluating safe plans. We have described several optimizations that improve the performance of safe plans. Our experiments have shown that SlimShot returns significantly better results than other MLN engines, at comparable or better speed.

One limitation of SlimShot is that it only works if the query and constraint can be made safe by determinizing a small number of relation names. In extreme cases that use a single relational predicate name, like the transitivity constraint $E(x, y) \wedge E(y, z) \Rightarrow E(x, z)$, SlimShot degenerates to a naive Monte Carlo evalution. Future work includes studying how SlimShot can be extended to such cases, for example by partitioning the database.

7. REFERENCES

[1] S. Abiteboul, O. Benjelloun, and T. Milo. The active XML project: an overview. *VLDB J.*, 17(5):1019–1040, 2008.

[2] `http://alchemy.cs.washington.edu/`.

[3] J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[4] R. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *IJCAI*, pages 1319–1325. Citeseer, 2005.

[5] S. Chakraborty, D. Fremont, K. Meel, S. Seshia, and M. Vardi. Distribution-aware sampling and weighted model counting for SAT. In *AAAI*, pages 1722–1730, 2014.

[6] K. Chang, W. Yih, B. Yang, and C. Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*, pages 1568–1579, 2014.

[7] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation (extended abstract). In *FOCS*, pages 142–149, 1995.

[8] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[9] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, pages 819–826, 2011.

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

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

[12] P. M. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[13] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, pages 601–610, 2014.

[14] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *ICML*, volume 28, pages 334–342. JMLR.org, 2013.

[15] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10, 2011.

[16] https://www.freebase.com/.

[17] E. Gribkoff and D. Suciu. SlimShot: Probabilistic inference for web-scale knowledge bases, 2015. Technical Report.

[18] E. Gribkoff, G. Van den Broeck, and D. Suciu. Understanding the complexity of lifted inference and asymmetric weighted model counting. In *UAI*, pages 280–289, 2014.

[19] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[20] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. *ECCC*, 17:72, 2010.

[21] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[22] A. K. Jha and D. Suciu. Probabilistic databases with markoviews. *PVLDB*, 5(11):1160–1171, 2012.

[23] R. Karp and M. Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64, 1983.

[24] D. Krompaß, M. Nickel, and V. Tresp. Querying factorized probabilistic triple databases. In *ISWC*, volume 8797, pages 114–129, 2014.

[25] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.

[26] F. Niu, C. Zhang, C. Re, and J. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In *Workshop on Searching and Integrating New Web Data Sources*, volume 884, pages 25–28. CEUR-WS.org, 2012.

[27] D. Olteanu, J. Huang, and C. Koch. SPROUT: lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, pages 640–651, 2009.

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

[29] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, pages 458–463, 2006.

[30] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996.

[31] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.

[32] `http://reasoning.cs.ucla.edu/sdd/`.

[33] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, pages 1094–1099, 2008.

[34] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[35] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: Exploiting random walk strategies. In *IAAI*, 2004.

[36] `http://www.cs.rice.edu/CS/Verification/Projects/WeightGen/`.

[37] C. Zhang, V. Govindaraju, J. Borchardt, T. Foltz, C. Ré, and S. Peters. Geodeepdive: statistical inference using familiar data-processing languages. In *SIGMOD*, pages 993–996, 2013.